Design and Verification of an Application Specific Integrated Circuit (ASIC)
Senior Project II

Kevin Cao, Whitley Forman, Dhruvit Naik,
Zachary Nelson* and Julie Swift
*Team Leader

Dr. Orlando Hernandez and Dr. Larry Pearlstein

May 2016

Fulfillment Page

Submitted to the Faculty of the Electrical and Computer Engineering Department
of The College of New Jersey

Written and Researched by:

Kevin Cao

Whitley Forman

Dhruvit Naik

Zachary Nelson

Julie Swift

In partial fulfillment of the Bachelor of Science degree in Electrical and Computer Engineering

Acknowledgements

The team would like to acknowledge Ian Patel's work on helping to set up the realistic FPGA test environment. Ian wrote the PSoC microcontroller code to generate an I2S audio stream and capture it back from the FPGA. He also wrote Matlab code that produced different types of waveforms in I2S format. We would also like to thank MOSIS for sponsoring the chip fabrication aspect of the project through the MOSIS Educational Program (MEP).

Abstract

The goal of this project is to gain experience in VLSI design by designing a chip that will process digital streaming audio data. More specifically, we will implement a 512-tap digital finite impulse response (FIR) filter, which will be applied to an input stream in order to create an output stream. We used the I2C protocol to allow a host to control the chip and the serial I2S protocol for transferring digital audio streams in and out.

Our hardware design was represented using Verilog register-transfer level (RTL) code. Development has been done using Xilinx ISE Design Suite 14.7. Test-benches were also designed and implemented using Verilog. The end goal of the project is to implement the design on a field-programmable gate array (FPGA). We will bring up our FPGA design with a realistic environment including an audio source, audio sink, and a microcontroller for reading and writing registers. We also plan to send a simple CMOS integrated circuit design for fabrication by MOSIS that will help us gain experience in physical chip design and prepare future groups to fabricate our full design.

Keywords: Application-specific integrated circuit (ASIC), Very large scale integration (VLSI) I2S, I2C, Digital filtering

Table of Contents

List of Tables 5
List of Illustrations 6
Nomenclature 8
Introduction 9
Specifications 11
Chapter 1: Background – Z. Nelson 12
Chapter 2: System Design – Z. Nelson 14
Chapter 3: I2S Interface – K. Cao 17
Chapter 4: Digital Filtering – D. Naik 37
Chapter 5: Register Block – J. Swift 52
Chapter 6: I^2C Slave Interface – W. Forman 61
Chapter 7: Simulation/Verification – K. Cao and D. Naik 75
Chapter 8: FPGA Implementation – Z. Nelson 79
Chapter 9: EDA Tools and Physical Design – D. Naik, J. Swift, and K. Cao 85
Chapter 10: Conclusion – Z. Nelson 89
References 90
Appendix A: Project Overview 91
Appendix B: Management 97
Appendix C: Source Code 129
Appendix D: Test Benches 346
Appendix E: Industry Specifications 581

List of Tables

Table 1.1: Comparison of MOSIS Academic Account Types 12
Table 3.1: Interface Signals for I2S Input 20
Table 3.2: Interface Signals for Synchronizer Sub-Module 24
Table 3.3: Interface Signals for Deserializer Sub-Module 25
Table 3.4: Interface Signals for BIST Generator Sub-Module 27
Table 3.5: Interface Signals for Multiplexer Sub-Module 28
Table 3.6: Interface Signals for FIFO Sub-Module 29
Table 3.7: Interface Signals for I2S Output 31
Table 3.8: Interface Signals for Serializer Sub-Module 35
Table 4.1: Interface Signals for Filter 39
Table 4.2: Interface Signals for Finite State Machine 41
Table 4.3: Interface Signals for Filter Storage Module 42
Table 4.4: Interface Signals for Mux Module 43
Table 4.5: Interface Signals for Accumulator Module 44
Table 4.6: Interface Signals for Shift Round Truncate Module 45
Table 5.1: Register Mapping of Each Bit in the Address it is Stored 54
Table 6.1: I^2C Register table 66
Table 8.1: Verilog Signal to FPGA Mapping 79

List of Illustrations

Fig. I.1: Process of Designing a System on a Chip 9
Fig. 1.1: Multi-Project Wafer 12
Fig. 2.1: Top Level Drawing of Chip 14
Fig. 2.2: Detailed Top-Level Drawing of Chip 15
Fig. 2.3: FPGA Testing Environment 16
Fig. 3.1: Overall Chip Diagram with I2S Interfaces Highlighted 18
Fig. 3.2: I2S Input Block Diagram 19
Fig. 3.3: Sample of I2S Input Interface Simulation 21
Fig. 3.4: Sample of Synchronizer Simulation 25
Fig. 3.5: Sample of Deserializer Simulation 26
Fig. 3.6: Sample of BIST Generator Simulation 27
Fig. 3.7: Sample of Multiplexer Simulation 28
Fig. 3.8: Sample of FIFO Simulation 30
Fig. 3.9: Block Diagram of I2S Output Interface 30
Fig. 3.10: Sample of I2S Output Interface Simulation 32
Fig. 3.11: Sample of Serializer Simulation 36
Fig. 4.1: Frequency Response of Filter Structures 37
Fig. 4.2: Chip Diagram with Filter Highlighted 38
Fig. 4.3: Top Level Block Diagram of Filter 39
Fig. 4.4: Detailed Block Diagram of Filter 40
Fig. 4.5: Finite State Machine: States and Transitions 44
Fig. 4.6: Impulse Response of 512-Tap Low-Pass Filter 46
Fig. 4.7: Simulation of Accumulator 46
Fig. 4.8: Simulation of Filter Storage Module Test 1 47
Fig. 4.9: Simulation of Filter Storage Module Test 2 47
Fig. 4.10: Simulation of Mux 47
Fig. 4.11: Simulation of Filter Test 1 48
Fig. 4.12: Simulation of Filter Test 2 48
Fig. 4.13: Simulation of Filter Test 3 49
Fig. 4.14: Simulation of Filter Test 4 49
Fig. 4.15: Simulation of Filter Test 5 50
Fig. 5.1: Chip Diagram with Register Highlighted 52
Fig. 5.2: Internal Micro-Architecture of Register 53
Fig. 5.3: Top Level Interface Design of reg.v 53
Fig. 5.4: The Start of the Test Fixture Simulation 57
Fig. 5.5: Bits trig_i2si_fifo_overrun_clr and trig_i2so_fifo_underrun_clr are Triggered 58
Fig. 5.6: Bits are Triggered in Second Testbench 58
Fig. 5.7: Test data properly being written from the I2C into the Register Part 1 59
Fig. 5.8: Test data properly being written from the I2C into the Register Part 2 59
Fig. 6.1: I2C Chip Diagram Highlight 61
Fig. 6.2: I^2C Full Coefficient Time Load Calculation 62
Fig. 6.3: I^2C block diagram in 64
Fig. 6.4: Open Drain Schematic 65

Fig. 6.5: I^2C State Machine Flow Diagram 68
Fig. 6.6: I^2C Previous block diagram versions 68
Fig. 6.7: I^2C Deserializer Block Diagram 70
Fig. 6.8: I^2C Sequencer Block Diagram 71
Fig. 6.9: I^2C Serializer Block Diagram 72
Fig. 6.10: I^2C Start, Stop Deserialize ISIM 73
Fig. 6.11: I^2C Isim Full Register Write Transaction (Truncated) 73
Fig. 6.12: I^2C Read Register Transaction 74
Fig. 7.1: Lowpass Filter: Impulse Response 76
Fig. 7.2: Lowpass Filter: Square Wave Samples 1000 - 2000 76
Fig. 7.3: Highpass Filter: Impulse Response 77
Fig. 7.4: Highpass Filter: Square Wave Samples 2000 - 3000 77
Fig. 7.5: Average Filter: Impulse Response 78
Fig. 7.6: Average Filter: Square Wave Samples 1000 – 2000 78
Fig. 8.1: FPGA Testing Set Up 82
Fig. 8.2: FPGA Waveforms (Pass-Through Filter) 83
Fig. 9.1: Project Manager Window 85
Fig. 9.2: New Project Wizard – Screen 1 86
Fig. 9.3: New Project Wizard – Screen 2 86
Fig. 9.4: 38 x 40 μm 25Ω Resistor 87
Fig. 9.5: 40 x 40 μm Spiral Inductor 88
Fig. 9.6: 38 x 40 μm capacitor 88

Nomenclature

ASIC: Application Specific Integrated Circuit
BIST: Built in Self-Test
CIF: Caltech Intermediate Form
DIP: Dual in-line package
EDA: Electronic Design Automation
FPGA: Field Programmable Gate Array
FSM: Finite-State Machine
GDSII: Graphic Data System II
I2C: Inter-Integrated Circuit
I2S: Integrated Interchip Sound
IC: Integrated Circuit
LSB: Least Significant Bit
MEP: MOSIS Educational Program
MPW: Multi-Project Wafer
MSB: Most Significant Bit
OCP: Open Cavity Plastic
PCB: Printed Circuit Board
RO: Read Only
RTL: Register-Transfer Level
RTR: Ready to Receive
RTS: Ready to Send
RW: Read/Write
WO: Write Only
XFC: Transfer Complete
VLSI: Very Large Scale Integration

Introduction

The goal of this project is to design a chip that is capable of processing digital streaming audio data and apply a 512-tap filter to the input stream. The design was implemented and tested on a Nexys 4 Artix-7 FPGA board. We also are close to submitting a simple circuit to MOSIS for fabrication in order to gain experience with using EDA tools. Two senior project groups for next year will be using certain aspects of our project. One group will be taking the Verilog code that we designed and tested and will fabricate a chip out of it. This group can also use the information that we recorded about using the EDA tools. The other group will be designing a RISC-V processor that may reuse some of the code that we have written. It should be noted that the terms VLSI and ASIC will be used interchangeably throughout the report and refer to designing the chip.

In industry, some of the advantages of an FPGA are it can be reprogrammed, there is no manufacturing involved, and it is as simple as just downloading code onto a board. Some of the advantages of an ASIC include being a fully custom design so that the device is manufactured to the designer's specifications, a lower unit cost when producing in high volume, a smaller chip since there are no unwanted components included, higher clock rates, and lower power dissipation. ASICs are used ubiquitously throughout industry (e.g. smartphones, laptops) and a major part of our senior project was learning the steps for producing one. We chose our application to be audio processing because digital audio filtering has a wide range of real world applications. We are implementing an extremely high order filter that is capable of producing almost any type of frequency response. The chip's application is relatively simple because the bulk of the time was spent going through the chip design process since this is the first chip any of the students have designed.

Fig. I.1: Process of Designing a System on a Chip

Even though we will not be following every step, Fig. I.1 shows the general process for designing a chip. The first task is system design where the requirements of the system are documented and the major functionality of the chip is described. The second task is RTL design

which involves writing code in a hardware description languages (HDL) such as Verilog in order to create a high-level representation of the chip. The next step is logic synthesis which involves turning the RTL design into a netlist of logic gates. Floor planning involves determining the location of the major blocks on the IC schematic. Place and optimization is placing all of the electronic components, circuitry, and logic elements into a limited amount of space in an effective manner. Routing is connecting all of the different components and must follow the rules and limitations of the fabrication process. This step is mostly done automatically by an EDA tool but sometimes has to be done manually. It should be noted that testing and verification will be performed throughout the entire process. The last step is submitting a verified design to be fabricated (commonly in the form of a GSDII file).

Specifications

General:

- Maximum Clock Rate: 100 MHz
- Clock frequency will be a minimum of 1200 times the audio sampling rate

I2S:

- I2S input and output interfaces comply with I2S standard, included here in Appendix E
- Maximum Serial Clock Rate: 1.44 MHz
- Audio input sample rates ranges of 8 kilosamples/sec - 48 kilosamples/sec
- Digital audio bit clock and word select lines will be controlled from I2S master, which can be our chip, or the external I2S source
- Audio input and output must have same sample rate – our chip will be the time-base master on the output I2S interface
- Audio input and output will be two 16 bit channels (i.e. one stereo pair)

Filter:

- Filter Design: FIR Filter
- 512-tap filter, 16-bit coefficients, all independently settable
- Programmable filter coefficients to achieve different filter types
- Maintain integer headroom of 4 bits

I2C:

- 8-bit slave address space
- 12-bit register address space
- Data transfer rate of up to 400 kbits/second desired (both I2C standard and fast modes supported)
- Slave only capability
- Write operation (Burst write functionality)
- Read operation (Single read, optional burst read request)
- User selectable slave address with strap pins
- Simple strobe interface to register block (read and write)

Register:

- Simple strobe interface to I2C block
- Provides read/write access to control/status registers, including the following
  - Source select bit (I2S vs. BIST)
  - Filter bypass bit (pass through or filter input streams)
  - 512 16-bit filter coefficients
  - Overflow/saturation detector
  - Audio FIFO overrun/underrun

Chapter 1: Background – Z. Nelson

1.1 Who is MOSIS?

The simple chip that we are designing will be produced by MOSIS, at no cost to The College of New Jersey. MOSIS was the first well-known multi-project wafer (MPW) service, and was established by DARPA (Defense Advanced Research Projects Agency) in 1981. The acronym MOSIS stands for Metal Oxide Semiconductor Implementation Service and the company has processed over 60,000 IC designs over the last 30 years [1]. The MOSIS Service is known for MPWs, which is how they are able to keep the cost of fabrication low. A MPW is when multiple IC designs are shared on a single wafer and an illustration of this concept in shown in Figure 1.1. This means that designs from private companies and students could be on the same wafer. The reasoning behind this approach is that the cost of fabrication can be kept at a reasonable price if the cost of mask making, wafer fabrication and assembly are shared throughout multiple projects. This idea of a MPW is also attractive because designers can create a prototype of their design without making a huge investment. It should also be noted that MOSIS offers using a single project for a wafer (dedicated run) for all processes and can start the fabrication at any time.

Fig. 1.1: Multi-Project Wafer

1.2 Fabrication Process

MOSIS offers three different options for an accredited college or university to use. These options are registering for a Commercial account, a MOSIS Educational Program (MEP) Instructional account, or a MOSIS Educational Program (MEP) Research account. A comparison of these three accounts is listed on MOSIS's website and is also shown in Table 1.1. Our advisors registered for a MEP Instructional account and we will be using a GlobalFoundaries 130 nm 8RF process.

Table 1.1: Comparison of MOSIS Academic Account Types

Topic	Commercial	MEP Instructional	MEP Research
Primary Purpose	Customers pays for fabrication and packaging	Classroom instruction	Unfunded research
Available Processes	All	1. ON Semi $0.50\mu m$ CMOS (C5N) 2. GlobalFoundaries 180 *nm* CMOS (7HV)	1. ON Semi $0.50\mu m$ CMOS (C5N) 2. GlobalFoundaries 130 *nm* SiGe BICMOS (8HP) 3. GlobalFoundaries 130 *nm* CMOS (8RF-DM)
Size Limits	No restrictions	5 deliverable parts of a project no larger than 1.5 mm x 1.5 mm	Less than $16mm^2$
Number of Submissions per Year	Unlimited	Annual request subject to review	One submission per academic year per institution after approval of proposal
Run Restrictions	All MPW runs except for MEP-only	MEP-only and space available for COM runs	Space available for COM runs
IP Access through MOSIS	Yes	No	Yes
Fabrication Costs	Customer pays fabrication cost	Free	Free
Packaging	No restrictions, customer pays	Free ceramic and OCP packaging; lids cannot be sealed. Fully encapsulated packaging not available.	No restrictions, customer pays

1.3 Submitting a Design to MOSIS

When submitting a MPW run, MOSIS has specified certain steps that need to be taken in order to submit a design. The first step is to submit a new project request by logging onto their website. The designer then needs to assign their Export Control Classification Number (ECCN) before they make a fabrication request. Next, a fabrication request needs to be made specifying the process that will be used. The last step is to submit that actual design layout to MOSIS in either Caltech Intermediate Form (CIF) or Graphic Data System II (GSDII) format. Both of these formats are used to describe the layout of the integrated circuit and will be generated by an EDA tool.

Chapter 2: System Design – Z. Nelson

The system design of our project started in April 2015 when we wrote a senior project proposal. The proposal discussed the basic functionality that should be accomplished and how the tasks should be broken down. A top level block diagram of the chip is shown in Fig. 2.1. The chip was broken down into five major modules. These modules included the I2S input interface, I2S output interface, filter, register block, and I2C slave interface. Kevin's responsibility was the I2S interface, Dhruvit's was the filter, Julie's was the register, and Whitley's was the I2C interface. Over the 2015 summer, each team member worked on creating a specification document for their module.

Fig. 2.1: Top Level Drawing of Chip

The main purpose of the I2S input interface was to convert a serial audio stream into left and right 16-bit parallel data. This module also included a Built in Self-Test (BIST) feature that would generate a predefined sawtooth signal as the audio data stream. After the audio stream was converted into parallel data, it was put through a 512-tap FIR filter in the filter module. The filter audio signal was then converted back to the I2S serial standard in the I2S output interface. The combination of these three modules was how data flowed within our chip.

Data Flow: I2S Input Interface → Filter → I2S Output Interface

The control flow of the chip started with the I2C slave interface module reading or writing register values to the chip. The register block is where the actual filter coefficient values were stored along with other information such as chip info, control bits, and status bits. The filter block would then read the 512 filter coefficients from the register block in order to implement the FIR filter.

Control Flow: I2C Slave Interface → Register Block → Filter

The next step in the system design process was creating block documents that stored the details of each module. These block documents can be found on the GitHub repository. After the interfaces for each module were agreed upon, we created a detailed schematic of the top-level module of our chip (refer to Fig. 2.2).

Fig. 2.2: Detailed Top-Level Drawing of Chip

The PSoC microcontroller will be used to generate the audio stream and provide I2C read/write access for the chip. Since we will create the audio stream in the PSoC environment, we should know exactly what the output stream will be based on the filter coefficients we are using. The microcontroller will also have the ability to read and write filter coefficients to the chip. This is an important feature of our project because the user will be able to manually set and check the

values of all 512 filter coefficients. A diagram of the PSoC microcontroller interfaced with the FPGA is shown in Fig. 2.3.

Fig. 2.3: FPGA Testing Environment

Chapter 3: I2S Interface – K. Cao

3.1 Introduction

Integrated Interchip Sound (I2S) buses are used to pass digital audio data into our chip for processing, and to carry processed digital audio streams that are outputted by our chip. I2S is an electrical serial bus interface standard that is used for connecting digital audio devices together. I2S is used to communicate pulse-code modulation (PCM) audio data between integrated circuits in an electronic device. PCM is a standard format for digital audio in computers. It is a method used to digitally represent sampled analog signals.

The **i2s_in** block is responsible for receiving the I2S signals that are passed to our chip, and converting this to a natural parallel PCM format, suitable for processing by the filter block. The **i2s_out** block is responsible for converting parallel PCM data produced by the filter block to a serial I2S stream, for output by the chip.

3.2 I2S Bus Specification

The I2S protocol contains three lines: serial data bus (SD), a bit clock (SCK), and word select (WS). These three lines are used to reduce the number of pins required and to keep wiring simple. The SCK signal pulses once for each discrete bit of data on the data line and its frequency can be calculated by multiplying the sample rate, number of bits per channel, and the number of channels. The WS signal informs the chip whether the left or right channel is being currently sent. I2S allows a device to have two channels that send data over the same data line. In this chip when WS is 0, data is being sent over the left channel and when WS is 1, data is being sent over the right channel. The WS signal is a 50% duty signal that has the same frequency as the audio sample frequency.

The SD signal is a serial representation of a two's complement signed value, and is transmitted with the most significant bit (MSB) first. The transmitter and receiver may have different natural word lengths. The transmitter and receiver doesn't need to know how many bits the other can handle. The least significant data bits are set to 0 when the receiver's word length is greater than the transmitter word length. However, if the sent bits are larger than the word length of the receiver, then the bits after the least significant bit are ignored. The MSB is always located in the same position, however, the least significant bit (LSB) position depends on the word length. The MSB of the new word is also always sent one clock period after the WS signal changes. The serial data being transmitted can either be synchronized with either the trailing or leading edge of the clock signal. In this implementation the serial data is synchronized with the leading edge. However, the serial data must be latched by the receiver on the leading edge of serial clock signal. The WS signal is also changes with the leading edge of the serial clock and is set to change one clock period before the MSB is transmitted. This early transition allows the slave transmitter to derive synchronous timing of the serial data.

3.3 I2S Interfaces

The I2S interfaces for the custom chip were split into two Verilog modules, **i2s_in** and **i2s_out**. The **i2s_in** module represented the receiver and handled all data being inputted into the I2S interface, and the **i2s_out** module represented the transmitted and handled all data being outputted. The following figure displays the overall block diagram of the chip, with the I2S interfaces highlighted.

Fig. 3.1: Overall Chip Diagram with I2S Interfaces Highlighted

3.4 I2S Input Interface

The I2S input interface handles all input audio and relays it to the filter for processing. It is primarily responsible for reading in serialized data and converting it back into parallel data. The I2S Input interface is made of five different sub-modules, the synchronizer, deserializer, Audio Built-In Self-Test (BIST) generator, mux, and FIFO. The following figure displays the block diagram of the I2S input interface with all of its signals and sub-modules.

Fig. 3.2: I2S Input Block Diagram

The following table displays all the input and output signals the I2S input interface interacts with. The table provides the name, direction, bit size, and a short description of each signal.

Table 3.1: Interface Signals for I2S Input

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rst_n	in	1	Reset
I2S Input Interface			
inp_sck	in	1	Digital Audio Bit Clock
inp_sd	in	1	Digital Audio Serial Data
inp_ws	in	1	Word Select
Control/Status Register Fields			
trig_fifo_overrun_clr	in	1	Reset FIFO Overrun
ro_fifo_overrun	out	1	Input Audio FIFO Overrun
rf_i2si_en	in	1	Serializer Enable Bit
rf_mux_en	in	1	Built in Self-Test (BIST)
rf_bist_start_val	in	12	Start Value
rf_bist_inc	in	8	Increment
rf_bist_up_limit	in	12	Upper Limit
Streaming Audio Interface with Filter Block			
i2si_rtr	in	1	Ready to Receive
i2si_rts	out	1	Ready to Send
i2si_data	out	32	Output Digital Audio

The signal inp_sck is the digital audio bit clock of the I2S interface, inp_sd is the serial data that is being sent in, and inp_ws is the word select signal that informs the module whether the left or right audio channel is being received. The signals i2si_rtr and i2si_rts are the handshake interface signals interacting with the filter block. They inform the I2S input interface when the filter block is ready to receive data, and inform the filter block when the I2S input interface is read to send

data. The i2si_data signal contains the 32 bit words that will be sent to the filter block for processing. The rf_i2si_en signal informs the I2S input block to start converting data from serial into parallel. The rf_mux_en signal is the select bit that informs the I2S input interface either to output the deserialized data or the BIST data to the filter block. The signals rf_bist_start_val, rf_bist_inc, and rf_bist_up_limit are used as parameters to create a sawtooth wave to test the chip. The signal ro_fifo_overrun indicates whether data is written to the block faster than it can be read. The trig_fifo_overrun signal indicates whether or not to clear the ro_fifo_overrun signal.

The I2S input interface was verified by inputting a list of 32 bit words into the inp_sd signal line. The i2si_data signal was then observed to see if it outputted the same words inputted. To make verification easier, two outputs files were created with the input and output words. A brief summary at the bottom of the text files confirmed if the test was successful or not. The inputted words were tested with random 32 bit values, and values coming from the BIST generator. It was observed that the same data inputted is also outputted within the block and appears to be working. The select bit for the multiplexer was also toggled to verify that the data outputted came from either the BIST or input serial data. It was confirmed that the data outputted was dependent on the select bit and the block was functioning properly. The i2si_rts signal line was also observed. The signal is supposed to pulse after each half word within the block. When observing the test benches this can be seen functioning correctly. Additional testing was conducted to test the underrun and overrun signals. Testing showed that both signals trigger under the correct conditions, and the signals can be cleared with the proper reset signals.

Fig. 3.3: Sample of I2S Input Interface Simulation

Sample of I2S Input Interface Text File

Input: d6095663	---	Output: d6095663	---	Pass
Input: 7b0d998d	---	Output: 7b0d998d	---	Pass
Input: 84655212	---	Output: 84655212	---	Pass
Input: e301cd0d	---	Output: e301cd0d	---	Pass
Input: f176cd3d	---	Output: f176cd3d	---	Pass
Input: 57edf78c	---	Output: 57edf78c	---	Pass
Input: e9f924c6	---	Output: e9f924c6	---	Pass
Input: 84c5d2aa	---	Output: 84c5d2aa	---	Pass
Input: f7e57277	---	Output: f7e57277	---	Pass

Input: d612db8f	---	Output: d612db8f	---	Pass
Input: 69f296ce	---	Output: 69f296ce	---	Pass
Input: 7ae84ec5	---	Output: 7ae84ec5	---	Pass
Input: 495c28bd	---	Output: 495c28bd	---	Pass
Input: 582d2665	---	Output: 582d2665	---	Pass
Input: 6263870a	---	Output: 6263870a	---	Pass
Input: 22802120	---	Output: 22802120	---	Pass
Input: 45aacc9d	---	Output: 45aacc9d	---	Pass
Input: 3e96b813	---	Output: 3e96b813	---	Pass
Input: 380dd653	---	Output: 380dd653	---	Pass
Input: dd6b2ad5	---	Output: dd6b2ad5	---	Pass
Input: 4a023eae	---	Output: 4a023eae	---	Pass
Input: e91d72cf	---	Output: e91d72cf	---	Pass
Input: 4923650a	---	Output: 4923650a	---	Pass
Input: 0aca4c3c	---	Output: 0aca4c3c	---	Pass
Input: bdf2618a	---	Output: bdf2618a	---	Pass
Input: b34134d8	---	Output: b34134d8	---	Pass
Input: f3781289	---	Output: f3781289	---	Pass
Input: 0deb65b6	---	Output: 0deb65b6	---	Pass
Input: f9c613ae	---	Output: f9c613ae	---	Pass
Input: 02bcdd2a	---	Output: 02bcdd2a	---	Pass
Input: 9a0bbe71	---	Output: 9a0bbe71	---	Pass
Input: 4185554f	---	Output: 4185554f	---	Pass
Input: 603b333a	---	Output: 603b333a	---	Pass
Input: 327e4b15	---	Output: 327e4b15	---	Pass
Input: 9bf14bd9	---	Output: 9bf14bd9	---	Pass
Input: 0762fb4c	---	Output: 0762fb4c	---	Pass
Input: 559fa18f	---	Output: 559fa18f	---	Pass
Input: a9f860b7	---	Output: a9f860b7	---	Pass
Input: 569f945c	---	Output: 569f945c	---	Pass
Input: c05b3789	---	Output: c05b3789	---	Pass
Input: 32493ed0	---	Output: 32493ed0	---	Pass
Input: c0d7fc51	---	Output: c0d7fc51	---	Pass
Input: 2f967f0c	---	Output: 2f967f0c	---	Pass
Input: cec2edc8	---	Output: cec2edc8	---	Pass
Input: 5a77ed3d	---	Output: 5a77ed3d	---	Pass
Input: db12007e	---	Output: db12007e	---	Pass
Input: 816de739	---	Output: 816de739	---	Pass
Input: 8f1ff6d3	---	Output: 8f1ff6d3	---	Pass
Input: 2f858878	---	Output: 2f858878	---	Pass
Input: 595b4b49	---	Output: 595b4b49	---	Pass
Input: ae3faf2a	---	Output: ae3faf2a	---	Pass
Input: 63583886	---	Output: 63583886	---	Pass
Input: 0c8ef29c	---	Output: 0c8ef29c	---	Pass
Input: 99fabc26	---	Output: 99fabc26	---	Pass
Input: 1773b4a3	---	Output: 1773b4a3	---	Pass
Input: a82f92b3	---	Output: a82f92b3	---	Pass
Input: 565f0d44	---	Output: 565f0d44	---	Pass
Input: fdf736cb	---	Output: fdf736cb	---	Pass

Input: 1ae6db5a	---	Output: 1ae6db5a	---	Pass
Input: c129e2ed	---	Output: c129e2ed	---	Pass
Input: 6cdab665	---	Output: 6cdab665	---	Pass
Input: e2b594df	---	Output: e2b594df	---	Pass
Input: 1979ff44	---	Output: 1979ff44	---	Pass
Input: 0cd00b2a	---	Output: 0cd00b2a	---	Pass
Input: adabcf0e	---	Output: adabcf0e	---	Pass
Input: addcbc9a	---	Output: addcbc9a	---	Pass
Input: 72fd02c3	---	Output: 72fd02c3	---	Pass
Input: ed56e94e	---	Output: ed56e94e	---	Pass
Input: 7667340a	---	Output: 7667340a	---	Pass
Input: b9b68a38	---	Output: b9b68a38	---	Pass
Input: 8779c4b8	---	Output: 8779c4b8	---	Pass
Input: bf945d93	---	Output: bf945d93	---	Pass
Input: 2c04ca59	---	Output: 2c04ca59	---	Pass
Input: 69dbde4d	---	Output: 69dbde4d	---	Pass
Input: b7d9266d	---	Output: b7d9266d	---	Pass
Input: e27649ca	---	Output: e27649ca	---	Pass
Input: 2db67195	---	Output: 2db67195	---	Pass
Input: 1a469b04	---	Output: 1a469b04	---	Pass
Input: 61f70769	---	Output: 61f70769	---	Pass
Input: bab48d88	---	Output: bab48d88	---	Pass
Input: d928042d	---	Output: d928042d	---	Pass
Input: 30c79c2e	---	Output: 30c79c2e	---	Pass
Input: e408451c	---	Output: e408451c	---	Pass
Input: a6fd0729	---	Output: a6fd0729	---	Pass
Input: 901c6786	---	Output: 901c6786	---	Pass
Input: 08dac63d	---	Output: 08dac63d	---	Pass
Input: a766c470	---	Output: a766c470	---	Pass
Input: 047399ba	---	Output: 047399ba	---	Pass
Input: b35e79fa	---	Output: b35e79fa	---	Pass
Input: 05d5f61a	---	Output: 05d5f61a	---	Pass
Input: 00b96137	---	Output: 00b96137	---	Pass
Input: 379621c0	---	Output: 379621c0	---	Pass
Input: 9f261ab6	---	Output: 9f261ab6	---	Pass
Input: 837d0fdc	---	Output: 837d0fdc	---	Pass
Input: 5786ff78	---	Output: 5786ff78	---	Pass
Input: 9b7e80db	---	Output: 9b7e80db	---	Pass
Input: b6cf2b79	---	Output: b6cf2b79	---	Pass
Input: e4faee61	---	Output: e4faee61	---	Pass
Input: 091778a1	---	Output: 091778a1	---	Pass
Input: 91869650	---	Output: 91869650	---	Pass

Number of Comparisons:	100
Number of Successful Comparisons:	100
Number of Failed Comparisons:	0

3.4.1 Synchronizer

The synchronizer sub-block takes the inputs of serial clock, word select, and serial data and delays the signals and sync them with the master clock. This is done in order to ensure that no metastability occurs. The sub-block was also created to consolidate and shorten the amount of code needed to be written. Multiple blocks make use of the delayed and synced signals. It was appropriate to define these signals once and then to output them to the other blocks that need them. The inputs and outputs of the synchronizer sub-block can be seen in the following table.

Table 3.2: Interface Signals for Synchronizer Sub-Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rst_n	in	1	Reset
Synchronizer Input Interface			
_sck	in	1	Digital Audio Bit Clock
_ws	in	1	Word Select
_sd	in	1	Digital Audio Serial Data
Synchronizer Output Interface			
sck	out	1	Delayed and Synced Serial Clock
sck_transition	out	1	Serial Clock Level to Pulse Converter
ws	out	1	Delayed and Synced Word Select
sd	out	1	Delayed and Synced Serial Data

The _sck signal is the serial clock input signal and sck is the same signal synced and delayed by 2 master cycles. The sck_transition signal is a level to pulse converter of the serial clock. The signal is high anytime sck transitions from 0 to 1. The _ws signal is the word select input signal and ws is the same signal synced and delayed by 4 master clock cycles. The _sd signal is the serial data input signal and sd is the same signal synced and delayed by 4 master clock cycles.

The synchronizer was verified by checking if the _sck, _ws, and _sd signals were delayed by correct amount of clock cycles. Within the sample simulation that the sck signal was delayed by 2 clock cycles, and both the ws and sd signals were delayed by 4 clock cycles. In addition, the

level to pulse converter signal, sck_transition was verified as high each time sck transitioned from low to high.

Fig. 3.4: Sample of Synchronizer Simulation

3.4.2 Deserializer

The deserializer sub-block is responsible for taking the audio serial data and converting it to parallel data. The list of inputs and outputs of the deserializer sub-block can be seen in the following table.

Table 3.3: Interface Signals for Deserializer Sub-Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rst_n	in	1	Reset
Deserializer Input Interface			
sck_transition	in	1	Serial Clock Level to Pulse Converter
in_sd	in	1	Digital Audio Serial Data
in_ws	in	1	Word Select (Left/Right Audio Channel)
Deserializer Output Interface			
out_lft	out	16	Left Audio Channel
out_rgt	out	16	Right Audio Channel
out_xfc	out	1	Read Data Transfer Complete
Control/Status Register Fields			
rf_i2si_en	in	1	I2S input Enabled

The inputs sck_transition, in_sd, and in_ws are the outputs coming from the synchronizer sub-block. These are the level to pulse converter of the serial clock and the delayed and synced signals of the serial data and word select signals. The rf_i2si_en signal is an enable bit for the deserializer sub-block. The signals out_lft and out_rgt are the outputs of the left and right audio parallel data respectively. The signal out_xfc informs the FIFO that the transfer of a word has been completed.

The deserializer was verified by running a simulation that checked if the left and right output channels properly read in the input of the serial data signal. First it was verified that the serializer began properly outputting data after it becomes active. After properly becoming active the inputted data was verified to match up with the output data. It was also ensured that the data was properly stored into the proper channels depending on the value of the word select signal. Next the least significant bit was checked to see if it was stored after the first clock cycle after the word select signal transitioned. Lastly, the xfc signal was verified to pulse after the last bit of the right channel was stored.

Fig. 3.5: Sample of Deserializer Simulation

3.4.3 Audio Built-In Self-Test (BIST) Generator

The BIST generator sub-block is used to create a saw-tooth wave to test the I2S input interface. The saw-tooth wave is then outputted to the multiplexer. The following table lists the inputs and outputs of the BIST generator sub-block.

Table 3.4: Interface Signals for BIST Generator Sub-Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rst_n	in	1	Reset
sck_transition	in	1	Serial Clock Level to Pulse Converter
BIST Reg Interface			
rf_bist_start_val	in	12	Start Value
rf_bist_inc	in	8	Increment
rf_bist_up_limit	in	12	Upper Limit
BIST Output Interface			
i2si_bist_out_xfc	out	1	Transfer Complete
i2si_bist_out_data	out	32	Output Data

Signals rf_bist_start_val, rf_bist_inc, and rf_bist_up_limit are used to create a saw-tooth wave. The signals inform what value to start at, how much to increment by, and what the upper limit of the saw-tooth wave is respectively. The signals i2si_bist_out_data is the output data describing the saw-tooth wave, and i2si_bist_out_xfc is the output that represents that a saw-tooth wave word transfer was complete.

The BIST generator was verified by running a simple test that set the starting value to 0x800. The data value was then incremented on every 32nd pulse of the serial clock by a value of 0x700. The xfc output signal was verified by checking if it was high after the data incremented. The sub-block was further tested by ensuring that the output was reset to starting value after the output data reaches the upper limit of the BIST generator.

Fig. 3.6: Sample of BIST Generator Simulation

3.4.4 Multiplexer

The multiplexer sub-block is responsible for outputting either the BIST data and xfc output signals or the deserializer data and xfc output signals. The following table lists the multiplexer's input and output signals.

Table 3.5: Interface Signals for Multiplexer Sub-Module

Signal Name	Direction	Bits	Comment
Data Lines Input Interface			
in_0_data	in	32	Input 0 Data
in_0_xfc	in	1	Input 0 Transfer Complete
in_1_data	in	32	Input 1 Data
in_1_xfc	in	1	Input 1 Transfer Complete
Enabled Bit			
sel	in	1	Select Bit
Multiplexer Output Interface			
mux_data	out	32	Data Output
mux_xfc	out	1	Transfer Complete Output

The signals in_0_data, in_0_xfc, in_1_data, and in_1_xfc are the outputs of either the BIST generator or deserializer data and xfc signals. The sel signal is the select bit that either outputs the BIST data and xfc or deserializer data and xfc signals. The signals mux_data and mux_xfc represent the data and xfc signals being outputted to the FIFO sub-block.

The multiplexer was verified by running a simple simulation that inputted two sets of values for the pairs in_0 and in_1. The enabled bit was then toggled to see if the multiplexer outputted the correct pair of values.

Fig. 3.7: Sample of Multiplexer Simulation

3.4.5 First-In-First-Out (FIFO) Buffer

The FIFO sub-block organizes and manipulates a data buffer, where the first entry is the first output. In this particular block the FIFO is responsible for manipulating the audio data inputted and outputted in the I2S interfaces. The following table lists the inputs and outputs of the sub-block.

Table 3.6: Interface Signals for FIFO Sub-Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rst_n	in	1	Reset
Streaming Audio Interface with Multiplexer			
fifo_inp_data	in	32	Input Data
fifo_inp_rts	in	1	Write Client Asserts Ready to Send
fifo_inp_rtr	out	1	Output FIFO Asserts Ready to Receive
Streaming Audio Interface with Filter Block			
fifo_out_data	out	32	Output Data
fifo_out_rts	out	1	Output FIFO Asserts Ready to Send
fifo_out_rtr	in	1	Read Client Asserts Read to Receive

The fifo_inp_data is the parallel audio data being transferred into the FIFO block. The fifo_inp_rts and fifo_inp_rtr signals are the handshaked interface signals that inform whether data is ready to be sent and read respectively between the multiplexer and filter if in the I2S input interface or between the filter and FIFO if in the I2S output interface. The fifo_out_data signal is the parallel audio output data being sent to either the filter block if in the I2S input interface or serializer if in the I2S output interface. The fifo_out_rts and fifo_out_rtr signals are the handshaked interface that signals whether data is ready to be sent between the filter and FIFO if in the I2S input interface or read between the FIFO and filter if in the I2S output interface.

The FIFO sub-block was verified by queuing and de-queuing random values within a test bench. A quick simulation was run to check if the FIFO queued and de-queued the values defined within the test bench. If the FIFO was full the FIFO should have stopped queuing more data into the buffer, and if the FIFO was empty the FIFO should have outputted nothing from the buffer.

Fig. 3.8: Sample of FIFO Simulation

3.5 I2S Output Interface

The I2S output interface is primarily responsible for converting the received parallel audio data from the filter block to serial data as an output. The I2S output interface is made of two sub-blocks, the serializer and FIFO. The following diagram shows the sub-blocks that make the I2S_out block and displays the input, output, and interconnecting signals.

Fig. 3.9: Block Diagram of I2S Output Interface

The following table displays all the input and output signals the I2S output interface interacts with. The table provides the name, direction, bit size, and a short description of each signal.

Table 3.7: Interface Signals for I2S Output

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rst_n	in	1	Reset
Streaming Audio Interface with Filter Block			
sck_transition	in	1	Serial Clock Pulse Converter
filt_data	in	32	Parallel Digital Audio
filt_rtr	out	1	Ready to Receive
filt_rts	in	1	Ready to Send
I2S Output Interface			
i2so_sck	out	1	Digital Audio Bit Clock
i2so_ws	out	1	Word Select (Left/Right Audio Channel)
i2so_sd	out	1	Digital Audio Serial Data
Control/Status Register Fields			
trig_fifo_underrun_clr	in	1	Reset FIFO Underrun
ro_fifo_underrun	out	1	Output Audio FIFO Underrun

The sck_transition signal is the level to pulse converter signal that will be provided by the I2S input interface. The signal filt_data will be provided by the filter block that contains the 32-bit parallel audio data. The signals filt_rtr and filt_rts are the handshaked interface signals that indicate whether the filter is ready to send data and the I2S output interface is ready to receive data. The i2so_sck is the serial clock that is outputted in the I2S Input interface that will be transmitted to the I2S output interface. The i2so_ws is an output signal that indicates whether the data being transmitted belongs either to the left or right channel. The i2so_sd signal is the serial data that is converted from the parallel audio input data. The ro_fifo_underrun signal indicates whether the

data being fed to the I2S output interface is slower than the rate it can process the data, and the trig_i2so_fifo_underrun_clr signal clears the ro_fifo_underrun bit.

The I2S output interface was verified by inputting a list of 32 bit words and confirming that they were being serialized upon exiting the block. To make verification easier, the test bench rebuilt the serial data back into a parallel form based on the word select signal. It was confirmed that the input matched the output within the text files, thus the serializer was working.

Fig. 3.10: Sample of I2S Output Interface Simulation

Sample of I2S Output Interface Text Files

Input: 12153524	---	Output: 12153524	---	Pass
Input: c0895e81	---	Output: c0895e81	---	Pass
Input: 8484d609	---	Output: 8484d609	---	Pass
Input: b1f05663	---	Output: b1f05663	---	Pass
Input: 06b97b0d	---	Output: 06b97b0d	---	Pass
Input: 46df998d	---	Output: 46df998d	---	Pass
Input: b2c28465	---	Output: b2c28465	---	Pass
Input: 89375212	---	Output: 89375212	---	Pass
Input: 00f3e301	---	Output: 00f3e301	---	Pass
Input: 06d7cd0d	---	Output: 06d7cd0d	---	Pass
Input: 3b23f176	---	Output: 3b23f176	---	Pass
Input: 1e8dcd3d	---	Output: 1e8dcd3d	---	Pass
Input: 76d457ed	---	Output: 76d457ed	---	Pass
Input: 462df78c	---	Output: 462df78c	---	Pass
Input: 7cfde9f9	---	Output: 7cfde9f9	---	Pass
Input: e33724c6	---	Output: e33724c6	---	Pass
Input: e2f784c5	---	Output: e2f784c5	---	Pass
Input: d513d2aa	---	Output: d513d2aa	---	Pass
Input: 72aff7e5	---	Output: 72aff7e5	---	Pass
Input: bbd27277	---	Output: bbd27277	---	Pass
Input: 8932d612	---	Output: 8932d612	---	Pass
Input: 47ecdb8f	---	Output: 47ecdb8f	---	Pass
Input: 793069f2	---	Output: 793069f2	---	Pass
Input: e77696ce	---	Output: e77696ce	---	Pass
Input: f4007ae8	---	Output: f4007ae8	---	Pass
Input: e2ca4ec5	---	Output: e2ca4ec5	---	Pass
Input: 2e58495c	---	Output: 2e58495c	---	Pass
Input: de8e28bd	---	Output: de8e28bd	---	Pass
Input: 96ab582d	---	Output: 96ab582d	---	Pass

Input: b2a72665	---	Output: b2a72665	---	Pass
Input: b1ef6263	---	Output: b1ef6263	---	Pass
Input: 0573870a	---	Output: 0573870a	---	Pass
Input: c03b2280	---	Output: c03b2280	---	Pass
Input: 10642120	---	Output: 10642120	---	Pass
Input: 557845aa	---	Output: 557845aa	---	Pass
Input: cecccc9d	---	Output: cecccc9d	---	Pass
Input: cb203e96	---	Output: cb203e96	---	Pass
Input: 8983b813	---	Output: 8983b813	---	Pass
Input: 86bc380d	---	Output: 86bc380d	---	Pass
Input: a9a7d653	---	Output: a9a7d653	---	Pass
Input: 359fdd6b	---	Output: 359fdd6b	---	Pass
Input: eaa62ad5	---	Output: eaa62ad5	---	Pass
Input: 81174a02	---	Output: 81174a02	---	Pass
Input: d7563eae	---	Output: d7563eae	---	Pass
Input: 0effe91d	---	Output: 0effe91d	---	Pass
Input: e7c572cf	---	Output: e7c572cf	---	Pass
Input: 11844923	---	Output: 11844923	---	Pass
Input: 0509650a	---	Output: 0509650a	---	Pass
Input: e5730aca	---	Output: e5730aca	---	Pass
Input: 9e314c3c	---	Output: 9e314c3c	---	Pass
Input: 7968bdf2	---	Output: 7968bdf2	---	Pass
Input: 452e618a	---	Output: 452e618a	---	Pass
Input: 20c4b341	---	Output: 20c4b341	---	Pass
Input: ec4b34d8	---	Output: ec4b34d8	---	Pass
Input: 3c20f378	---	Output: 3c20f378	---	Pass
Input: c48a1289	---	Output: c48a1289	---	Pass
Input: 75c50deb	---	Output: 75c50deb	---	Pass
Input: 5b0265b6	---	Output: 5b0265b6	---	Pass
Input: 634bf9c6	---	Output: 634bf9c6	---	Pass
Input: 571513ae	---	Output: 571513ae	---	Pass
Input: de7502bc	---	Output: de7502bc	---	Pass
Input: 150fdd2a	---	Output: 150fdd2a	---	Pass
Input: 85d79a0b	---	Output: 85d79a0b	---	Pass
Input: b897be71	---	Output: b897be71	---	Pass
Input: 42f24185	---	Output: 42f24185	---	Pass
Input: 27f2554f	---	Output: 27f2554f	---	Pass
Input: 9dcc603b	---	Output: 9dcc603b	---	Pass
Input: 1d06333a	---	Output: 1d06333a	---	Pass
Input: bf23327e	---	Output: bf23327e	---	Pass
Input: 0aaa4b15	---	Output: 0aaa4b15	---	Pass
Input: 78d99bf1	---	Output: 78d99bf1	---	Pass
Input: 6c9c4bd9	---	Output: 6c9c4bd9	---	Pass
Input: 31230762	---	Output: 31230762	---	Pass
Input: 2635fb4c	---	Output: 2635fb4c	---	Pass
Input: 4fa1559f	---	Output: 4fa1559f	---	Pass
Input: 47b9a18f	---	Output: 47b9a18f	---	Pass
Input: 7c6da9f8	---	Output: 7c6da9f8	---	Pass
Input: dbcd60b7	---	Output: dbcd60b7	---	Pass

Input: cfc4569f	---	Output: cfc4569f	---	Pass
Input: ae7d945c	---	Output: ae7d945c	---	Pass
Input: adcbc05b	---	Output: adcbc05b	---	Pass
Input: 44de3789	---	Output: 44de3789	---	Pass
Input: a4ae3249	---	Output: a4ae3249	---	Pass
Input: e8233ed0	---	Output: e8233ed0	---	Pass
Input: ebfec0d7	---	Output: ebfec0d7	---	Pass
Input: a8c7fc51	---	Output: a8c7fc51	---	Pass
Input: 4b212f96	---	Output: 4b212f96	---	Pass
Input: 061d7f0c	---	Output: 061d7f0c	---	Pass
Input: e12ccec2	---	Output: e12ccec2	---	Pass
Input: 6457edc8	---	Output: 6457edc8	---	Pass
Input: bb825a77	---	Output: bb825a77	---	Pass
Input: 1ef2ed3d	---	Output: 1ef2ed3d	---	Pass
Input: 090cdb12	---	Output: 090cdb12	---	Pass
Input: bf05007e	---	Output: bf05007e	---	Pass
Input: 36e5816d	---	Output: 36e5816d	---	Pass
Input: 1cd9e739	---	Output: 1cd9e739	---	Pass
Input: 0fd28f1f	---	Output: 0fd28f1f	---	Pass
Input: e9ebf6d3	---	Output: e9ebf6d3	---	Pass
Input: 42d92f85	---	Output: 42d92f85	---	Pass
Input: bc148878	---	Output: bc148878	---	Pass

Number of Comparisons:	100
Number of Successful Comparisons:	100
Number of Failed Comparisons:	0

3.5.1 Serializer

The serializer sub-block is where the parallel audio data is converted into serial data. The following table shows the inputs and outputs of the serializer sub-block.

Table 3.8: Interface Signals for Serializer Sub-Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rst_n	in	1	Reset
Serializer Input Interface			
sck_transition	in	1	Serial Clock Level to Pulse Converter
filt_i2so_lft	in	16	Left Audio Channel
filt_i2so_rgt	in	16	Right Audio Channel
filt_i2so_rts	in	1	Ready to send
filt_i2so_rtr	out	1	Ready to read
Serializer Output Interface			
i2so_sd	out	1	Digital Audio Serial Data
i2so_ws	out	1	Word Select

As previously stated the sck_transition signal is the output created from the synchronizer sub-block. The signals filt_i2so_lft and filt_i2so_rgt are the parallel audio data channels that were outputted by the FIFO. The filt_i2so_rts and filt_i2so_rtr are the handshaked interface signals that indicate whether the FIFO is ready to send and the serializer is ready to read data. However, it is important to note in this case that the serializer only cares when the rtr signal becomes high on its first instance. The serializer will continue to be active regardless if the rtr signal becomes 0 again. The i2so_sd and i2so_ws as mentioned before are the output of the I2S output interface of the serial data and word select signals.

The serializer sub-block was verified by running a test bench that fed in data to the left and right channels. To ensure that the block was working properly it needed to become active after the first rtr signal that goes from low to high. After becoming active the serial data output should correspond to the input data coming in from the left and right channel depending on the value of word select. These conditions were all verified within the test bench simulation.

Fig. 3.11: Sample of Serializer Simulation

Chapter 4: Digital Filtering – D. Naik

4.1 Introduction

The main function of the ASIC chip is as a digital filter for audio processing. A digital filter operates on an input signal and produces a new signal, or filtered signal. In DSP, two of the most basic uses of a digital filter are signal separation and signal restoration. Separation is done when an input signal is polluted with interference, noise, or other signals. Filtering, removes any unwanted features from an input signal. Restorations is used when the signal has been distorted. One example of restorations is when an image is focused after being acquired due to a shaky lens. These filters are implemented in many types of devices and range in complexity. More advance filters are used for enhancements, compensation for listening room dynamics, special effects, and many other features. For example, MP3 players have a multitude of digital filters structures for different applications within the system. The four most basic types of filters are high pass, low pass, band pass, and band reject. Below in Figure 3.1, shows the frequency response of each filter structure. A digital filter can be represented with its frequency response, which is the discrete Fourier transform of its impulse response.

Fig. 4.1: Frequency Response of Filter Structures

Above, each filter structure has 3 interesting regions. The passband is the area in which frequency is allowed to pass freely. The stopband is the area in which the frequency is attenuated. Finally, the transition band is the transition between the passband and the stopband. Digital filters can be implemented two ways: by convolution and by recursion. Convolution is also called finite impulse response or FIR and recursion is called infinite impulse response or IIR. In FIR filters the filter coefficients are derived from the impulse response of the system. The given filter coefficients characterize the type of the digital filter. Convolution is defined by the equation:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

Here x is the discrete time input signal and h is the filter coefficients/impulse response. For the TCNJ ASIC, the filter will have 512 taps or 512 delays. Thus the filter will need to store, present and past inputs and index them appropriately.

In order to design a digital filter for audio processing, the human auditory system needs to be taken into consideration. The perception of continuous sound, can be separated into three parts: loudness, pitch, and timbre. Our audio quality goal will be to match the sound quality from a high fidelity compact disc. The specifications for the sound quality are the following. The chip will support sampling rates up to 48 kHz with precision of 16 bits per audio channel (left and right). The bit depth refers to the number of bits of precision when quantizing the sampled signal. The number of bits reduces the quantization error. Thus reducing the "hiss" associated with lower bit precision.

4.2 Filter

The filter block is a sub-block of the top level chip. Figure 3.2 shows the filter with respect to the other interfaces.

Fig. 4.2: Chip Diagram with Filter Highlighted

The filter interfaces with the I2S input and output blocks, and the Register block. The incoming audio signal is streamed in from the I2S input as serial data. Both channels (right and left) are sent as the same data line. In addition to providing control bits for the shift and clip, the register is responsible for providing the filter with filter coefficients. The input signal, including past input samples are convolved with the filter coefficients. Before the filter output is sent to the I2S output, three operations are performed: rounding, shifting, and clipping. Figure 3.2 is the top level view of the filter. Table 3.1 describes the corresponding signals for the top level filter.

Fig. 4.3: Top Level Block Diagram of Filter

Table 4.1: Interface Signals for Filter

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rstb	in	1	Asynchronous Reset
Audio In			
filter_aud_in	in	32	Input Digital Audio
filter_aud_in_rts	in	1	Input FIFO Asserts Ready to Send
filter_aud_in_rtr	out	1	Filter STM Asserts Ready to Receive
Audio Out			
filter_aud_out	out	32	Output Digital Audio
filter_aud_out_rts	out	1	Filter STM Asserts Ready to Send
filter_aud_out_rtr	in	1	Output FIFO Asserts Read to Receive
Register Fields			
rf_filter_shift	in	3	number of bit positions to shift
rf_filter_clip_en	in	1	1-perform clipping 0-no clipping
rf_filter_coeff [0:511]	in	8192	Filter coefficient

Referring to Table 3.3, *rstb*, represents an asynchronous reset. An asynchronous reset is typically active low, meaning that it triggers on the negedge as opposed to a synchronous reset, which is often active high. The advantage of using an asynchronous reset is, the reset has priority over any signal, including the master clock. This allows the block to reset with or without the presence of a clock. The second signals to consider are *filter_aud_in* or *filter_aud_out*. These digital audio signals contain bits for both audio channels. The right audio channel bits are stored in the first 16 LSB [15:0] and the left audio channel bits are stored in the first 16 MSB [31:16]. The final signal to consider further is *rf_filter_coeff*. The filter coefficients are assumed to be stable. If they change during any calculation of an output sample the result will be unpredictable. Figure 3.3 is a more detailed block diagram of the filter. Two things of note. First the multiplication will be signed multiplication. This is done by defining wires and registers as signed type. Secondly, there will be two instantiations of the accumulator for left and right audio signals. *Filter_aud_in* will be split into two before the multiplication and concatenated before the three operations.

Fig. 4.4: Detailed Block Diagram of Filter

The following subsections, will highlight specific submodules of the filter and describe them in detail.

4.2.1 Filter State Machine

The filter is controlled by a centralized finite state machine (FSM). A FSM can only function in a single state, or the current state. It is defined by its different states and trigger conditions (transitions). The states of a FSM can be encoded in various ways – one convient encoding is known as One-Hot. In a One-Hot encoding each state is represented by a single bit.

For the TCNJ ASIC, the filter has 4 different states, so for this instance 4 flip-flops will be reserved to represent the filter FSM. Table 3.2 shows the highlights and describes the signals for the FSM.

Table 4.2: Interface Signals for Finite State Machine

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rstb	in	1	Asynchronous Reset
Audio Signals			
filter_aud_in	in	32	Input Digital Audio
rf_filter_coeff	in	16	Filter Coefficient
filter_aud_out	out	32	Output Digital Audio

In addition to the signals described above, the submodule uses additional internal wires and signals. These specific signals can be found in the Source Code section in the Appendix.

IDLE:

The FSM enters the IDLE state if, and only if, the RESET signal is asserted. Upon assertion of XFC the FSM will transition to the TRANSFER state.

TRANSFER:

Once XFC is asserted, the filter will transition into the TRANSFER state. In this state, *filter_aud_in* is written into the storage module and the write pointer is incremented. Once the transfer is complete, read enable for the storage module and *filter_aud_in_rtr* are deasserted. The FSM will transition to MULTIPLY_1ST after this is complete.

MULTIPLY_1ST:

The convolution phase is separated into two states because the accumulator needs to load the first product. After the first computation is completed the state will transition to MULTIPLY.

MULTIPLY:

In the MULTIPLY_1ST and MULTIPLY states the FSM access the storage module for computation. The TCNJ ASIC filter is a 512 tap filter. The filter needs to access previous inputs, as far as 511 inputs from the past. Further discussion of the storage module is found in the next subsection. The 512 most recent input samples are multiplied against a corresponding filter coefficient. The filter coefficients are accessed from 0 to 511 for each iteration of the states. The inputs are accessed from the newest input to the input from the last iteration. Below are a few iterations of the convolution process.

1) $y[0] = h[0]\,x[0] + h[1]\,x[-1] + \ldots + h[510]\,x[-510] + h[511]\,x[-511]$

2) y[1] = h[0] x[1] + h[1] x[0] + … + h[510] x[-509] + h[511] x[-510]
3) y[2] = h[0] x[2] + h[1] x[1] + … + h[510] x[-508] + h[511] x[-509]

Each of these products are computed each clock cycle. Each of these products are sent to an accumulator, which stores the sum of products until the next iteration. The accumulator will store a 40-bit signal, so there is little chance of an overflow. This 40-bit signal is rounded, clipped, and shifted and is ready to be sent to the Output I2S. Once the state is ready for a new input, read enable for both storage modules is deasserted and *filter_aud_in_rtr* is asserted. The filter will go through each of the discussed states until the chip is reset and the filter will transition into the IDLE state.

4.2.2 Filter Storage Module

The storage module is responsible for storing, indexing, writing, and reading the input signal. Table 3.3 shows the highlights and describes the signals for the Filter Storage Module. These signals are representations for both instances of the submodule.

Table 4.3: Interface Signals for Filter Storage Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
Write Data			
wren	in	1	1-wrdata will be written into wrptr
wrptr	in	9	Current Location of Write Pointer
wrdata	out	32	Data to be Written
Read Data			
rden	out	1	1-rddata will be read from rdptr
rdptr	in	9	Current Location of Read Pointer
rddata	out	32	Data Being Read

In addition to the signals stated above, the storage module defines a 2D array. The width of the array is 32 bits and the depth is 512 locations. The array behaves in a circular manner. Once the *wrptr* or *rdptr* indexes the last position it will wrap back to the beginning of the array. Consequently, the data will be overwritten. The signals *wren*, *rden*, *wrptr*, and *rdptr* are controlled by the state machine.

4.2.3 Mux

The storage module is sufficient for indexing and reading filter coefficients but does not satisfy our requirements. Using the storage module will be inefficient in reading the filter coefficients because the data is already stored in Register block and would be redundant to write them in an array located in the Filter block. Instead, a mux can be used to read the correct filter coefficients using a bus of wires.

In the Register block, a single filter coefficient is represented by two signals: rf_filter_coeff*n*_a and rf_filter_coeff*n*_b. Instead of reading two signals for a single filter coefficient, the mux concatenates the two wires into a single wire. Signal rf_filter_coeff*n*_a is stored in the first 8 LSB [7:0] and rf_filter_coeff*n*_b is stored in the first 8 MSB [15:8]. Below in Table 3.4, shows the signals for the mux submodule. A clock is placed to keep it in sync with *filter_aud_in* values.

Table 4.4: Interface Signals for Mux Module

Signal Name	Direction	Bits	Comment
General			
clk	in	32	Signal being performed on
Select Data			
rden	in	1	Read enable
rdptr	in	9	Location of data to be selected
rf_filter_coef[0:511]	in	8196	Filter Coefficient
rddata	out	15	Selected Data

4.2.3 Accumulator

An accumulator, at the posedge clock, takes the input signal and adds it to the current value that is stored in a D flip-flop. The resulting value is stored for future computations until the accumulator is reset. To avoid overflow, the value being stored will be 40 bits long. The signals are defined in the below table. When *enable* is asserted, the accumulator will function as normal. When *load* is enabled, *D* is set to the new value.

Table 4.5: Interface Signals for Accumulator Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rstb	in	1	Asynchronous Reset
Control Signals			
enable	in	1	1-Accumulate 0-Idle
load	in	1	Load new value
Input/Output Data			
D	in	32	Input Value
Q	out	40	Current Total

4.2.3 Shift, Round, and Truncate

This module is to receive the final accumulated value and perform shifting, rounding, and truncating. The final value is a 16-bit value representing digital audio data. Below is a diagram of the arithmetic.

Fig 4.5: Finite State Machine: States and Transitions

Table 4.6: Interface Signals for Shift Round Truncate Module

Signal Name	Direction	Bits	Comment
General			
clk	in	1	Master Clock
rstb	in	1	Asynchronous Reset
Control Fields			
rf_shift	in	3	Shift Amount
rf_clip	in	1	Allow clipping when overflow
trig_filter_ovf_flag	in	1	Reset overflow flag
ro_filter_ovf_flag	out	1	Triggered when overflow occurs
Audio Data			
acc_in	in	40	Final accumulator input
filter_out	out	16	Output Digital Audio

4.3 Verification

To verify the Filter Block, I performed two types of tests, individual unit testing for each submodule and full block level test. The full block level test was to verify the functionality of the block. I did testing of actual filter responses. The following sections will outline testing from the lowest module to the highest module. As far as functionality, the filter block passes all the current tests. The filter was also verified with the entire chip in the Verification Section. This was done by using filter coefficients produced by the Matlab FDA toolbox. Below are is an example impulse response of 512-tap Low-Pass filter. Using coefficients produced by Matlab, I compared results from the Verilog simulation with a Java Model.

Fig. 4.6: Impulse Response of 512-Tap Low-Pass Filter

4.3.1 Accumulator

To verify the Accumulator, I tested three functions: loading, normal operation, and reset. The first test was to verify when load and enable were both asserted. What I expected was that the output would not accumulate and would equal to the input signal. The second test was to verify when load = 1'b0 and enable = 1'b1. I expected to see the accumulator increment by the input signal each clock cycle. The output signal has a 4 bit overhead to avoid overflow. The final test was to check what occurred when rstb = 1'b0. I expected the reset to take priority over the other signals and set the output to 1'b0. Below is the simulation output. All three of my assertions were proven correct.

Fig. 4.7: Simulation of Accumulator

4.3.2 Filter Storage Module

To verify the Filter Storage Module, I first tested basic writing and reading. I set the *wren = 1'b1*. I expect this to allow the module to write into the array at the given pointer. Starting with *wrdata = 32'hAAAAAAAA*, then *wrdata = 32'hBBBBBBBB*, and finally *wrdata = 32'hCCCCCCCC*. I increment the *wrptr* by 1'b1 after a 100 ns delay. To verify the values were

stored correctly, I set *wren = 1'b0* and *rden = 1'b1*. I read the array from ram[0] to ram[2], by incrementing *rdptr* by 1'b1 after a 100ns delay. To verify that the data is not lost when reading it, once I reached ram[2], I decremented by 1'b1 after 100ns delay. I expected to see *rddata = 32'hAAAAAAAA*, *rddata = 32'hBBBBBBBB*, *rddata = 32'hCCCCCCCC*. *rddata = 32'hCCCCCCCC*, *rddata = 32'hBBBBBBBB*, *rddata = 32'hAAAAAAAA*. Below is the simulation output. My assertions were proven correct.

Fig. 4.8: Simulation of Filter Storage Module Test 1

The second, test I performed was to check how storage module performed when the write and read pointer went above 511. I expect the storage module to perform circularly. To perform this test, I incremented the *wrptr* by 513. I expected the pointer to go back to *wrptr = 1'd1*. The write data was *wrdata = 32'hBBBBBBBB*. I then read ram[1]. I expected to have an output of *rddata = 32'hBBBBBBBB*. Below is the simulation output.

Fig. 4.9: Simulation of Filter Storage Module Test 2

4.3.3 Mux

To verify the mux, I tested if I received the correct 16 bit filter_coefficients. In addition, I tested if the mux correctly concatenated *rf_filter_coeffn_a* and *rf_filter_coeffn_b*. The 'a' component is stored from [7:0] and the 'b' component is stored from [15:8]. I set *rf*_filter_coeff(0-8)*_a* to 8'h00 to 8'h08, respectively. For *rf_filter_coeff(0-8)_b*, I set it to 8'h08 to 8'h00, respectively. For example, *rf_filter_coeff0* should output *16'h0800*. Another example would be, rf*_filter_coeff3* should output *16'h0503*. My test was successful as seen in the simulation output below.

Fig. 4.10: Simulation of Mux

4.3.4 Filter STM/Filter

I tested by filter state machine by testing my top level. For the first test I observed if the states were transitioning properly. For my design, the filter will kick-off when XFC is complete. After a delay of 100ns, I asserted *aud_in_rts*. Below is my simulation. Note: for testing I am using an 8 tap filter. In the simulation once the filter kicks-off, it will cycle through the states until it is reset around 1000ns. The *do_transfer, do_multiply_1st, and do_multiply* represent when the states are running. For my STM implementation, it takes a single clock cycle to transition.

Fig. 4.11: Simulation of Filter Test 1

The next test I performed was to vary the *aud_in_rts* signal from deasserted to asserted. It is possible for the I2S block to be not ready to send data when the filter block is ready to take data. To test this, I asserted aud_in_rts and deasserted it after 100ns. I then waited 400ns to asserted it again. The MULTIPLY states will not be affected by this, but the TRANSFER state will get delayed until *aud_in_rts* is asserted again.

Fig. 4.12: Simulation of Filter Test 2

Above the simulation, shows that the TRANSFER state is prolonged as it is waiting for XFC to be complete before it can transfer aud_in and change states. This can be proven by the simulation below. Here when *aud_in_rtr* and *aud_in_rts* are both deasserted the *aud_in data* will not be stored in the ram array. Around 700ns, *aud_in_rts* is asserted and *aud_in* is transferred into the current *wrptr*. After the transfer the state changes to the MULTIPLY_1ST stage.

Fig. 4.13: Simulation of Filter Test 3

The next test I performed was to observe the behavior of the read pointer for both the mux and storage module. I am expecting the read pointer for the storage module to decrement and increment for the mux. Each cycle should have 8 (or the number of Taps) increments or decrements.

Fig. 4.14: Simulation of Filter Test 4

Note that the last clock cycle of the multiply state is superfluous for the current cycle. Also for the read pointer for the storage module (*rd_addr_x*), it starts at the current location of the writer pointer. This proves that the filter follows, which is the convolution process.

1) y[0] = h[0] x[0] + h[1] x[-1] + … + h[510] x[-510] + h[511] x[-511]
2) y[1] = h[0] x[1] + h[1] x[0] + … + h[510] x[-509] + h[511] x[-510]
3) y[2] = h[0] x[2] + h[1] x[1] + … + h[510] x[-508] + h[511] x[-509]

The next test, was functionality of the filter. Here are the following testing parameters:

rf_coeff0_b | rf_coeff0_a = 16'h0001
rf_coeff1_b | rf_coeff1_a = 16'h0002
rf_coeff2_b | rf_coeff2_a = 16'h0003
rf_coeff3_b | rf_coeff3_a = 16'h0004
rf_coeff4_b | rf_coeff4_a = 16'h0005
rf_coeff5_b | rf_coeff5_a = 16'h0006
rf_coeff6_b | rf_coeff6_a = 16'h0007
rf_coeff7_b | rf_coeff7_a = 16'h0008

The audio values vary from 32’h00000000 to 32’h00FF00FF, again the 16 most MSB are the left audio and 16 most LSB are the right. We are assuming that the I2S is always ready to send for this test. We tested I2S stalls in filter test 2. The filter output is a 32-bit value, representing both right and left audio channels.
Below is the simulation result.

Fig. 4.15: Simulation of Filter Test 5

To check our simulation, I am going to check the results by hand. Note: all values in Hex

Here is the input signal and impulse response:
x[n] = {00af003a, 00aa00aa, 00aa003a, 00dd00aa, 001a00a2, 001a00a2, 001100da, 00aa00ff}
xleft[n] = {00af, 00aa, 00aa, 00dd, 001a, 001a, 0011, 00aa}
xright[n] = {003a, 00aa, 003a, 00aa, 00a2, 00a2, 00da, 00ff}
h[n] = {0001, 0002, 0003, 0004, 0005, 0006, 0007, 0008}
The sum of products is defined by:
Note: I am using hex values for the index values
y[0] = x[0]h[0] + x[7]h[1] + x[6]h[2] + x[5]h[3] + x[4]h[4] + x[3]h[5] + x[2]h[6] + x[1]h[7]
There are two multipliers and accumulators for left and right audio. The calculations, done with WolframAlpha.com , are the following:
Left:
*yLeft = 00af*0001 + 00aa*0002 +0011*0003 + 001a*0004 +001a*0005 + 00dd*0006 +00aa*0007 + 00aa*0008*
yLeft = 0x01244
Right:
*yRight = 003a*0001 + 00ff*0002 +00da*0003 + 00a2*0004 +00a2*0005 + 00aa*0006 +003a*0007 + 00aa*0008*

yRight = 0x0155a
If I concatenate yLeft and yRight, I get *y = 0x012440155a*. Then we clip the least significant bit to get a 32-bit value of *0x01244015*.
This matches what we get in our simulation.

Chapter 5: Register Block – J. Swift

5.1 Introduction

The register block provides storage and access for control and status information. In Fig 5.1 the interaction of register block and the other blocks can be observed. The I2C slave interface sends byte-wide control data to the register block. Many of the registers represent filter coefficients and act as the filter's parameters. These parameters can be changed by the user and written to the register block to control the audio filter behavior. Each filter coefficient is represented by two 8 bit values that are concatenated when sent to the filter control block. The I2C must be able to specify an address and write data to the specified address for the register to hold. When reading data, the I2C block will be able to read the stored data of a specified address. When an underrun occurs in the I2S's FIFO, the register block will be able to clear the sticky bit that is flagged when the filter is producing data at a lower rate than the I2S Output is consuming.

Fig. 5.1: Chip Diagram with Register Highlighted

Internally, the register block is broken down into a data demultiplexer that takes the write enable, address, and write data, and writes to the appropriate register bits based on the address. With the rising edge of the write enable, the data demultiplexer allows data to be stored in the register. This low level logic is controlled by the rising edge of the clock, which can be seen in Fig 5.2.

Fig. 5.2: Internal Micro-Architecture of Register

5.2 Interfaces

The top level interface of the register block is described in Fig 5.3 which shows the signal's flow of direction and number of bits in each signal. The addr input signal is an 11-bit variable that holds the value of the address in the register block in which data can be read from or written to.

Fig. 5.3: Top Level Interface Design of reg.v

The wdata is an input variable array that writes 8-bits of data to a specified 11-bit address. Similarly, the rdata output array returns the 8-bits of data demanded by any signal request of a

specified address. The write_xfc is an input 1-bit transfer signal that indicates when the transfer of the data written is complete. The read_xfc is an output 1-bit signal that similarly goes high when read data transfer is complete. Master clock, clk, and reset, rst, are 1 bit signals that regulate activity to occur on the positive edge of the clock or the negative edge of the rst signal. Lastly, the i2s_inoverrun will turn into a sticky bit that is manipulated when I2S FIFO has an overrun or underrun condition.

5.3 Register Mapping

The register map seen in Table 4.1 includes a list of all the signals assigned to an address and shows the breakdown of bits in each address. The natural 32-bit word aggregation is indicated in bold. Each signal is prefixed with “ro” or “rf” in the beginning of the field name. When a signal is prefixed with “ro” it means the signal represents a “Read Only” field, and is an input signal to the register block. The “rf” prefix indicates that the signal is a “Read/Write” register bit, and is an output signal. The CONTROL signals are either on, meaning they are set to the value 1, or off when set to a value of 0. The I2S_CLOCK_CONTROL are addresses that control the clock for the I2S module. The STATUS addresses are 1 bit signals used when the I2S FIFO exhibits an overrun or underrun in which the signals will then be set to a value of 1 to alert the FIFO malfunction. BIST addresses will be dedicated to I2S’s predefined sawtooth wave. The read/write BIST signals are responsible for the starting value of the sawtooth wave, incrementing the sawtooth wave, and determining the upper limit of the sawtooth wave. The I2C_REG_INDIR_ADDR is the address register used for indirect addressing via I2C. Lastly, the 512 filter coefficients are broken down into two 8 bit coefficients with parts a and b. Each subdivided coefficient is assigned to an address in which data can be written and read from.

Table 5.1: Register Mapping of Each Bit in the Address it is Stored

Address	Register Name	Field Name	Bits	Description	RO/W O/RW	Default Value
0x004	**CONTROL**					
0x004		rf_i2si_bist_en	0:0	0- audio source is i2si. 1- audio source is BIST	RW	0x1
0x004		rf_filter_shift	3:1	number of bit positions to shift after filter accumulator	RW	0xF
0x004		rf_filter_clip_en	4:4	1- performs clipping 0- no clipping	RW	0x1

Address	Register Name	Field Name	Bits	Description	RO/ WO/ RW	Default Value
0x004		rf_i2si_en	5:5	Enable bit for Deserializer. 0 = inactive, 1 = active	RW	0x01
0x008	**STATUS**					
0x008		trig_fifo_overrun	0:0	i2s input audio fifo overrun clear	WO	NA
0x008		ro_fifo_overrun	1:1	i2s input audio fifo overrun	RO	0
0x008		trig_fifo_unerrun	2:2	i2s output fifo underrun clear	WO	NA
0x008		ro_fifo_underrun	3:3	i2s output audio fifo underrun	RO	0
0x008		trig_filter_ovf_flag_clear	4:4	filter overflow flag clear	WO	NA
0x008		ro_filter_ovf_flag	5:5	filter overflow flag	RO	0
0x00C	**BIST**					
0x00C		rf_i2si_bist_incr	7:0	increment of sawtooth wave	RW	0x010
0x00D		rf_i2si_bist_start_val_a	7:0	start value of sawtooth wave	RW	0x80
0x00E		rf_i2si_bist_start_val_b	3:0		RW	0x0
0x010	**BIST**					
0x010		rf_i2si_bist_upper_limit_a	7:0	upper limit of the sawtooth wave	RW	0x7F
0x011		rf_i2si_bist_upper_limit_b	3:0		RW	0xF
0x0400	**FILT_COEFFS_0_1**					
0x0400		rf_filter_coeff0_a	7:0	Filter Coefficient 0	RW	0x0
0x0401		rf_filter_coeff0_b	7:0	Filter Coefficient 0	RW	0x0

Address	Register Name	Field Name	Bits	Description	RO/WO/RW	Default Value
0x0402		rf_filter_coeff1_a	7:0	Filter Coefficient 1	RW	0x0
0x0403		rf_filter_coeff1_b	7:0	Filter Coefficient 1	RW	0x0
0x0404	**FILT_COEFFS_2_3**					
0x0404		rf_filter_coeff2_a	7:0	Filter Coefficient 2	RW	0x0
0x0405		rf_filter_coeff2_b	7:0	Filter Coefficient 2	RW	0x0
0x0406		rf_filter_coeff3_a	7:0	Filter Coefficient 3	RW	0x0
0x0407		rf_filter_coeff3_b	7:0	Filter Coefficient 3	RW	0x0
0x7FC	**FILT_COEFFS_510_511**					
0x07FC		rf_filter_coeff510_a	7:0	Filter Coefficient 510	RW	0x0
0x07FD		rf_filter_coeff510_b	7:0	Filter Coefficient 510	RW	0x0
0x07FE		rf_filter_coeff511_a	7:0	Filter Coefficient 511	RW	0x0
0x07FF		rf_filter_coeff511_b	7:0	Filter Coefficient 511	RW	0x0

5.4 Sub-Blocks

5.4.1 trig_generator.v

The trig_generator.v is responsible for generating a signal to clear the overrun and underrun status bits. The two output bits, trig_i2si_fifo_overrun_clr and trig_i2so_fifo_underrun_clr, are initialized to zero when not rst. If the address is the 11-bit hexadecimal number 0x00C and the xfc_write is set to one, then the if-statement returns true and the trigger bits are edited. Trig_i2si_fifo_overrun_clr and trig_i2so_fifo_underrun_clr are first initialized to zero in order to allocate them later only in the case of an overrun or underrun. When the data transfer is complete, meaning the XFC is 1, and the address is 0x00C, two if-statements are then checked. One stating that when data written to 0x00C is 0, the trig_i2si_fifo_overrun_clr bit is set to 1, and the other stating that when data written to 0x00C is 2, the trig_i2so_fifo_underrun_clr bit is set to 1. These 2-bits in the address 0x00C report back to the I2S block where the FIFO is signaled to either have an overrun or underrun.

5.4.2 register.v

The register.v is responsible for initializing all the signals and filter coefficients along with assigning all signals/coefficients to the proper address. The 16-bit filter coefficients are initialized

to zero and the rest of the signals are initialized to the hexadecimal default values given in Table 4.1. When the write enable (w_enable) and the write file transfer (wxfc) are both 1, a case statement is entered; given an address, the corresponding bit(s) within the address are broken down and assigned to the data written to the register. A second always block includes setting the read file transfer (rxfc) to not-write enable (~w_enable) and write file transfer (wxfc). An if-statement states that if not-write enable and write file transfer are 1, the case-statement can begin. When entering the case-statement, a specified address is can be called and the stored data can be read. The register.v allows data to be written to an address that correlate to specific signal(s), and allows data to be read from signal(s) that are stored in a specific address.

5.5 Test Fixtures

5.5.1 trig_generator_testbench.v

The purpose of the trig_generator_testbench.v is to write data to the address and trigger the trig_i2si_fifo_overrun_clr and trig_i2so_fifo_underrun_clr bits at the appropriate time. First the test fixture initializes the count and clk variables to zero to create a fresh simulation. The wdata is set to 8-bit address 8'hFF. The count variable counts to the hexadecimal value 20 where the 11-bit address is then initialized by 4 every time the count variable is incremented as seen in Fig 5.4. The rising edge of the XFC signals that the transfer of data has been completed, which allows data to be written to the address space and increment the address signal by 4 from 0000 to 0100. This test fixture writes to every address in order to ensure data is properly being written and stored.

Fig. 5.4: The Start of the Test Fixture Simulation

When the count variable reaches 20 and the hexadecimal value 0x00C is reached, the trig_i2si_fifo_overrun_clr and trig_i2so_fifo_underrun_clr bits are triggered to become 1 as seen in Fig 5.5. After the address 0x00C, or binary address 1100, goes low, the trigger bits go high for the length of the next address. This shows that the trigger bits are recognizing an overrun and underrun from the I2S FIFO because the conditions, XFC = 1, and address = 0x00C, are true. After the next clock cycle, the trigger bits will reset back to 0 because the I2S will have been signaled that an error has occurred in the buffer. A clock cycle after the address is incremented to 0x020, the XFC goes low to 0 and everything has reset again.

Fig. 5.5: Bits trig_i2si_fifo_overrun_clr and trig_i2so_fifo_underrun_clr are Triggered

5.5.2 trig_generator_testbench1.v

The second testbench for the trigger generator differs from the first testbench by initializing the address first to the 11-bit address 0x00C. Like the first testbench, when wdata is less than the address 0x020, wdata is incremented by 1 and the XFC is set to 1. The overrun trigger bit is signaled every clock cycle when the XFC goes high. The underrun trigger bit is signaled through the duration of four addresses, or two clock cycles, and when the XFC is set to 1. The testbench in Fig 5.6 indicates the trig_generator.v is fully functional because the overrun trigger bit is cleared every time data is trying to be written to the full I2S FIFO, and the underrun trigger bit is cleared every four addresses when the empty I2S FIFO is trying to be read.

Fig. 5.6: Bits trig_i2si_fifo_overrun_clr and trig_i2so_fifo_underrun_clr are Triggered in Second Testbench

5.5.3 i2c_reg_test_automated.v

To test the register.v file, a testbench was created that incorporated the I2C block. I2C was set up in a subsystem test with the register to check that coefficients could properly be written, read, and stored properly from the I2C into the register block. Test data was loaded into

the 512 two-part coefficients and the state of the coefficient registers were set to the 11-bit address 0x400. The slave address was set to the hexadecimal value 0xD5 with the 3-bit I2C address bits set to the binary value 101. After waiting 100ns for the global reset, a series of conditions were produced in order to serialize the slave address and the read/write bit, and serialize the two-part register address. The resultant waveforms generated showed the test data being properly written and stored within to the register.

Fig. 5.7: Test data properly being written from the I2C into the Register Part 1

Fig. 5.8: Test data properly being written from the I2C into the Register Part 2

5.6 C Code Involvement

5.6.1 initialize_coeffs.c

To initialize all 512 16-bit filter coefficients to a hexadecimal value of 0, a C program was generated in order to generate the thousand lines of code. Each coefficient is split into 8 bits, creating coefficients a and b. This created 1024 lines of Verilog code that was generated into a filter_coeffs_initialized text file that would loop 512 times and print the correct formatting of each

coefficient. The C program generated both coefficients a and b with a variable y incremented each loop cycle. The code outputs the two tabs such that when copying the code from the text file into Xilinx, it is formatted correctly. A pointer to the 16 bit hexadecimal 0 value is next to each coefficient in order to initialize all filter coefficients to zero.

5.6.2 set_coeffs.c

To assign all 1024 8-bit filter coefficients to the correct address, a C program was created to generate the correct number of tabs, case statement of the 11-bit hexadecimal address, and a new line of the name of the coefficient pointing to the 8-bit written data variable. In C, there is a compatible hexadecimal format that permits an integer to be incremented by hexadecimal 0x01 while still including the correct hexadecimal character formatting as hexadecimal should. After the address case statement, the correct name of the filter coefficient is expressed in which coefficients a and b are toggled between such that every other for loop iteration switches between coefficient a and coefficient b. The 1024 coefficients are properly formatted such that the contents of filter_coeffs_set.txt could be copied and pasted into Xilinx directly.

5.6.3 set_read_coeffs.c

The set_read_coeffs.c program generates the correct formatting for the case-statement that varies slightly from set_coeffs.c. The fprintf() statement is rearranged from set_coeffs.c to have rdata point to the signal requesting to be read, instead of having the signal point to wdata. The 1024 filter coefficients are created that same way as set_coeffs.c, but printed differently to the text file such that the filter block is able to read the filter coefficients values that are written in the first always block.

Chapter 6: I^{2}C Slave Interface – W. Forman

6.1 Introduction

The I^{2}C Slave interface of this chip serves a basic function to the system as a whole, to take filter coefficient values defined by the user outside of the chip and transfer them to the chip's register block. As an additional verification functionality it can then read the values from the register block to ensure that they were written properly. The I^{2}C bus was chosen over other busses such as RS-485, RS-232, CAN-bus and SPI because of its slave acknowledgement capabilities, multiple clock rates, low pin count and widespread usage in the industry. The function of the bus is to take in serial data from a master controller and convert it to a parallel format and then deliver it to the register block with the operation code, data and register address at the same time using a single strobe transfer for all parameters. If the operation is a read operation, then there is an added function to then take data information from the register block as parallel data and then transmit it back to the master I^{2}C controller over the bus in a serial format. The block can be seen in Fig. 5.1 of the overall system chip diagram highlighted in red.

I^{2}C was created and maintained by Philips Semiconductor now known as NXP semiconductor. Several updates to the original specification from 1982 have been made to keep up with the changing semiconductor industry to allow for faster transfer speeds up to 5 MHz from the original 100 kHz speed.

Fig. 6.1: I2C Chip Diagram Highlight

6.2 Requirements

When analyzing the scope of the project and the I2C-bus specification and user manual [2] document UM10204 from NXP semiconductor, several decisions were made about the functionality of the I2C block in regards to the necessity of the project and the time constraints to implement them. Several functions were not included in this project and the overall requirements were defined before any RTL coding was started. A more concise definition of the current requirements can be seen in section 5.2.1 and the tradeoff study which decided what functionality to include and exclude is found in section 5.2.2.

6.2.1 Current Requirements

Below is a list of the requirements for the current design of the I^2C block of the chip being made with reasoning and some details of each requirement.

- The I^2C-bus specifies that a slave of the bus must be able to respond to a master unit when called. There are 2 different slave address sizes to choose from, 7 bit and 10 bit, our group decided that the 7-bit address mode was optimal as there would more than likely only be 1 slave on the bus at any given time so the 10-bit addressing would not be necessary.
- The register address size is 12 bits so in the case of I^2C where data is sent only in bytes, 8 bits, before an acknowledge bit, the address space must be broken up into 2-byte sized data transmissions separated by acknowledge bits.
- The data to be written to each register is 8 bits so a normal byte transaction is sufficient to transfer the data to be written or read.
- The data transfer rate of I^2C comes in several speeds. For the needs of this project the standard mode speed of 100 kHz maximum and fast mode of 400 KHz maximum were considered. The chip clock frequency is to be 10 MHz so the fast mode plus at 1 MHz was considered to be too fast and could possibly cause problems as far as sampling at 1:10, so the fast mode with a ratio of 1:25 was chosen to be the fastest speed. Also the group considered that with an address space of 12 bits and a data width of 8 bits that all 4096 filter coefficients could be written in 0.0921 seconds, as seen in figure ##, which would be fast enough for our purposes.

$$512\ registers * 9\frac{bits}{transition} = 4608\ bits\ total$$

$$\frac{4608\ bits}{400\ kbits/second} = 0.01152\ seconds$$

Fig. 6.2: I^2C Full Coefficient Time Load Calculation

For this application, our chip needed only to be a slave for to a user.

- The burst write functionality is necessary to efficiently transmit large amount of sequential addressed data. Our I^2C interface will be made to have a start address transmitted and then repeated bytes of data come in. Part of the functionality of the I^2C block will be to

increment the address associated with each byte of data by incrementing the address from the original address with each incoming data byte.

- The read functionality will be similar to the write functionality where the master will transmit an address to be read from and then the I^2C block will be capable of serving repeated sequential read requests.
- The slave address of the chip could have been hardcoded into the chip, but it was decided by the group that having user selectable off chip switched for the 3 LSB of the address was appropriate. This will be realized through pull-up resistors or DIP switches in the FPGA board and test PCB that will be created when the final chip comes back from manufacturing.
- The transfer method from the I^2C block to the register block will be done with a simple strobe. When the data, address and operational code are ready for transmission then a single bit for 1 clock cycle will go high and then low.

6.2.2 Trade-off Study

Several I^2C functions found in the specification were left out due to the scope of this project. Below is a list of functions that were left out and short reasoning for them not being necessary.

- Clock stretching by a slave was considered unnecessary as our chip will not be running other processes during operation and there in theory will never be a time when it is busy and needs to delay transmission.
- Arbitration was deemed unnecessary as for our purposes we will be testing one chip on the bus at a time with a single master.
- Software reset was deemed unnecessary as this is an optional feature of I^2C and not widely utilized.
- Bus clear was deemed unnecessary as this functionality would be useful in critical applications or a product for sale, but not for the scope of this project.
- Device ID was deemed unnecessary as our chip is not made by a manufacturer, is a single and only design by our team and will be the only revision.

6.3 Top Block Interfaces

The I^2C Interface of this chip has only 3 external interfaces, the I^2C master unit, the register block and the user definable slave address pins. Below is a detail of how each interface was implemented and in section 5.4.2 some of the previous designs that were changed.

6.3.1 I^2C Master

The I^2C master interfaces with the chips I^2C slave module via a 2 wire interface using SCL and SDA lines which stand for serial clock and serial data respectively. Both of these single line wires are active high signals using pull-up resistors and the transitions of the signals are driven low by the transmitter during the transaction. The i2c_SCL input is only an input read by the slave

and driven by the master, but the SDA line has both input and output directionality to it. Although specifically called i2c_SDA_in and i2c_SDA_out these two input and output lines would meetup during the EDA tools floor planning and place & route stages to become one line by using a MOSFET open drain configuration as seen in Figure 6.4.

Data coming in on the SDA input line is sent in 1 byte increments so it was necessary that the deserializer be implemented in such a way that the first 2 bytes of data would be concatenated into a single 12-bit address register and the 4 MSB be discarded as they would all be zero.

During transmission of any data, the SCL line acts as a clock and the SDA line transition is only allowed during the low cycle of SCL. This fact was a large driver in signal filtering and conditioning of both SCL and SDA lines. To do this, it was decided that the SCL line would be double ranked through D flip-flops and the SDA line would be triple ranked through D flip-flops. This would stabilize the rise and fall times as well as filter out any spikes on each line that the logic circuitry would see. This also has a dual benefit as the signal transitions would now be matched in the chip's clock domain making it easier to use in RTL synthesis.

The i2c_SDA_out wire acts as the data transmission line for the data read request implemented by the serializer but also the acknowledgement sent by the slave when it sends an ACK bit by driving the SDA line low after each byte received.

The I^2C bus specification calls for many design details such as maximum bus capacitance, timing for different data conditions such as rise and fall times and frequencies. For the scope of the project for RTL synthesis rise and fall times and frequencies were considered as bus capacitance would not be a factor until EDA tools were being utilized as well as the test PCB being created. The frequencies for this block had already been chosen during the requirements phase of the project.

Fig. 6.3: I^2C block diagram in

Fig. 6.4: Open Drain Schematic

6.3.2 Register Block

Interface with the register block in contrast with the I^2C serial interface is much simpler. The interface between blocks is parallel where all bits are sent at once with a distinct wired connection for each bit and utilizing a single clock cycle strobe as a transfer signal rather than a fully handshake interface reduces wiring and complexity.

The output of the I^2C to the register block consists of 4 lines, i2c_op, i2c_addr, i2c_wdata and i2c_xfc_write and each has a specific role in the data transmission transaction. The i2c_op wire is the operational code of the transaction and is either a high for a read and low for a write. The i2c_addr wire is actually 12 bits wide and this designates the current register address of the register block to be written to or read from depending on the operation code. The i2c_wdata wire is a byte wide and it has on it the data to be written during a write request and is neglected during a read request. When all data is ready in the operational code, address and data output wires to the register block the i2c_xfc_write, which is the strobe, is driven high for 1 clock cycle at which point the register block is told to capture the data, address and op code and perform its action depending on the operation code. If a write is requested, then the register write the data to that address and if a read is requested then the I^2C block waits for the register to transmit back.

The input of the I^2C block from the register block consists of 2 wires, the first is i2c_rdata which is 1 byte wide and holds the data from the requested read address, and the second is i2c_xfc_read which is the strobe of the transaction where the wire goes high for 1 clock cycle telling the I^2C block to capture the data to be serialized on the isc_SDA_out wire of the I^2C master interface.

6.3.3 Slave Address Pins

The user definable external address pins select the 3 LSB of the chip's slave address. This would allow for more than 1 device to be put on the bus. The 3 bits would be selected by simply driving a pin on the chip's package or a selected pin on an FPGA to be driven high and therefore would result in a user defined LSB address. The other 4 bits of the address are coded into the RTL and cannot be changed. 1010 was selected as the 4 MSB as this address space was allowed and open according the I^2C bus specification. The logic for defining the address was not clocked and

at any given moment is exactly what is defined by the pins without any delay as it was deemed unnecessary to clock this logic.

6.4 Register Mapping

The interconnections of the I^{2}C block that do not interface outside the block connects all of the sub block together to form a cohesive unit. All of the data signals being transmitted can be seen in the table 5.1 register map as well as the control signals of the block diagram in Figure 6.3.

Table 6.1: I^{2}C Register table

Signal Name	Direction	Bits	Comment
clk	in	1	clock
rst	in	1	reset
i2c_scl	in	1	serial clock
i2c_sda_in	in	1	external pin combines in and out using open drain
i2c_sda_out	out	1	external pin combines in and out using open drain
i2c_op	out	1	1: write 0: read
i2c_addr	out	12	register address
i2c_wdata	out	8	data to be written for a write op
i2c_xfc_write	out	1	strobe transfer enable
i2c_rdata	in	8	read data
i2c_xfc_read	in	1	strobe transfer enable
i2c_addr_bits	in	3	External Pins, 3 LSB of slave address

The data signal flow between the deserializer and the sequencer is similar to the interface between the sequencer and the register block i2c_RW is the same as i2c_op as well as serial_data being the same as i2c_wdata in width and information. The wire burst_addr is similar to i2c_addr in the fact that they are the same width and represent the data register address but burst_addr is the first address specified by the I^{2}C master during a burst write or read. The sequencer increments the address as each new byte of data comes in. This is all controlled by the state of the machine and what is happening in regards to the data transitions being sent on the i2c_SDA_in and i2c_SCL interface signal.

6.4.1 States

The states of the I^{2}C block is driven by events on the i2c_SDA_in and i2c_SCL interface wires and how and when their transitions happen with regard to one another. After certain timed events in the amount of data byte transactions as well as start and stop conditions drive how the control logic of the block that can be in the four wires addr_ack, data_ack, slave_ack and stop.

After a reset or a stop condition the block is an idle state. During this state the deserializer is the only active block and it is looking for a start condition on the SDA and SCL lines. When this occurs, it is still the only active block and it starts taking in 1 byte of serial data from the I^{2}C master interface as the slave address call from the master and check it against the actual slave address. If

the address is not the same then the system stays idle looking for another start condition, but if it is a match then the state changes to look for the burst start address after sending a pulse on the slave_ack wire to control the serializer to send a single ack pulse on the i2c_SDA_out wire and tell the I^2C master that the slave is waiting for data.

Once this state is active the deserializer takes in 2 more bytes of data and concatenates them together to become the 12 bit burst_addr wire between the deserializer and the sequencer. Once the burst_addr has been captured then the slave_ack is driven high again as it does in between all received bytes of data to acknowledge to the master the reception of the data. The addr_ack also pulses so that the sequencer can take in the burst_addr and use it for sequencing.

Once these 2 bytes of data have been received as the burst_addr and the acknowledgement has been sent a state change is driven in the deserializer and now it knows that depending on the opcode bit that the data coming in will be of data to be filled into registers or that the address and opcode should be sent to the register block and a read was requested.

If a write request happens then after each data byte transmitted the deserializer pulses slave_ack for the acknowledgement over SDA as well as pulsing data_ack to the sequencer to notify it that the data has been captured is on the serial_data line between the deserializer and the sequencer. Every time that the data_ack is pulsed, then the sequencer updates the register address and the data and when it is ready it strobes i2c_xfc_write for one chip clock cycle to send the information to be written and then clears itself. This will happen continually until a stop or reset condition occurs. It is possible for a repeated start condition to occur after the slave acknowledgement in which case the slave looks for a slave address again. It is possible that in the event of a stream write data coming in that it will in fact be more than what the register block has allocated as valid address space. During future testing the team will decide if this functionality will be necessary as a control for invalid data and addresses. This would enlarge the size of I^2C block and its complexity as there would be a need for an entire register block map to be coded into the sequencer block. This itself could cause unwanted operation so it will have to be decided as a group during testing to see if it is necessary or not. If a read was requested in the opcode, then the sequencer does not wait for data to be ready as it is not necessary and the sequencer will prepare the address and opcode to be strobe transferred to the register block. If a repeated start condition is sent in lieu of an acknowledgement by the master, then this sequence will look again for a slave address and start the process over again.

The states can always be altered by a reset from the chip or from a stop condition seen on the I^2C master interface. Either of these will force a reset of the whole block and goes back to state 0 and look for a start condition. With the structure described above the deserializer block is the state machine for the entire I^2C block as it drives the state of the other blocks depending on what the master controller is requesting.

Fig. 6.5: I^{2}C State Machine Flow Diagram

Fig. 6.6: I^{2}C Previous block diagram versions

6.5 Sub Blocks

The I^2C block was broken up into 3 distinct parts including the deserializer, the sequencer and the serializer. The design considerations are outlined in section 5.5.4 and details about the design of each sub block is outlined below in the following sections.

All sub-blocks use always statements for posedge clock and negedge reset to control when a reset happens. If statements are used inside of the always blocks to decide what happens based upon the state of each sub block. All always statements include as a first if statement a reset capability of all registers to be sent back to initial conditions.

6.5.1 Deserializer

The deserializer of this block became the defacto state control machine for the other sub blocks as it would be being controlled by the I^2C master. The deserializer block has to perform several functions beyond simply deserializing the data coming from the master.

The deserializer must filter and condition the isc_SCL and the i2c_SDA_in signals in order to make them usable in the Verilog code. Edge detection pulses were extracted from each line after it went through the D flip flop ranking described above. States of the lines were also made during this conditioning step to have chip clock synched signals that were highly reliable during RTL synthesis.

To deserialize the data, there are 2 known possibilities in Verilog to accomplish this. The first being a shift register and the second being a case statement. Although the shift register was simple, the case statement seemed to give more control of multiple events occurring at the same time. The case statement is very similar to using a MUX where individual bits of a register are populated as a bit counter increments and sequential case statements become active. This allowed for the 8th case statement to include control signals to acknowledge, state change, and transfer data to different registers. Although logic intensive, this gave the designer a great deal of ease when it came to doing the multiple functionality of the block. The slave address, register address and data are all deserialized in the same way, but the different states of the machine drive how the control signals are handled.

The deserializer has to look for start and stop conditions coming from the I^2C master. These conditions either turn on the deserializer to start deserializing data as well as checking the slave address coming in against the slave's programmed address as well as extract the read or write enable bit from the LSB of the slave address sent and assign it to the i2c_RW line. Once this happens, a state change occurs and an acknowledgement signal sent to the serializer if the addresses match or the deserializer resets itself and continues looking for another start condition.

Once a state change occurs from the slave address check then the deserializer goes into register address fetch mode where it takes in 2 bytes of serial data from the I^2C master and then concatenates them to form the 12-bit register burst_addr signal. Once this occurs, another acknowledge signal is sent to the serializer in between the 2 bytes sent and after the second byte sent. At the same time the addr_ack signal is enabled going to the sequencer sub block to inform it that the address is ready to be received.

Depending on the i2c_RW bit at the strobe of the addr_ack signal, the deserializer will either reset itself if a read is requested as it has no purpose in the future of the transaction or it will go into data fetch mode. During data fetch mode the deserializer will continually deserialize data write it to the serial_data wire and then strobe the data_ack wire and then loop itself for another byte of data until a stop condition occurs.

The deserializer also has the capability to send out a stop signal when either a stop condition occurs or a reset occurs to ensure all other sub blocks have reset themselves.

Fig. 6.7: I^2C Deserializer Block Diagram

6.5.2 Sequencer

The sequencer block acts as slave of sorts to the deserializer block. Its sole purpose is to transfer data, address and operation code information to the register block. Depending on the acknowledge signals coming in, it acts accordingly.

The sequencer acts similarly to the deserializer in that it conditions the acknowledge signals into single cycle pulses so that a signal is not captured more than once with the system clock always blocks. From this it will capture the address and data when these strobes occur.

This data is handled in two separate ways: the first during a read and the second during a write.

Upon a read opcode the sequencer simply captures the burst_addr into the i2c_addr output line and then strobes the i2c_xfc output line, and then after that resets itself. This is done using 3 if statements under the main always statement. A state variable for this sequence is used called xfc_ready which goes high when the data is captured so that at the next clock cycle the strobe occurs. Using the opcode as part of the if statement makes sure that this only occurs when a read is requested. The last step is to reset each signal again using an if statement and the i2c_xfc to control it. The opcode acts as a control object for how the I^2C block interacts with the register block determining a read or write transaction.

During a write operation a very similar sequence occurs, but the strobe does not occur until a data-ack signal strobe has been seen. Using a similar if statement setup and the same state variable. This same variable is used because the if statements can be differentiated by the operational code. This sequence takes 4 if statement to complete and the first is the same as the

read sequence where the address is captured as well as the opcode. Next the sequence waits until a data_ack signal is received to capture the data and then to enable xfc_ready. During this time another variable comes into play. The address is rewritten to have an increment variable added to the address. For the first instance of this if statement a 0 is added, but later the increment itself increments by 1 so during a burst write cycle the address is incremented each time this if statement goes active. The next if statement strobes the i2c_xfc for the opcode, address and data to be sent to the register block. This sequence will continue to loop and not reset itself until a stop condition is seen as a reset or a stop signal from the deserializer.

Fig. 6.8: I^2C Sequencer Block Diagram

6.5.3 Serializer

The serializer, like the sequencer, acts as a slave of sorts to the register block and to the deserializer. Its sole purpose is to either send acknowledge bits during the 9 bit of the byte transaction over the SDA line or to serialize the data that it receives from register block. It has no output except that of the SDA line.

Similarly, the serializer conditions that data on the SDA_in and SCL lines so that it is aware of when it can make transitions to the SDA out line which is during SCL low states.

Start and stop conditions are handled by using the slave_ack and i2c_xfc_read signals. If either one of these are asserted, then the system goes into a start condition.

To serialize the data there is an if statement under an always block that uses a new variable of serialize_done and the i2c_neg_edge_pulse signals. Inside of this if statement there is a case statement that will then write to the i2c_SDA_out line to serialize the data using the captured i2c_rdata from the register block during the i2c_xfc_read strobe sent by the register block. This case statement also uses a counter to sequentially increment the case statement MUX and at the last of the 8 bits it then writes to the serialize_done register. This register will then deactivate the if statement so no more data is written and causes a stop condition inside the sub block.

If the acknowledge signal is to be serialized, then another if statement under the same always block is enabled using the slave_ack signal and the i2c_neg_edge_pulse. Only 2 statements exist under this if statement and one is to set i2c_SDA_out high and then activate the serialize_done register. This will cause a stop condition and the whole sub block goes back to waiting for a signal from the deserializer of the register block.

There is essentially a mini state machine inside of the serializer that dictates how the data is serialized taking into account what is being serialized, an ack or data, and if that has occurred or not.

Fig. 6.9: I^{2}C Serializer Block Diagram

6.7 Test Fixture Implementation

Testing for this block using Verilog test benches were employed by creating an I^{2}C master in Verilog inside the test bench. A clock was defined that at 400 kHz to simulate an I^{2}C SDA line and then the line was driven by the Verilog master in the test bench. Data for testing such as the slave address, register address and register data were defined in the test bench and could be manipulated to create different test stimulus. The test bench rubric that was now created is used for several tests to verify the I^{2}C block is functioning the way it should namely with regard to full read and write transactions.

Earlier versions of test benches were also used for some simpler tests including start and stop condition, reset, deserialization of data and response to slave address. These test benches utilized hard coded, bit banged if you will, SDA and SCL transitions to mimic stimulus on the I^{2}C bus similar to the full transaction test benches but simpler in usability as specific odd transitions could be made easily such as random stop conditions or clock delays.

6.7.1 Start, Stop, Deserialize Address

These tests utilized the simpler test bench rubric and multiple different stimulus patterns were created. Start conditions were created and every time the I^{2}C block responded correctly and transitioned into state 1 where it was ready to deserialize a slave address.

Stop conditions regardless of their timing in the transaction always resulted in the deserializer sub block going back to state 0, erasing its data contents and sending out the stop to the other sub blocks to stop them if they were currently performing a task and resetting their temporary values back to state 0/reset conditions. Several of these were performed at normal places during the transaction and also randomly during a data transfer and the results were consistent.

When the correct start conditions were seen and data stimulus was driven on the SDA line by the test bench the I^{2}C deserializer performed as expected and deserialized the address correctly each time. Once that was completed with 1 clock cycle the address was checked against the defined address pins and acted upon it. It was either driven into state 2 to start deserializing the address bytes while simultaneously sending out an ACK signal to the serializer sub block.

The serializer sub block performed as expected, albeit with some tuning along the way, to send out correct stimulus to the open drain circuit that would eventually be crated in the chip.v module. Multiple addresses were made and on all tests the I^{2}C module responded correctly to a right or wrong address at multiple different addresses.

Fig. 6.10: I^{2}C Start, Stop Deserialize ISIM

6.7.2 Write Registers

In order to do a full write transaction, and then a full register transfer of all 512 filter coefficients a more automated solution was needed. For this the I2C master that was written for the test benches was utilized. A full set of 1024-byte size coefficients was created for this test module. Each coefficient was numbered in ascending order to go with the register it was supposed to go with, this made checking a simple task in Isim where several samples were analyzed throughout the registers and all were in place. A smaller version of this was used to set the chip status register to set the floating point offset and the BIST generator/I2S select. In conjunction with Kevin and Dhruvit this test also tested the I2S and BIST to confirm that they were working properly when selected.

Fig. 6.11: I^{2}C Isim Full Register Write Transaction (Truncated)

6.7.3 Read Registers

The read register test utilized the same I2C master test bench with relevant data to read specific registers. This I2C module is made to only read a single byte at a time and then call another register to read. This test was performed on a few random registers that was loaded previously by the same I2C master in the test bench. It was modified to do both transactions and what was read back was exactly what was written.

Fig. 6.12: I^{2}C Read Register Transaction

6.8 PSOC Controller

In order to control the FPGA once it has been programmed there was a need to control the filter by writing and reading back filter coefficients via the I2C interface. The readily available PSOC 5LP was utilized due to its ease of use and predefined I2C functionality. Several control algorithms were created to send sets of filter coefficients defining a low pass, high pass and band pass filters.

6.8.1 Structure/Interface

A push button interface was written in PSOC Creator so that there would be three buttons and each would start an algorithm to transfer all 512 filter coefficients depending on the filter envelope selected and then exit and wait for more input.

6.8.2 Test Data

Separate header files were added to the project to define each filter type individually. These files were created by modify a C code script that helped create the coefficient registers themselves for the RTL design of this project and use it to create the list of 1024-byte size coefficient parts from Matlab where the filters were designed.

While this worked for testing purposes for this project, as a user application it would be desirable to be able to create any type of filter on the chip running the I^{2}C master so that a few rotary encoders or potentiometers could change the filter parameters, (center frequency, Q and attenuation). This is above and beyond the scope of this project so this is just a thought on usability of the chip.

Chapter 7: Simulation/Verification – K. Cao and D. Naik

To verify the RTL design of the audio filtration ASIC, multiple simulations were run in Xilinx. Each simulation tested for a different filter configuration. A test bench for the top module of the chip was created. Each simulation was given the same input, a square wave. 10,000 words were inputted into I2S for each simulation. The input data values described a square wave with an amplitude of 3,276 and -3,276. This value is a tenth of the value of the max digital audio value for CD quality audio. This value was chosen to ensure that no overflow occurs within the filter block. If overflow occurred, the output audio would not be considered to be valid for testing. To ensure that the serial data inputted to the I2S block was being outputted instead of the BIST data, the enable signals for the I2S deserializer was set high and the enable signal for the multiplexer was set low.

The first type of verification that was performed was a simple pass through of data. The filter allows only the current input to output. The output of the chip was compared with the input data to ensure that the data matched exactly. This verified if data was moving through the chip correctly. Later tests would verify if the ASIC was capable of applying the correct filter. To have the input to simply pass through the filter, the very first filter coefficient was set to 1, and the rest of the 511 coefficients were set to 0. These coefficient values were changed within the register block when reset is asserted high. Then to simply apply the coefficients the reset signal is asserted.

The verification process for the lowpass, highpass, and average filters were similar to the pass through verification. The only difference in these verifications are the coefficients values. MATLAB was used to calculate the different coefficient values for each filter. Using the Filter Design and Analysis tool, or FDAtool, in MATLAB was used to calculate the different coefficient values for the lowpass and highpass filter. Matlab produced coefficient values that were in decimal format. Using the Java Program, the coefficients were converted into pairs of 2's complementary binary numbers for each coefficient value. The register block was then edited each time when applying a new filter.

To verify the data for the lowpass, highpass, and average filters a model was created with Java. The Java model calculated the filtered values of the input data and stored the values within a text file. Within this file each line contained a word that was filtered by the chip. The various simulations within Xilinx also outputted a text file of the filtered input serial data in the same format as the Java model text file. Both text files were compared with each other to verify if the chip was working as intended.

Our initial tests for the low pass and high pass failed. The output for the Verilog simulations had similar shapes, but looked time shifted compared to the model. We hypothesized that there was a startup associated with the Verilog and the filter was not receiving the first couple I2S inputs due to it. After further digging, this was found to be true. After adjusting the input signal for the model to match the signal entering the filter module, the tests passed.

The filter takes time to reach steady state, dependent on the magnitude of the number of taps of the filter. For our FIR filter, the first 512-1024 inputs are deemed invalid. This is due to the filter not having enough past inputs to convolute correctly.

Our results are below. The first figure contains the impulse response of the filter. The second figure contains samples 1000 to 2000. This is when the filter reaches steady state.

Fig. 7.1: Lowpass Filter: Impulse Response

Fig. 7.2: Lowpass Filter: Square Wave Samples 1000 - 2000

Fig. 7.3: Highpass Filter: Impulse Response

Fig. 7.4: Highpass Filter: Square Wave Samples 2000 - 3000

Fig. 7.5: Average Filter: Impulse Response

Fig. 7.6: Average Filter: Square Wave Samples 1000 –2000

Chapter 8: FPGA Implementation – Z. Nelson and W. Forman

8.1 Programming the FPGA

The FPGA implementation was primarily a way to show that our design worked in realistic setting. The first step of the FPGA portion of the project was purchasing a Nexys 4 Artix-7 FPGA. After we received the FPGA, we needed to assign pins to the inputs and outputs of our design. Table 8.1 shows how we mapped the signals to the FPGA pins. The pin assignments were done using the PlanAhead software that is included in the Xilinx ISE Design Suite.

Table 8.1: Verilog Signal to FPGA Mapping

Signal Name	Direction	FPGA Pin	PlanAhead Site
clk	in	internal (crystal)	E3
rst_n	in	reset button	C12
i2si_sck	in	JB3	V11
i2si_ws	in	JB4	V15
i2si_sd	in	JB7	K16
i2so_sck	out	JB8	R16
i2so_ws	out	JB9	T9
i2so_sd	out	JB10	U11
i2c_addr_bits	in	JA9:JA7	DC18, C17, G13
i2c_scl	in	JA1	B13
i2c_sda_in	in	JA2	F14
i2c_sda_out	out	JA3	D17

Since we now had synthesized Verilog code and pin assignments, we could generate a programming file and download it to the FPGA. A FGPA programming guide was created and the steps are listed below.

Step #1: Click the Synthesize button in Xilinx ISE. (This will take several minutes)

Step #2: Click the Implement Design button in Xilinx ISE. (This will take several minutes)

Step #3: Click the Generate Programming File button in Xilinx ISE. (This will take several minutes)

Step #4: Plug in FPGA via USB.

Step #5: Under “Configure Target Device”, click the Manage Configuration Project (IMPACT) button. The ISE iMPACT tool window will now open.

Step #6: Run the wizard by clicking Edit\Launch Wizard. The “Welcome to iMPACT” window will open.

Click OK.

Click Yes.

Select the .bit file that you generated in Step 3.

Select No.

Click OK.

Step #7: If there are no errors, you will see an "Identify Succeeded" message. Right-click on the green Xilinx box and click program.

Step #8: Look at the port mappings in rtl_description.xlsx and configure test setup.

8.2 I2S FPGA Testing

Since our FPGA was now successfully programed, we needed to create I2S and I2C PSoC programs that could test the FPGA. Fig. 8.1 below shows the FGPA wired up to two PSoC microcontrollers (the I2S one is showed in the picture). The I2S program was created by Ian Patel and the I2C program was created by Whitley Forman. The I2S program was originally intended to generate and capture back I2S data to verify that the filtering component of the design was working. However, there was some kind of bug in the I2S capture that we could not fix. For some reason, we could not get accurate data captured back to the PC. The I2S program was successfully

able to generate different types of input waveforms including sinusoidal, square, and triangular waves.

Fig. 8.1: FPGA Testing Set Up

Without the I2S capture working we were still able to do some testing of the FPGA. We first set the default filter coefficients to create a pass-through filter (the output signal is the same as the input signal). We then applied a square wave to the input since it can easily be observed on an oscilloscope. Fig. 8.2 show the serial clock and word select that was taken from the chip. Note that the input and output serial clock and word select matched in frequency. Fig. 8.2 also shows the square wave input and the corresponding output. It can be seen that output exactly matched the input besides being slightly delayed. This simple test gave us some physical proof that our design's data flow worked not only in simulation but on hardware.

Fig. 8.2: FPGA Waveforms (Pass-Through Filter)

8.3 I2C FPGA Testing

Testing the PSOC controller with FPGA presented a few problems that had to be worked through. Grounding, external pin assignment, and the open drain circuit were all factors that caused delays for the I^2C testing for implementation. These problems were worked through systematically and methodically and now the PSOC and FPGA are working in conjunction with each other with the FPGA responding to the stimulus from the PSOC.

Grounding was a key factor to get stable waveforms and after initial tests were done, analyses of the waveforms suggested that a grounding issue may be present. Also noticed is that the I^2C slave was not responding to the master. It was later found that the I^2C slave address pins that were routed to external pins were not being driven low or high and therefore were floating. The PSOC was addressing for a low so the pins were grounded through 1k ohm resistors and the slave immediately began responding.

At the same time there was also a need to revisit how the open drain was defined. The conditional statement in the chip.v file where all external pins are defined it was seen that there was previous miscommunication as far as how the open drain would be operated. The problem was expeditiously fixed inside the I^2C serializer sub block and the conditional statement modified. This relatively simple change corrected the issue of multiple drive of the SDA line and now works as expected.

6.8.4 Future Work for Usability

As of right now testing, troubleshooting and modification still continues on the read sequence as the transaction is getting cut short for some reason. Work will continue on this module until the issue is resolved and full working I^2C module is ready for next year's team. Contact information has been given to both of next year's teams to assist in the development of the RISC-V and the Digital Filter Chip designed in this project.

Chapter 9: EDA Tools and Physical Design – D. Naik, J. Swift, and K. Cao

9.1 Setting up Mentor Graphics

Once Pyxis has been installed on a Linux machine, users need to set up environmental variables to use the products. If the user is using a bash shell, a script can be placed in the .bashrc folder. This script will run every time the user logs in, which allows the user to avoid setting the variables on every log in. The instructions are the following.

1. cd /home/username/
2. vi .bashrc
3. insert following script

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
export MGC_HOME=/home/eda/pyxis/pyxis_v10.5_3/
export MGLS_LICENSE_FILE=1717@lmservices2
```

4. logout and re-login

Once the environmental variables are set up, the user can invoke the Pyxis tool from the command line. Use *$MGC_HOME/bin/dmgr_ic* to launch the Pyxis project manager.

The user will see the following screen:

Fig. 9.1: Project Manager Window

Within this screen, click on "Create New Project" at the top right.
A project wizard will pop up. Select the project path. For this example, we are using the *generic13* libraries provided from Mentor Graphics. Find the path for the libraries. Click ok.

Fig. 9.2: New Project Wizard – Screen 1

In the next screen, click on "Add Standard Libraries" this will populate the screen. Click ok. The project is now set up. Within the manager, schematics and layouts can be created.

Fig. 9.3: New Project Wizard – Screen 2

9.2 Physical Design

Given the time constraint of the project and being only given the generic libraries, we decided to make a physical design of passive components to send out to manufacturing. The layout would consist of one resistor, fifty-six spiral inductors, and three capacitors. The fifty-six inductors were places in series to create a higher inductance and the three capacitors were places in parallel to create a high capacitance. Using polysilicon on the predefined layer POLY2, a resistor was created using the rectangle function. Figure 9.4 shows a 25ohm resistor that measures 38 by 40 μm in dimensions. Each rectangle is 2 μm wide with the spacing of all polysilicon wires to be at least 2 μm apart. The 3 by 3 μm contact was created from metal on the predefined layer M1 with a 2 by 2 μm carbon-oxide (CO layer) on top of the metal. These contacts serve as connectors to wire the like components together and also wire the components to the top layer of the chip where they will be assigned to pins.

Fig. 9.4: 38 x 40 μm 25Ω Resistor

The inductor was the next component created from metal on the predefined layer M1. If time permitted, we would have produced a coiled conductor as well, but for now, the spiral inductor was chosen to be the type of inductor used on the chip. Taking into account the dimensions, the coil spacing, the number of turns, and the wire width, the inductor was calculated to be 1.068nH. The Stanford Microwave Integrated Circuits Laboratory (SMIrC Lab) produces an integrated spiral inductor calculator was available online in order to calculate the inductance based on the dimensions. This inductance is very small to measure when getting back the manufactured chip, therefore 56 inductors were placed in series in the layout in order to create a total inductance of 59.808nH, which may be a little easier to measure. Figure 9.5 shows a 56 by 58 μm spiral inductor. The same dimensioned contacts as before were placed at the beginning and end of the inductor using VIA1 as the layer that connects the M1 inductor to the M2 wire.

Fig. 9.5: 40 x 40 μm Spiral Inductor

The last component created was the capacitor. This component was created using polysilicon and metal. The polysilicon base measures 20 by 40 μm. A 2 by 23 μm metal wire was placed 2 μm down from the top of the polysilicon and 2 μm up from the bottom of the polysilicon. The wires extend 5 μm onto the polysilicon and the two 18 μm wires that extend out of the polysilicon are connected by a 2 by 36 μm wire. Once again, the same dimensioned contacts using the VIA1 layer and CO layer were placed onto the M1 wires of the capacitor in order to connect to the M2 wire that will connect all three capacitors in parallel. Figure 9.6 shows the capacitor with an unknown capacitance. The capacitance was not able to be found due to the lack of resources that could be gathered in the short amount of time that was given to create these passive components.

Fig. 9.6: 38 x 40 μm capacitor

Chapter 10: Conclusion – Z. Nelson

In conclusion, we were able to successfully deliver fully functional Verilog code that satisfies the goals of the project. We performed extensive testing via testbench simulations and additional testing on a FPGA using different input signals. We also went through the process of installing EDA tools and using them to create a simple circuit with a resistor, inductor, and capacitor on it. Documenting the chip design process and code we created will definitely save next year's groups time. Overall, one of the most valuable skills we learned was being able to write high quality synthesizable Verilog code. Some of the team members plan on using this skill in industry after graduation. Overall, this senior project was a valuable experience for us because we learned about the many steps that need to be completed in order to produce a chip.

References

[1] Mosis.com, 'About Us', 2015. [Online]. Available: https://www.mosis.com/what-is-mosis. [Accessed: 11 November 2015].

[2] nxp.com, '*I^2C-bus specification and user manual - UM10204*', 2015. [Online]. Available: http://www.nxp.com/documents/user_manual/UM10204.pdf. [Accessed: 10 November 2015]

Appendix A: Project Overview

Biography 92
Engineering Standards and Realistic Constraints Form 94
Engineering Standards, Specifications, and Codes 95
Modern Engineering Tools 96

Biography:

Kevin Cao:

- Kevin is from Morris Plains, NJ and is a Computer Engineering major who is planning to enter the workforce after graduating at TCNJ. Kevin has interned as a software engineer at LGS Innovations, located in Florham Park, NJ.

Whitley Forman:

- Whitley is from Ocean Grove, NJ and is an Electrical Engineering major who is continuing his education in the electrical field and will be obtaining his Master Electrician's license after graduation. He is planning on using his new knowledge and experience with his current business to expand into new ventures.

Dhruvit Naik:

- A resident of Mount Laurel, NJ, Dhruvit is a senior Computer Engineering major at TCNJ. He plans on entering the workforce after graduation and continuing his education in the coming years. He is the Vice-President of a startup, ThinkSOAS, INC.

Zachary Nelson:

- From Cream Ridge, NJ, Zachary is a Computer Engineering major who has enrolled in a Ph.D. program at Johns Hopkins University specializing in controls starting in Fall 2016. He has experience as a software engineering intern at Teletronics Technology Corporation and an undergraduate student researcher at TCNJ as part of the MUSE program.

Julie Swift:

- From Robbinsville, NJ Juliann is a Computer Engineering major who is planning on entering the workforce after college. She has experience as a AutoCAD Designer interning at Linearization Technology and a Software Development Life Cycle Analyst interning at Educational Testing Services. She participated in the undergraduate student research program, MUSE, at TCNJ.

Engineering Standards and Realistic Constraints Form:

SCHOOL OF ENGINEERING
The College of New Jersey

Realistic Constraints and Engineering Standards

PROJECT: Design and Verification of a Complete Application Specific Integrated Circuit

Team Leader: Zachary Nelson

Checklist Completed: _____ Technical Advisor 4/5/16 Date

DIRECTIONS:
This checklist should be completed by the team leader and signed by the technical advisor.

In the self-check column, place a C if covered in body of report, a N if not covered in body of report but is covered in the remarks column, and a N/A if not applicable (in all cases, C, N, or N/a, include a justification in the remarks column).

	Self-Check C	N	N/A		Remarks
Economic	X				The EDA tool cost and other economic constraints are discussed in the report. Specifically, the economics of custom chips versus FPGAs will be discussed.
Environmental		X			The only environmental concern with our project is how much electricity it takes to run the chip and where that electricity is coming from. Since full custom chips are more power efficient than FPGAs, power consumption is a major design requirement for many applications.
Social			X		There are no social constraints for this project since the chip will not be manufactured for public consumption.
Political		X			The project cannot violate any patents or laws.
Ethical			X		The project is used for audio filtering, which does involve any ethical issues.
Health and Safety		X			The major safety concern with this project is making sure that no wires are exposed when testing the design. Another safety concern is to understand what chemicals would be emitted if the chip started smoking.
Manufacturability	X				The manufacturability and fabrication of the chip is discussed in the report.
Sustainability			X		The project uses very minimal resources (mostly software), so sustainability is not a constraint.
Standards	X				The project must comply with the I2S and I2C standards included in the report.

Engineering Standards, Specifications, and Codes:

- *DIP Package:* Dual in-line package (DIP) is a packaging standard that includes two rows of pins that extend from a rectangular plastic material. The spacing between the pins are meant to be compatible with breadboards. The details of this standard can be found in Appendix E. Joint Electron Device Engineering Council (JEDEC), "JEDEC Publication 95", pp. 306-307.
- *GDSII Stream Format*: Graphic Database System (GDS) II is a binary file format that is commonly used in industry for transmitting the design of an integrated circuit. The details of this standard can be found in Appendix E. Calma Company, "GDSII^{TM} Stream Format Manual", Documentation No. B97E060, Release 6.0, February 1987.
- *I2C:* A serial communication standard that we used for uploading and reading registers to our FPGA. The details of this standard can be found in Appendix E. NXP Semiconductors, "I^2C-bus specification and user manual", Rev. 6, April 2014.
- *I2S:* A serial communication standard for transferring digital audio. The I2S bus uses a bit clock line, word clock line, and data line. The details of this standard can be found in Appendix E. Phillips Semiconductors, "I^2S bus specification", February 1986.
- *Verilog IEEE Standard 1364-2001:* A hardware description language (HDL) that can be used to design digital systems (such as an FPGA). This language is a high-level abstraction of a design that can be synthesized into low-level logic gates.

Modern Engineering Tools:

- *CORE 9*: System design software that we used to model the basic structure of the system. We also used it to keep track of the system requirements and use cases.
- *DropBox*: Cloud storage service that we used to store all confidential materials.
- *Git/GitHub*: Revision control software that allows a group to keep multiple versions of source code. We stored all documents and code in the public GitHub repository.
- *Red Hat® Enterprise Linux® Operating System*: Operating system that was required for all Mentor Graphics software (ModelSim and Pyxis).
- *Microsoft Project*: Project management tool that we used to keep track of the group's schedule and progress.
- *ModelSim HDL Simulator*: Simulation and verification tool that we used to test our Verilog code.
- *Pyxis Custom IC Design Platform*: Multiple pieces of software that we used for the design of our simple chip.
- *Xilinx ISE Design Suite*: Design environment for Xilinx FPGA products. We used this tool for writing Verilog code, writing Verilog test benches, debugging Verilog code, synthesizing our design, and programming the FPGA.

Appendix B: Management

Schedule 98
Meeting Minutes 100
List of Contacts 123
Safety Considerations 126
Material List 127
Financial Budget 128

Schedule:

	Task Name	% Comp	Finish
1	System Design	100%	Mon 9/14/15
2	General Planning	100%	Thu 6/18/15
3	Complete I2S In Document	100%	Wed 9/9/15
4	Complete I2S Out Document	100%	Wed 9/9/15
5	Complete Filter Document	100%	Wed 9/9/15
6	Complete I2C Document	100%	Wed 9/9/15
7	Complete Register Document	100%	Wed 9/9/15
8	Finish System Design	100%	Wed 9/9/15
9	Document System in CORE	100%	Wed 9/9/15
10	RTS/RTR and XFC Protocols	100%	Wed 9/16/15
11	Create a Detailed Schedule	100%	Mon 9/14/15
12	Install Microsoft Project	100%	Mon 9/14/15
13	Verilog RTL Coding	100%	Wed 1/6/16
14	Create i2si.v	100%	Sat 10/31/15
15	Create and Test i2si_fifo.v	100%	Fri 9/18/15
16	Create and Test i2si_deserializer.v	100%	Wed 9/30/15
17	Create and Test i2si_bist.v	100%	Wed 9/30/15
18	Create i2si.v Top-Level	100%	Wed 10/21/15
19	Test i2si.v	100%	Sat 10/31/15
20	Create i2so.v	100%	Sat 10/24/15
21	Create and Test i2so_fifo.v	100%	Wed 9/23/15
22	Create and Test i2so_serializer.v	100%	Wed 10/7/15
23	Create i2so.v Top-Level	100%	Wed 10/21/15
24	Test i2so.v	100%	Sat 10/31/15
25	Create filter.v	100%	Sat 10/31/15
26	Create and Test filter_convolution.v	100%	Mon 9/28/15
27	Create and Test filter_accumulator.v	100%	Fri 10/2/15
28	Create and Test filter_fifo.v	100%	Wed 10/7/15
29	Create and Test filter_round_shift_clip.v	100%	Fri 10/16/15
30	Create filter.v Top-Level	100%	Wed 10/21/15
31	Test filter.v	100%	Sat 10/31/15

	Task Name	% Comp	Finish
32	**Create register.v**	**100%**	**Fri 10/30/15**
33	Create and Test register_trig_gen.v	100%	Wed 10/14/15
34	Create register.v Top-Level	100%	Wed 10/14/15
35	Test register.v	100%	Wed 1/6/16
36	**Create i2c_slave.v**	**100%**	**Fri 10/30/15**
37	Create and Test i2c_slave_deserializer.v	100%	Wed 9/30/15
38	Create and Test i2c_slave_sequencer.v	100%	Wed 10/14/15
39	Create and Test i2c_slave_writetoreg.v	100%	Wed 10/14/15
40	Create and Test i2c_slave_serializer.v	100%	Wed 10/21/15
41	Create i2c_slave.v Top Level	100%	Mon 10/26/15
42	Test i2c_slave.v	100%	Sat 10/31/15
43	**Create chip.v**	100%	**Fri 10/30/15**
44	**Finish RTL Design**	100%	**Fri 10/30/15**
45	Install EDA Tools	100%	Wed 2/10/16
46	Determine EDA Requriments	100%	Sun 9/6/15
47	Install OS	100%	Fri 9/25/15
48	Install EDA Tool	100%	Tue 9/29/15
49	Install Libraries	100%	Wed 2/10/16
50	Create a Simple Test Case	100%	Mon 10/5/15
51	Simulation/Verification	100%	Wed 3/2/16
52	Test i2s_in.v	100%	Wed 1/20/16
53	Test i2s_out.v	100%	Wed 1/20/16
54	Test filter.v	100%	Wed 1/20/16
55	Test chip_reg.v	100%	Wed 1/20/16
56	Test i2c.v	100%	Wed 1/20/16
57	Test Control Flow Subsystem	100%	Wed 3/2/16
58	Test Data Flow Subsystem	100%	Wed 3/2/16
59	Test chip.v	100%	Wed 3/2/16
60	Synthesis	100%	Wed 3/9/16
61	Synthesize i2s_in.v	100%	Wed 2/10/16
62	Syntehsize i2s_out.v	100%	Wed 2/10/16

	Task Name	% Comp	Finish
63	Synthesize filter.v	100%	Wed 2/10/16
64	Synthesize chip_reg.v	100%	Wed 2/10/16
65	Synthesize i2c.v	100%	Wed 2/10/16
66	Synthesize chip.v	100%	Wed 3/9/16
67	FPGA Implementation	100%	Wed 4/6/16
68	Research FPGA's	100%	Tue 10/27/15
69	Purchase FPGA	100%	Thu 10/29/15
70	PSoC for I2S Stream	100%	Wed 1/27/16
71	Implement Design on FPGA	100%	Wed 3/23/16
72	Test FPGA	100%	Wed 4/6/16
73	Physical Design	100%	Wed 4/27/16
74	Database for MOSIS	100%	Wed 5/4/16
75	MOSIS Fabrication Deadline	100%	Wed 5/4/16
76	Documentation	100%	Wed 5/4/16
77	Create Web Page	100%	Fri 10/2/15
78	Write Final Report and Presentation	100%	Wed 12/2/15
79	TCF Poster	100%	Sat 3/19/16
80	Final Presentation and Report II	100%	Wed 5/4/16

Meeting Minutes:

Chip Requirements (June 11th Meeting)

Members in Attendance: Dr. Pearlstein, Julie Swift and Zachary Nelson

- **Overall Goal**
    - **Produce an Audio Processing Integrated Circuit**
- I2S Interface
    - Audio Input: 2 channel stereo channel I2S (master interface)
    - Support audio input sample rates of 8kilosample/sec – 48kilosamples/sec
    - Output is the same sample rate as the input and I2S
    - Digital audio bit clock and the word select (ws) line will be controlled from master
    - Input and Output will be 2 channel 16 bits
- I2C Interface
    - Support single master configuration
    - 7-bit addressing and we will consume entire I2C address space
- Uses an external clock input
    - Clock frequency will be a minimum of 1200 times the audio sampling rate
    - Maximum clock rate will be 100 MHz
- Has an external reset pin
    - Power on reset
- Register Block
    - 512 bit frequency coefficients
    - 10k register bits
    - Register fields will include
        - Source select bit
            - Allows user to select between I2S and BIST (built in self test)
        - Filter Bypass Bit
            - 0 doesn't bypass, 1 bypass
        - 512 16-bit signed coefficients stored as 2's compliment
            - The effective radix point of the coefficients (4 bit number for data point, 4 bit number for coefficient)
        - Read only status register bits
            - Overflow/saturation detector – audio clipping
            - Input FIFO overrun
            - Output FIFO underrun
        - Control Bit Fields
            - 1 sticky bit to clear overrun
            - 1 sticky bit to clear underrun (stay until you clear them)
            - Clear Overflow Flag

            - Filter Order to Support
                - 9 bit number to represent 1 to 512
    - Presents an array of registers for hosts control and status monitoring (through I2C read and write operations).
- Provide built in self test function
    - Test gadgets for I2S
        - Test by converting I2S to audio coeffs
        - Audio coeffs to I2S
- Chips made by service call MOSIS
    - IBM7RF process, geometry, drawn gate length, 180nm gates, mixed signals,
    - Chip Area: no more than 3 by 3 squared mm
- Filter Audio
    - Implement an FIR filter on the input data based on the coefficients stored in the programmable registers
        - Programmable from 1-512 taps
        - Filter Order Control (9-bit number)
            - Support filter from 1-512 taps
        - Maintain intermediate precision of 4
- Produce a microcontroller platform to configure the audio input and output modules
    - UDA 1380
    - Produce a test fixture to show that the chip works
        - Sample analog audio and covert it to I2S
        - Receive I2S and convert it to analog audio
        - Allow user to create filter coeffs and upload them to the chip that was designed
        - Has to have software to configure the analog audio subsystem
        - Parametrize low pass filters, high pass, band pass, and comb filters
            - Use a button on a PSOC?
            - Could use slider on PSOC to change the frequency and upload the new coeffs to the chip
        - Create a board that we can plug the chip into to do testing
            - Bread board?
        - Will not have to design a printed circuit board for this chip

June 18th Senior Project Meeting

Armstrong 144, 3:30 P.M – 5:05 P.M.

Members in Attendance: Dr. Pearlstein, Zachary Nelson, Julie Swift, Whitley Forman and Dhruvit Naik

- Went through the CORE requirements hierarchy for the chip **(System Design Step)**
  - Zach will make the top-level requirements the elements from the chip block diagram so that the requirements are broken down in a more logical way
  - Zach will make an Enhanced Functional Flow Block Diagram (EFFBD) on CORE with the functions that were generated as part of creating the requirements
- Discussed the design flow of creating an integrated circuit
  - A figure illustrating this is on GitHub under Chip-Design/proj_asic/docs/design_flow.png
  - Major Steps
    - System Design
    - RTL Design
    - Logic Synthesis
    - Design for Test (DFT) Implementation – may be able to skip this
    - Floor Planning
    - Place and Optimization
    - Routing
    - Verification
- Deadline for the chip
  - November 30th: We would get the chip back in the Spring semester and be able to see if it works
  - March 2016: Would get the chip back after graduation and would implement our design on a FPGA during the Spring semester instead
- Dropbox will be used for storing private files
  - The rest of the code and documents will be stored on the public GitHub account
- Assigned Verilog modules to group members **(RTL Design Step)**
  - Everyone will start working on the modules over the summer
  - Zach: i2s_in.v and i2s_out.v
  - Julie: register.v
  - Whitley: i2c_slave.v
  - Dhruvit: filter.v
  - Kevin: chip.v
- Julie and Whitley need to hand in the NDA forms (electronically or in person)
- Kevin and Dhruvit need to accept the invite to the GitHub account

July 7th I2S Senior Project Meeting

Dr. Pearlstein's Office, 2:00 P.M – 5:10 P.M.

Members in Attendance: Dr. Pearlstein and Zachary Nelson

- Register block bits will either start with "ro", "rf", or "trig".
  - ro label for input.
  - rf label for outputs.
  - trig label when the bit causes something to trigger.
- Add the bit i2s_in_en to the I2S_IN block.
  - Low to high: coming out of reset.
  - High to low: goes back into a reset state.
- Need variables to characterize the BIST saw-tooth wave.
  - rf_bist_start_val (16 bit signed value) – start value.
  - rf_bist_inc (8 bit integer) – increment between 0 and 255.
  - rf_bist_upper_limit (16 bit signed value) – upper limit.
- We will create a synthesized sclk by dividing the system clock.
- Make all the flops clock with the system clock (always statements must be with clk).
- We will have a register holder that takes in one bit at a time.
- Add trig_i2sin_fifo_overrun_rst bit
  - Example:

```
if (ro_fifo_overrun)  → happens when rts=1 and rtr=0
        ro_fifo_overrun = 1
else if (trig_i2sin_fifo_overrun_rst)
        ro_fifo_overrun = 0
```

July 30th Senior Project Meeting

Dr. Pearlstein's Office, 2:00 P.M – 3:40 P.M.

Members in Attendance: Dr. Pearlstein and Zachary Nelson

- Discussed how register.v will use register addressing to access data.
- Discussed cell utilization and a rough approximation about how much area our chip will use (60% utilization)
- Went through **register_map.xlsx** with Dr. Pearlstein
    - Added new registers
    - Fixed register addressing
    - Added missing default values
- Added register fields that enable clipping for the filter and one that shifts the number of bit positions after the filter accumulator.
- Reviewed the block diagram of the **i2si.v** block.
    - This diagram seems to be correct and the block can start being designed and verification tests can be created.
- Discussed details of how the **i2so.v** will work.
    - Clock divider will go in this block
    - Zach wrote down a quick block diagram
- Block Documents Status
    - register.v – not started
    - i2si.v – near completion
    - i2so.v – not started
    - filter.v - not started
    - i2c.v - not started
- Zach and Kevin will switch roles.
    - Zach will be responsible for chip.v and can help with all blocks (especially i2s blocks)
    - Kevin will be responsible for the i2s blocks.

September 2nd Register Senior Project Meeting

Dr. Pearlstein's Office, 1:00-2:30

Members in Attendance: Dr. Pearlstein, Zachary Nelson, and Julie Swift

- Project Goals Slide
  - Why are we doing this project?
    - This is not a common project for undergraduates because of the high cost associated with the fabrication of a chip.
    - Describe the MOSIS Education Service
    - Free fabrication for 3mm by 3mm
  - Also refer to this as VLSI
  - Explain the difference between an FPGA and an Integrated Circuit
  - Why did we choose to process audio as our application?
    - Complex and interesting enough
    - Not too hard so that this project is able to be completed
  - Take out parameterized filters
  - Put a picture of a chip
- Chip Overview Slide
  - I2S is digital but represents an analog signal
- DropBox is for confidential files while GitHub is for everything else
- I2C Slide
  - sda_in and sda_out are both interfaces
- Filter Slide
  - Fix the summation sign
- DFT Slide
  - We will most likely not be doing this task but it is good to have as background information.
- Gate Level Simulation
  - We will not do gate level simulation that intensively.
- Place and Route
  - We will not manually do the place and route, the EDA tools will do it for us.
  - We may be doing manual floor planning.
- Create a Block Level Testbenches Slide
  - We will have tb for each individual block as well as the overall chip
- Discussed interfaces for the register.v module
- Discussed a block diagram of the register.v module.
  - Ask Julie or Zach for more specifics
- MOSIS information
  - We can order one lot of 40 chips
  - We will fab using the Global Foundries 180 nm CMOS (7HV) process
  - MOSIS technology code for the 7HV process is GF_7HV
  - Customer Submission date for 7HV is **March 7th, 2016**

September 2nd Senior Project Meeting

Armstrong Hall 137, 3:30 - 4:15 P.M.

Members in Attendance: All Members

- Add a presentation slide on input/output cells
- The Format of the weekly meeting will be:
    - Discuss what the team has done over the past week
    - Discuss what the team is going to do over the next week and beyond
    - The most important thing is to discuss any roadblocks that we have
- One goal is to have the design implemented on a FPGA by the end of the fall semester
- Some tasks that can take place during the spring semester:
    - Place and Route
    - Verification
- We need to adopt a project management tool and create a schedule that is more sophisticated than the one that Dr. Katz has provided us with
    - One potential tool that we will look into is Microsoft Project
    - Be specific enough for about 3 milestones per week
- Manage the amount of time being put into senior project
    - Put the needed amount of time into the project but if we find ourselves each putting much more than 6 hours/week, we may need to narrow the scope of the project.
- Grading for Senior Project
    - 2 students will be graded by Dr. Hernandez and 3 students will be graded by Dr. Pearlstein
    - The rubrics for Senior Project I and II was distributed by Dr. Katz in an email
- What FPGA will we use for the project?
    - We could use one that is already in stock
    - We could purchase one with the $500 that the group has
- Julie and Dhruvit will be responsible for EDA Installation
    - Step 1: Figure out the EDA requirements and install OS
    - Step 2: Install EDA
    - Step 3: Simple test case with libraries from MOSIS and doing a place and route test
- We need to discuss who will be responsible for configuring the test fixtures.
- We will hold off on the website design until we hear more information.

<u>For Next Week:</u>

- We need to create a detailed plan/schedule for the project using a tool like Microsoft Project
- Everybody needs to finish their block documents for their specific modules
- Everybody needs to add the information about their module to the CORE 9 project

September 9th Senior Project Meeting

Armstrong Hall 137 and 144: 4:00 – 5:15 P.M.

Members in Attendance: All Members

<u>Next Week's Work Plans:</u>

- **Zach**
  - Fix Schedule
  - RTR/RTS and XFC Protocol
  - Buying an FPGA and seeing if the school one will work
  - CORE for I2S Blocks
  - Start coding one of the I2S submodules (BIST Generator?)
- **Whitley**
  - Start coding the I2C module
  - CORE 9 for I2C Requirements
- **Julie**
  - EDA Tool Installation
  - Complete Register Block Documentation
  - Start coding the Register module
  - CORE 9 for Register Requirements
- **Kevin**
  - Complete one submodule for the I2S Block (Deserializer?)
  - Create test benches and test this submodule
  - CORE 9 for I2S Blocks
- **Dhruvit**
  - EDA Tool Installation
  - Start coding the Filter module
  - CORE 9 for the Filter module

<u>Meeting Notes:</u>

- The RTS/RTR and XFC protocols need to be in one place
  - Zach assigned as the owner to this
  - Should be finished by next Wednesday
- EDA Tools should be installed and a simple place and route test should be completed by next week
  - We will be downloading Mentor Tools
  - An action item is to contact by email or in person Mike about installing the tools
- Things that need to be added/changed for the Microsoft Project schedule
  - Creating test benches
  - Create test cases for individual blocks
  - Creating test cases
  - Correction: Microcontroller not "board" will be linked to the chip
  - Shorten the milestone names

    - Block level synthesis
    - Full chip simulation
- CORE requirements and use cases can be integrated to write test specifications
    - The new CORE license works
- Dr. Hernandez will give feedback on the schedule
- Dhruvit and Julie have access to 144-B
    - This is where we will be installing the EDA tools
    - Can we remotely connect to this computer-Ask Mike?
- Chip.v
    - Will instantiate everyone's modules
    - We will need to create input/output submodules
- Create model in another environment that takes in the register states and inputs and produces the correct output
    - An end to end test case
    - This is used to verify that we are producing the correct outputs in our chip
    - Does not need to be the most sophisticated model due to time
- Test benches will be stored on the "tb" folder on GitHub
- FIFO will be very similar for all blocks
    - The only difference will be the size and the width
- Need to buy crystal oscillators (10, 20, or 100?)
- Whitley will do board design in spring semester
- PSoC configuration will take 1-2 weeks of time
    - Whitley may be assigned to this task

September 23rd Senior Project Meeting

Armstrong Hall 137: 3:30-4:00 P.M.

Members in Attendance: Dr. Orlando Hernandez, Zachary Nelson, Julie Swift, Dhruvit Naik and Kevin Cao

Last Week's Work:

- Zach
  - Cleaned up RTS/RTR and XFC Protocols
  - Updated CORE 9 Files
  - Updated Schedule on Microsoft Project
    - Everyone agreed on due dates
    - Created a calendar on Google
  - Continued working on i2si_bist_gen.v
- Kevin
  - Continued working on i2si_deserializer.v
  - Looked into setting up the project website
- Dhruvit
  - Continued working on filter_convolution.v
- Julie
  - Continued working on register block document and register.v
- Whitley
  - Continued working on i2c_slave_deserializer.v

Next Week's Due Dates:

- Friday, September 25th
  - Install Linux on Machine in Room 144B (Dhruvit and Julie)
- Monday, September 28th
  - Finish Design and Testing: filter_convolution.v (Dhruvit)
  - Install EDA Tool on Machine in Room 144b (Dhruvit)
- Wednesday, September 30th
  - Finish Design and Testing: i2c_salve_deserializer.v (Whitley)
  - Finish Design and Testing: i2si_bist_gen.v (Zach)
  - Finish Design and Testing: i2si_deserializer (Kevin)

Meeting Notes:

- We should have a testing specification when we perform testing
  - Multiple smaller unit tests are better than a couple of large comprehensive tests for our blocks
- Dr. Hernandez offered to present his presentation slides on Verilog to the group
  - All of the Verilog modules will have 3 always statements
  - The group will look over the slides and let Dr. Hernandez know if this will be beneficial

September 30th Senior Project Meeting

Armstrong Hall 137: 3:30-3:30

Members in Attendance: All Members

Last Week's Work:

- Linux Installed
  - Permission errors
  - Need to email passwords
  - Root password: icchip2015
- Zach
  - Bist generator
  - Rts and rtr

Next Week's Due Dates:

- Last Week:
  - Finish Design and Testing of i2si_deserializer.v (Zach and Kevin)
    - 1 week for the deserializer.v
  - filter_convolution.v (Dhruvit)
    -
  - Testing of Whitley's block
    - 1 week for testing
- Friday, September 2nd
  - Create Web Page (Kevin)
  - Finish Design and Testing of filter_accumulator.v (Dhuvit)
- Wednesday, September 7th
  - Finish Design and Testing of i2so_serializer.v (Zach and Kevin)
  - Finish EDA Tool Installation (Dhruvit and Julie)
  - Place and Route Test (Dhruvit and Julie)
  - PDR Presentation #2 (All)

Meeting Notes:

- Zach will make a test fixture
- Action items
  - Investigate the use of mentor tools for place and route ()
- Asynchronous vs synchronous reset
  - We will use asynchronous
  - Add to rtr protocols

January 26th Senior Project Meeting

Armstrong Hall 148: 2:00PM – 2:45PM

Members in Attendance: All Members

Goals for Next Week:

- Kevin
  - Fix BIST issues
  - Testbenches with different clock ratios
  - Learn how to use Mentor Tools
- Dhruvit
  - Fix Filter Warnings
  - Mentor Tools
- Whitley
  - Register/I2C Subsystem
  - Subsystem Test
- Zach
  - I2S/Filter Subsystem
  - Register Help
  - FPGA
- Julie
  - Register/I2C Subsystem
  - Subsystem Test
- Ian
  - PSoC Program

February 2nd Senior Project Meeting

Armstrong Hall 148: 2:00PM – 3:00PM

Members in Attendance: All Members

Goals for Next Week:

- Updated Microsoft Project Plan
  - Plan 2.0
- Coordinate with Ian
- Whitley
  - Last Week
    - Got the submodule test finished
  - Next Week
    -
- Julie
  - Last Week
    - Got the submodule test finished
  - Next Week
    - Continue working on
- Kevin
  - Last Week
    - More testing on I2S
    - Looked into EDA tools
    - Matching tools with different stools
  - Next Week
    - Get libraries installed
    - Place and Route
    - Working with Dhruvit
- Zach
  - Last Week
    - Fixed issues
  - Next Week
    - Coordinate with Ian
    - Create New Plan
    - FPGA Implementation
- Dhruvit
  - Last Week
    - EDA tools are stables
  - Next Week
- Steps
  - Coding
  - Simulation/Verification

- Synthesis → netlist
- Physical Design
- Place and Route
- GSDII Masks File

OR you can start

- Manual schematic capture of IO cells

February 10th Senior Project Meeting

Armstrong Hall 148: 2:00PM – 3:00PM

Members in Attendance: All Members

- Zachary
  - Last Week
    - Created the presentation
    - Worked on the I2S/Filter Subsystem
    - Updated the top-level drawing
    - Created an updated schedule
    - Added all code to chip.v
      - All Verilog code is in chip.v and is synthesizable
      - Need help with a testbench
  - Next Week
    - Top-level testbench
    - Work on chip.v
- Whitley and Julie
  - Last Week
    - Register and I2C blocks working together
    - Loading register values works
  - Next Week
    - Top-level testbench
    - Implement FPGA for I2C
- Kevin
  - Last Week
    - Helped I2S/Filter module
    - EDA Tools tutorials
- Dhruvit
  - Last Week
    -
  - Next Week
    - More verification of filter block

March 1st Senior Project Meeting
Armstrong Hall 148: 2:00PM – 2:45PM

Members in Attendance: All members

- Zach
  - Last Week
    - Uploaded Ian's PSoC Program
    - Updated chip.v
    - Worked TCF poster
    - Updated schedule
    - Worked with PlanAhead Tutorial
    - Created document for final report
    - Realistic constraints form
    - Xilinx ISE will work
  - Next Week
    - Work with PlanAhead
    - Help finish Ian's I2S PSoC Program
    - Work on TCF Poster
  - Questions
    - When should we give you a TCF poster draft by?
      - Before break
    - What speed is FPGA? -3, -2, -1
      - Look in the data sheet
    - Can you help with PlanAhead?
      - Will meet with Dr. Pearlstein
    - How do we specify the I/O ports?
      - Will meet with Dr. Pearlstein
- Julie
  - Last Week
    - Looked into EDA tools from Princeton
    - Updated register Verilog code
    - Updated register Excel document
  - Next Week
    - Physical Design
- Dhruvit
  - Last Week
    - Filter model in Java
    - Fixed issues with filter Verilog code
  - Next Week
    - Physical Design

- Kevin
    - Last Week
        - Top-level simulation/verifications
        - I2S debugging
    - Next Week
        - Continue verification
- Whitley
    - Last Week
        - Created multiple test benches with different register values
    - Next Week
        - Continue verification
- Additional Comments
    - Source and sink could come from UDA 1300
    - Might need to buy this

March 8th Senior Project Meeting

Armstrong Hall 148: 2:00PM – 2:30PM

Members in Attendance: Zach, Dhruvit, Kevin, Whitley, Dr. Hernandez, and Dr. Pearlstein

- Zach
  - Last Week
    - Finished TCF Poster
    - Learned how to use Plan Ahead software
    - Set up clock signal and specified I/O ports
    - Fixed Realistic Constraints Form
  - This Week
    - Warnings for Xilinx ISE
    - Finish design in Plan Ahead
    - Get poster printed
- Dhruvit
  - Last Week
    - Top-Level Simulation
    - Looked at Physical Design presentation
  - This Week
    - Get a plan for the physical design
- Julie
  - Last Week
    - Looked at Physical Design presentation
  - This Week
    - Get a plan for the physical design
- Kevin
  - Last Week
    - Top-Level Simulation
    - Text file comparison
  - This Week
    - Install Model-Sim (Dhruvit might help)
    - Fix warnings
    - Clean up code
- Whitley
  - Last Week
    - Top-Level Simulation
  - This Week
    - I2C PSoC Program
    - Fix I2C always block
- Automate testing process using make file

March 22nd Senior Project Meeting

Armstrong Hall 148: 2:00PM – 3:00PM

Members in Attendance: Zach, Dhruvit, Kevin, Whitley, and Dr. Pearlstein

- Zach
  - Last Week
    - Fixed Abstract
    - Worked with I2S PSoC program
    - Fixed some RTL issues
  - This Week
    - Work with Ian on PSoC program
    - Keep updating top-level
    - Get clock verified on FPGA
    - Open drain and resistor for I2C
    - Output voltage on Artix board
    - DA conversion in PSoC
- Dhruvit
  - Last Week
    - Researched how to use EDA tools
    - Having trouble logging into machine while on campus
  - This Week
    - Continue working with physical design
- Kevin
  - Last Week
    - Downloaded ModelSIM
    - I2S block works without optimization
  - This Week
    - Continue fixing synthesis warnings
- Whitley
  - Last Week
    - Started I2C PSoC program
  - This Week
    - Continue I2C PSoC program
    - Test I2C on FPGA

March 29th Senior Project Meeting

Armstrong Hall 148: 2:00PM –

Members in Attendance: Zach, Dhruvit, Kevin, Whitley, and Dr. Pearlstein

- Zach
  - Last Week
    - Got all correct port mappings
    - Fixed all but one FPGA warnings
    - Ian finished PSoC program and show me how to use it
      - I tested it and it seems to work
    - Wrote FPGA guide
    - Created a realistic test environment with I2S input stream
      - Nothing on I2S output (might be a reset issue)
    - Updated website
  - This Week
    - Further testing and debugging of FPGA
    - More code documentation
- Dhruvit
  - Last Week
    - Drew a transistor in EDA tools
  - This Week
    - Design rule check
    - Research on how to make inductor
    - Power and ground routing
    - MOSIS GSDII file requirements
    - Implementation warning
- Kevin
  - Last Week
    - Fixed warnings
  - This Week
    - Help Dhruvit
- Whitley
  - Last Week
    - Fixed I2C warnings
    - PSoC I2C program
  - This Week
    - I2C program
    - Default register state
- Julie
  - Last Week
    - Playing around and learning the EDA tool
    - Fixed overrun/underrun setting that Kevin told me to correct

        - Added comments to code
    - This Week
        - Help Dhruvit

April 5th Senior Project Meeting

Armstrong Hall 148: 2:00PM – 3:00PM

Members in Attendance: Zach, Dhruvit, Kevin, Whitley, Dr. Pearlstein, and Dr. Hernandez

- Zach
  - Last Week
    - Set up final report and presentation
    - Added comments to code
    - Fixed chip output
    - Passed sine, triangular, and square wave through FPGA
    - Started a testing document for FPGA
  - This Week
    - Work on reset warning
    - Further testing of FPGA (maybe with filtering and I2C)
  - Questions
    - VDDIO on PSoC? → Should be fixed now
- Dhruvit
  - Last Week
    - Made some changes to register.v, filter_mux.v, filter_round_truncate.v
    - Created a new testbench that does not use I2C
    - Worked with EDA tools
    - Worked on FPGA warning
  - This Week
    - Create different filter coefficient combinations to test on FPGA
    - Continue working with physical design of a chip
- Kevin
  - Last Week
    - Testing of chip.v
    - Worked with EDA tools
  - This Week
    - Continue working with physical design of a chip
- Whitley
  - Last Week
    - Worked on I2C PSoC program
  - This Week
    - Finish I2C PSoC program
    - Test I2C PSoC program on FPGA
- Julie
  - Last Week
    - Worked with EDA tools
  - This Week
    - Continue working with physical design of a chip

April 12th Senior Project Update

Dhrvuit:

- What I've Done Last Week
  - Get filter coefficients for a lowpass, highpass, bandpass (finished)
  - Look into mentor design checks and rules (working on it)
  - Get model data (next task)
- Goals for Upcoming Week
  - Get model data
  - Document Verilog code and Java model
- Blockers
  - None

Julie:

- Met with Dhruvit and reviewed progress and milestones that need to be accomplished in the EDA tool.
- Research towards the development of the resistor design using Pyxis layout and played around with the tools.

Kevin:

- What I've Done Last Week
  - Looked at tutorials of how to create a resistor in layout.
  - Began designing the resistor in layout
- Goals for Upcoming Week
  - Continue working on resistor layout
  - Determine types of layers needed
  - Verify resistor design
  - Create GDSII file
- Blockers
  - Found no tutorials for Mentor, only found tutorials for Cadence
  - Unsure of what layers to use for the resistor

Whitley:

- No report received

Zach:

- Goals for Past Week
  - Figure out what the "reset" warning means and if we can ignore it.
  - Further testing of the FPGA (maybe with filtering).
- What I Accomplished
  - I looked into the "reset" warning in more depth. First, I tried removing the reset signal from filter_round_truncate.v like Dhruvit previously did. I also found that the warning jumped to other modules. Next, I found a report online that claims this warning occurs when using an asynchronous reset and can be ignored. The report

states “The above warning cites that while an RST signal was generated by the DCM, it does not drive clock pins such as flip-flop operations. This is because an asynchronous reset is used in the circuitry, and this does not pose any problem”. Next, I decided to test the reset signal on the FPGA. The reset signal is now defined from the red reset button on the FPGA. When the button is pressed, all of the outputs turn to zero. This means that the reset signal is reaching the submodule blocks. Overall, this makes me agree with the report I found and think that we can ignore the warning.
    - I recorded and verified the outputs of the FPGA test (passing audio stream through) in a FPGA Testing document. This document can be found at project_asic/fpga/FPGA Testing.docx.
    - Filled in the Appendix sections of the final report.
    - Updated the website.
- Goals for Upcoming Week
    - Work with Whitley to get the I2C PSoC program interfaced with the FPGA.
    - Work with Whitley to test the I2C PSoC program.
    - Test the FPGA with filtering enabled
        - Using I2C if the PSoC program is working
        - If not, I will create separate projects with different default filter coefficient values.
- Blockers
    - Dhruvit needs to give me filter coefficient values for a couple different types of filters.
    - Dhruvit also needs to use his filter model to find what the output audio signal should be for the input audio signals that I sent him.

April 19th Senior Project Meeting

Armstrong Hall 148: 2:00PM –2:30PM

Members in Attendance: Zach, Dhruvit, Kevin, Whitley, Julie, and Dr. Pearlstein

- Zach
  - Last Week
    - Worked with Whitley on testing/debugging I2C on FPGA
    - Tried testing filtering on FPGA but ran into issues
  - This Week
    - Meet with Dhruvit to finish testing filter
    - Meet with Whitley to continue debugging I2C
  - Notes
    - Make center frequency gain low
    - Fft will be constant for white noise
- Dhruvit
  - Last Week
    - Got signal outputs for a specific filter
    - Worked on modifying filter code
    - Worked with Mentor
  - This Week
    - Meet with Zach to finish testing filter
- Kevin
  - Last Week
  - This Week
    - Continue working with Mentor
- Whitley
  - Last Week
    - Worked on improving PSoC program
    - Met with Zach to test/debug I2C on FPGA
  - This Week
    - Meet with Zach to continue debugging I2C
- Julie
  - Last Week
    - Worked on report
  - This Week

Clean up code and continue working on report

List of Contacts:

- Ann Zsilavetz - Computer Science Program Assistant (zsilave2@tcnj.edu)
- Chris Collins - Lab Technician (collinsc@tcnj.edu)
- Katie Thacker - CORE University Program Coordinator (kthacker@vitechcorp.com)
- Michael Mensch – Information Technology (mensch@tcnj.edu)

Safety Considerations:

The major safety concern with this project involved interfacing the microcontroller with the FPGA. By taking the following precautions, no events compromising the safety of the team occurred. Since both the FPGA and microcontroller used 3.3V logic, we did not have to worry about any problems with high voltage. We made sure there were no loose wires by keeping our wiring as organized as possible. When working with the FPGA, we did not allow any food or drinks around the board in order to prevent damage. We were also aware of the fact that either the microcontroller or FPGA could start smoking. These chemicals could potentially be dangerous and we were aware to contact Chris Collins if this ever occurred.

Material List:

Software Materials:

- CORE 9 University
- Dropbox
- Git/GitHub
    - URL: https://github.com/SeniorProject-2016/Chip-Design
- Red Hat® Enterprise Linux® Operating System
- Microsoft Project 2013
- Mentor Graphics Software
- ISE Design Suite 14.7

Hardware Materials:

- CY8CKIT-050 PSoC 5LP Development Kit
- Nexys 4 Artix-7 FPGA Board
- Stock Room Materials (wires, wire cutters, etc.)
- TI PCM9211 Digital Audio Transceiver and ADC

Financial Budget:

Item	Cost	Comment
CORE 9 University	$0.00	Vitech's CORE in the Classroom program
Dropbox	$0.00	Free up to 2GB of storage
Git/GitHub	$0.00	Free software
Linux RedHat Operating System	$0.00	TCNJ Subscription
Microsoft Project 2013	$0.00	Free from the Computer Science Department's DreamSpark Program
Mentor Graphics Software	$0.00	TCNJ Subscription
ISE Design Suite 14.7	$0.00	Free software
CY8CKIT-050 PSoC 5LP Development Kit	$0.00	School already owns
TI PCM9211 Digital Audio Transceiver and ADC	$72.87	Purchased from Digilent
Nexys 4 Artix-7 FPGA Board	$192.41	Purchased from Digilent
Total Budget	$500.00	
Total Costs	$265.28	
End Balance	$234.72	Well under-budget

Appendix C: Source Code

chip.v (top-level module) 130
Filter Code 166
- filter.v 166
- filter_accumulator.v 192
- filter_mux.v 193
- filter_stm.v 210
- filter_storage.v 214
- filter_round_truncate.v 222

I2C Code 224
- i2c.v 224
- i2c_deserializer.v 225
- i2c_sequencer.v 233
- i2c_serializer.v 235

I2S Input Code 238
- fifo.v 238
- i2s_in.v 240
- i2si_bist_gen.v 243
- i2si_deserializer.v 245
- i2si_mux.v 248
- i2si_synchronizer.v 249

I2S Output Code 251
- i2s_out.v 251
- i2so_serializer.v 253

Register Code 256
- chip_reg.v 256
- register.v 280
- trig_generator.v 340

PSoC Code 341
- I2C 341
- I2S 343

chip.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             chip.v
// Create Date:             12/20/2015
// Last Modification:       2/14/2016
// Author:                  Zachary Nelson
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module chip(clk, rst_n,                                          // General
            i2si_sck, i2si_ws, i2si_sd,                          // I2S Input
            i2so_sck, i2so_ws, i2so_sd,                          // I2S Output
            i2c_addr_bits, i2c_scl, i2c_sda_in, i2c_sda_od);     // I2C

    // CHIP INTERFACES
    //-----------------------------------------------------------------------
    //-----------------------------------------------------------------------
    // General
    input clk;                          // master clock
    input rst_n;                        // reset not

    // I2S Input
    input i2si_sck;                     // I2S input serial clock
    input i2si_ws;                      // I2S input word select
    input i2si_sd;                      // I2S input serial data

    // I2S Output
    output i2so_sck;                    // I2S output serial clock
    output i2so_ws;                     // I2S output word select
    output i2so_sd;                     // I2S output serial data

    // I2C
    input i2c_addr_bits;                // 3 LSB I2C address select
    input i2c_scl;                      // serial clock
    input i2c_sda_in;                   // serial data
    output i2c_sda_od;
    //-----------------------------------------------------------------------
    //-----------------------------------------------------------------------

    assign i2c_sda_od = i2c_sda_out ? 0:1'bZ;

    // BLOCK CONNECTIONS
    //-----------------------------------------------------------------------
    //-----------------------------------------------------------------------
    // Inputs to I2S Input Block
    wire rf_i2si_en;                        // enable bit for deserializer
    wire [11:0] rf_bist_start_val;          // BIST start value
    wire [7:0] rf_bist_inc;                 // BIST increment value
    wire [11:0] rf_bist_up_limit;           // BIST upper limit
    wire rf_mux_en;                         // multiplexer select bit
    wire i2si_rtr;                          // I2S ready to receive
    wire trig_fifo_overrun_clr;             // signal to reset ro_fifo_overrun

    // Inputs to I2S Output Block
    wire i2si_sync_sck;                     // synchronized serial clock
    wire i2si_sck_transition;               // synchronized serial clock transition
    wire [31:0] filt_out_data;              // I2SO input data
    wire trig_fifo_underrun;                // signal to reset ro_fifo_underrun

    // Inputs to Filter Block
    wire filt_rts;                          // I2S input ready to send
    wire [31:0] filt_input_data;            // I2SI output data
    wire filt_rtr;
    wire [2:0] rf_filter_shift;             // # of bits to shift after accumulator
    wire rf_filter_clip_en;                 // select bit (1- perform clipping, 0- no clipping)
    wire [7:0] rf_filter_coeff0_a, rf_filter_coeff0_b,rf_filter_coeff1_a,
rf_filter_coeff1_b,rf_filter_coeff2_a, rf_filter_coeff2_b,rf_filter_coeff3_a,
```

```
rf_filter_coeff3_b,rf_filter_coeff4_a, rf_filter_coeff4_b,rf_filter_coeff5_a,
rf_filter_coeff5_b,rf_filter_coeff6_a, rf_filter_coeff6_b,rf_filter_coeff7_a,
rf_filter_coeff7_b,rf_filter_coeff8_a, rf_filter_coeff8_b,rf_filter_coeff9_a,
rf_filter_coeff9_b,rf_filter_coeff10_a, rf_filter_coeff10_b,rf_filter_coeff11_a,
rf_filter_coeff11_b,rf_filter_coeff12_a, rf_filter_coeff12_b,rf_filter_coeff13_a,
rf_filter_coeff13_b,rf_filter_coeff14_a, rf_filter_coeff14_b,rf_filter_coeff15_a,
rf_filter_coeff15_b,rf_filter_coeff16_a, rf_filter_coeff16_b,rf_filter_coeff17_a,
rf_filter_coeff17_b,rf_filter_coeff18_a, rf_filter_coeff18_b,rf_filter_coeff19_a,
rf_filter_coeff19_b,rf_filter_coeff20_a, rf_filter_coeff20_b,rf_filter_coeff21_a,
rf_filter_coeff21_b,rf_filter_coeff22_a, rf_filter_coeff22_b,rf_filter_coeff23_a,
rf_filter_coeff23_b,rf_filter_coeff24_a, rf_filter_coeff24_b,rf_filter_coeff25_a,
rf_filter_coeff25_b,rf_filter_coeff26_a, rf_filter_coeff26_b,rf_filter_coeff27_a,
rf_filter_coeff27_b,rf_filter_coeff28_a, rf_filter_coeff28_b,rf_filter_coeff29_a,
rf_filter_coeff29_b,rf_filter_coeff30_a, rf_filter_coeff30_b,rf_filter_coeff31_a,
rf_filter_coeff31_b,rf_filter_coeff32_a, rf_filter_coeff32_b,rf_filter_coeff33_a,
rf_filter_coeff33_b,rf_filter_coeff34_a, rf_filter_coeff34_b,rf_filter_coeff35_a,
rf_filter_coeff35_b,rf_filter_coeff36_a, rf_filter_coeff36_b,rf_filter_coeff37_a,
rf_filter_coeff37_b,rf_filter_coeff38_a, rf_filter_coeff38_b,rf_filter_coeff39_a,
rf_filter_coeff39_b,rf_filter_coeff40_a, rf_filter_coeff40_b,rf_filter_coeff41_a,
rf_filter_coeff41_b,rf_filter_coeff42_a, rf_filter_coeff42_b,rf_filter_coeff43_a,
rf_filter_coeff43_b,rf_filter_coeff44_a, rf_filter_coeff44_b,rf_filter_coeff45_a,
rf_filter_coeff45_b,rf_filter_coeff46_a, rf_filter_coeff46_b,rf_filter_coeff47_a,
rf_filter_coeff47_b,rf_filter_coeff48_a, rf_filter_coeff48_b,rf_filter_coeff49_a,
rf_filter_coeff49_b,rf_filter_coeff50_a, rf_filter_coeff50_b,rf_filter_coeff51_a,
rf_filter_coeff51_b,rf_filter_coeff52_a, rf_filter_coeff52_b,rf_filter_coeff53_a,
rf_filter_coeff53_b,rf_filter_coeff54_a, rf_filter_coeff54_b,rf_filter_coeff55_a,
rf_filter_coeff55_b,rf_filter_coeff56_a, rf_filter_coeff56_b,rf_filter_coeff57_a,
rf_filter_coeff57_b,rf_filter_coeff58_a, rf_filter_coeff58_b,rf_filter_coeff59_a,
rf_filter_coeff59_b,rf_filter_coeff60_a, rf_filter_coeff60_b,rf_filter_coeff61_a,
rf_filter_coeff61_b,rf_filter_coeff62_a, rf_filter_coeff62_b,rf_filter_coeff63_a,
rf_filter_coeff63_b,rf_filter_coeff64_a, rf_filter_coeff64_b,rf_filter_coeff65_a,
rf_filter_coeff65_b,rf_filter_coeff66_a, rf_filter_coeff66_b,rf_filter_coeff67_a,
rf_filter_coeff67_b,rf_filter_coeff68_a, rf_filter_coeff68_b,rf_filter_coeff69_a,
rf_filter_coeff69_b,rf_filter_coeff70_a, rf_filter_coeff70_b,rf_filter_coeff71_a,
rf_filter_coeff71_b,rf_filter_coeff72_a, rf_filter_coeff72_b,rf_filter_coeff73_a,
rf_filter_coeff73_b,rf_filter_coeff74_a, rf_filter_coeff74_b,rf_filter_coeff75_a,
rf_filter_coeff75_b,rf_filter_coeff76_a, rf_filter_coeff76_b,rf_filter_coeff77_a,
rf_filter_coeff77_b,rf_filter_coeff78_a, rf_filter_coeff78_b,rf_filter_coeff79_a,
rf_filter_coeff79_b,rf_filter_coeff80_a, rf_filter_coeff80_b,rf_filter_coeff81_a,
rf_filter_coeff81_b,rf_filter_coeff82_a, rf_filter_coeff82_b,rf_filter_coeff83_a,
rf_filter_coeff83_b,rf_filter_coeff84_a, rf_filter_coeff84_b,rf_filter_coeff85_a,
rf_filter_coeff85_b,rf_filter_coeff86_a, rf_filter_coeff86_b,rf_filter_coeff87_a,
rf_filter_coeff87_b,rf_filter_coeff88_a, rf_filter_coeff88_b,rf_filter_coeff89_a,
rf_filter_coeff89_b,rf_filter_coeff90_a, rf_filter_coeff90_b,rf_filter_coeff91_a,
rf_filter_coeff91_b,rf_filter_coeff92_a, rf_filter_coeff92_b,rf_filter_coeff93_a,
rf_filter_coeff93_b,rf_filter_coeff94_a, rf_filter_coeff94_b,rf_filter_coeff95_a,
rf_filter_coeff95_b,rf_filter_coeff96_a, rf_filter_coeff96_b,rf_filter_coeff97_a,
rf_filter_coeff97_b,rf_filter_coeff98_a, rf_filter_coeff98_b,rf_filter_coeff99_a,
rf_filter_coeff99_b,rf_filter_coeff100_a, rf_filter_coeff100_b,rf_filter_coeff101_a,
rf_filter_coeff101_b,rf_filter_coeff102_a, rf_filter_coeff102_b,rf_filter_coeff103_a,
rf_filter_coeff103_b,rf_filter_coeff104_a, rf_filter_coeff104_b,rf_filter_coeff105_a,
rf_filter_coeff105_b,rf_filter_coeff106_a, rf_filter_coeff106_b,rf_filter_coeff107_a,
rf_filter_coeff107_b,rf_filter_coeff108_a, rf_filter_coeff108_b,rf_filter_coeff109_a,
rf_filter_coeff109_b,rf_filter_coeff110_a, rf_filter_coeff110_b,rf_filter_coeff111_a,
rf_filter_coeff111_b,rf_filter_coeff112_a, rf_filter_coeff112_b,rf_filter_coeff113_a,
rf_filter_coeff113_b,rf_filter_coeff114_a, rf_filter_coeff114_b,rf_filter_coeff115_a,
rf_filter_coeff115_b,rf_filter_coeff116_a, rf_filter_coeff116_b,rf_filter_coeff117_a,
rf_filter_coeff117_b,rf_filter_coeff118_a, rf_filter_coeff118_b,rf_filter_coeff119_a,
rf_filter_coeff119_b,rf_filter_coeff120_a, rf_filter_coeff120_b,rf_filter_coeff121_a,
rf_filter_coeff121_b,rf_filter_coeff122_a, rf_filter_coeff122_b,rf_filter_coeff123_a,
rf_filter_coeff123_b,rf_filter_coeff124_a, rf_filter_coeff124_b,rf_filter_coeff125_a,
rf_filter_coeff125_b,rf_filter_coeff126_a, rf_filter_coeff126_b,rf_filter_coeff127_a,
rf_filter_coeff127_b,rf_filter_coeff128_a, rf_filter_coeff128_b,rf_filter_coeff129_a,
rf_filter_coeff129_b,rf_filter_coeff130_a, rf_filter_coeff130_b,rf_filter_coeff131_a,
rf_filter_coeff131_b,rf_filter_coeff132_a, rf_filter_coeff132_b,rf_filter_coeff133_a,
rf_filter_coeff133_b,rf_filter_coeff134_a, rf_filter_coeff134_b,rf_filter_coeff135_a,
rf_filter_coeff135_b,rf_filter_coeff136_a, rf_filter_coeff136_b,rf_filter_coeff137_a,
rf_filter_coeff137_b,rf_filter_coeff138_a, rf_filter_coeff138_b,rf_filter_coeff139_a,
rf_filter_coeff139_b,rf_filter_coeff140_a, rf_filter_coeff140_b,rf_filter_coeff141_a,
rf_filter_coeff141_b,rf_filter_coeff142_a, rf_filter_coeff142_b,rf_filter_coeff143_a,
rf_filter_coeff143_b,rf_filter_coeff144_a, rf_filter_coeff144_b,rf_filter_coeff145_a,
```

rf_filter_coeff145_b,rf_filter_coeff146_a, rf_filter_coeff146_b,rf_filter_coeff147_a,
rf_filter_coeff147_b,rf_filter_coeff148_a, rf_filter_coeff148_b,rf_filter_coeff149_a,
rf_filter_coeff149_b,rf_filter_coeff150_a, rf_filter_coeff150_b,rf_filter_coeff151_a,
rf_filter_coeff151_b,rf_filter_coeff152_a, rf_filter_coeff152_b,rf_filter_coeff153_a,
rf_filter_coeff153_b,rf_filter_coeff154_a, rf_filter_coeff154_b,rf_filter_coeff155_a,
rf_filter_coeff155_b,rf_filter_coeff156_a, rf_filter_coeff156_b,rf_filter_coeff157_a,
rf_filter_coeff157_b,rf_filter_coeff158_a, rf_filter_coeff158_b,rf_filter_coeff159_a,
rf_filter_coeff159_b,rf_filter_coeff160_a, rf_filter_coeff160_b,rf_filter_coeff161_a,
rf_filter_coeff161_b,rf_filter_coeff162_a, rf_filter_coeff162_b,rf_filter_coeff163_a,
rf_filter_coeff163_b,rf_filter_coeff164_a, rf_filter_coeff164_b,rf_filter_coeff165_a,
rf_filter_coeff165_b,rf_filter_coeff166_a, rf_filter_coeff166_b,rf_filter_coeff167_a,
rf_filter_coeff167_b,rf_filter_coeff168_a, rf_filter_coeff168_b,rf_filter_coeff169_a,
rf_filter_coeff169_b,rf_filter_coeff170_a, rf_filter_coeff170_b,rf_filter_coeff171_a,
rf_filter_coeff171_b,rf_filter_coeff172_a, rf_filter_coeff172_b,rf_filter_coeff173_a,
rf_filter_coeff173_b,rf_filter_coeff174_a, rf_filter_coeff174_b,rf_filter_coeff175_a,
rf_filter_coeff175_b,rf_filter_coeff176_a, rf_filter_coeff176_b,rf_filter_coeff177_a,
rf_filter_coeff177_b,rf_filter_coeff178_a, rf_filter_coeff178_b,rf_filter_coeff179_a,
rf_filter_coeff179_b,rf_filter_coeff180_a, rf_filter_coeff180_b,rf_filter_coeff181_a,
rf_filter_coeff181_b,rf_filter_coeff182_a, rf_filter_coeff182_b,rf_filter_coeff183_a,
rf_filter_coeff183_b,rf_filter_coeff184_a, rf_filter_coeff184_b,rf_filter_coeff185_a,
rf_filter_coeff185_b,rf_filter_coeff186_a, rf_filter_coeff186_b,rf_filter_coeff187_a,
rf_filter_coeff187_b,rf_filter_coeff188_a, rf_filter_coeff188_b,rf_filter_coeff189_a,
rf_filter_coeff189_b,rf_filter_coeff190_a, rf_filter_coeff190_b,rf_filter_coeff191_a,
rf_filter_coeff191_b,rf_filter_coeff192_a, rf_filter_coeff192_b,rf_filter_coeff193_a,
rf_filter_coeff193_b,rf_filter_coeff194_a, rf_filter_coeff194_b,rf_filter_coeff195_a,
rf_filter_coeff195_b,rf_filter_coeff196_a, rf_filter_coeff196_b,rf_filter_coeff197_a,
rf_filter_coeff197_b,rf_filter_coeff198_a, rf_filter_coeff198_b,rf_filter_coeff199_a,
rf_filter_coeff199_b,rf_filter_coeff200_a, rf_filter_coeff200_b,rf_filter_coeff201_a,
rf_filter_coeff201_b,rf_filter_coeff202_a, rf_filter_coeff202_b,rf_filter_coeff203_a,
rf_filter_coeff203_b,rf_filter_coeff204_a, rf_filter_coeff204_b,rf_filter_coeff205_a,
rf_filter_coeff205_b,rf_filter_coeff206_a, rf_filter_coeff206_b,rf_filter_coeff207_a,
rf_filter_coeff207_b,rf_filter_coeff208_a, rf_filter_coeff208_b,rf_filter_coeff209_a,
rf_filter_coeff209_b,rf_filter_coeff210_a, rf_filter_coeff210_b,rf_filter_coeff211_a,
rf_filter_coeff211_b,rf_filter_coeff212_a, rf_filter_coeff212_b,rf_filter_coeff213_a,
rf_filter_coeff213_b,rf_filter_coeff214_a, rf_filter_coeff214_b,rf_filter_coeff215_a,
rf_filter_coeff215_b,rf_filter_coeff216_a, rf_filter_coeff216_b,rf_filter_coeff217_a,
rf_filter_coeff217_b,rf_filter_coeff218_a, rf_filter_coeff218_b,rf_filter_coeff219_a,
rf_filter_coeff219_b,rf_filter_coeff220_a, rf_filter_coeff220_b,rf_filter_coeff221_a,
rf_filter_coeff221_b,rf_filter_coeff222_a, rf_filter_coeff222_b,rf_filter_coeff223_a,
rf_filter_coeff223_b,rf_filter_coeff224_a, rf_filter_coeff224_b,rf_filter_coeff225_a,
rf_filter_coeff225_b,rf_filter_coeff226_a, rf_filter_coeff226_b,rf_filter_coeff227_a,
rf_filter_coeff227_b,rf_filter_coeff228_a, rf_filter_coeff228_b,rf_filter_coeff229_a,
rf_filter_coeff229_b,rf_filter_coeff230_a, rf_filter_coeff230_b,rf_filter_coeff231_a,
rf_filter_coeff231_b,rf_filter_coeff232_a, rf_filter_coeff232_b,rf_filter_coeff233_a,
rf_filter_coeff233_b,rf_filter_coeff234_a, rf_filter_coeff234_b,rf_filter_coeff235_a,
rf_filter_coeff235_b,rf_filter_coeff236_a, rf_filter_coeff236_b,rf_filter_coeff237_a,
rf_filter_coeff237_b,rf_filter_coeff238_a, rf_filter_coeff238_b,rf_filter_coeff239_a,
rf_filter_coeff239_b,rf_filter_coeff240_a, rf_filter_coeff240_b,rf_filter_coeff241_a,
rf_filter_coeff241_b,rf_filter_coeff242_a, rf_filter_coeff242_b,rf_filter_coeff243_a,
rf_filter_coeff243_b,rf_filter_coeff244_a, rf_filter_coeff244_b,rf_filter_coeff245_a,
rf_filter_coeff245_b,rf_filter_coeff246_a, rf_filter_coeff246_b,rf_filter_coeff247_a,
rf_filter_coeff247_b,rf_filter_coeff248_a, rf_filter_coeff248_b,rf_filter_coeff249_a,
rf_filter_coeff249_b,rf_filter_coeff250_a, rf_filter_coeff250_b,rf_filter_coeff251_a,
rf_filter_coeff251_b,rf_filter_coeff252_a, rf_filter_coeff252_b,rf_filter_coeff253_a,
rf_filter_coeff253_b,rf_filter_coeff254_a, rf_filter_coeff254_b,rf_filter_coeff255_a,
rf_filter_coeff255_b,rf_filter_coeff256_a, rf_filter_coeff256_b,rf_filter_coeff257_a,
rf_filter_coeff257_b,rf_filter_coeff258_a, rf_filter_coeff258_b,rf_filter_coeff259_a,
rf_filter_coeff259_b,rf_filter_coeff260_a, rf_filter_coeff260_b,rf_filter_coeff261_a,
rf_filter_coeff261_b,rf_filter_coeff262_a, rf_filter_coeff262_b,rf_filter_coeff263_a,
rf_filter_coeff263_b,rf_filter_coeff264_a, rf_filter_coeff264_b,rf_filter_coeff265_a,
rf_filter_coeff265_b,rf_filter_coeff266_a, rf_filter_coeff266_b,rf_filter_coeff267_a,
rf_filter_coeff267_b,rf_filter_coeff268_a, rf_filter_coeff268_b,rf_filter_coeff269_a,
rf_filter_coeff269_b,rf_filter_coeff270_a, rf_filter_coeff270_b,rf_filter_coeff271_a,
rf_filter_coeff271_b,rf_filter_coeff272_a, rf_filter_coeff272_b,rf_filter_coeff273_a,
rf_filter_coeff273_b,rf_filter_coeff274_a, rf_filter_coeff274_b,rf_filter_coeff275_a,
rf_filter_coeff275_b,rf_filter_coeff276_a, rf_filter_coeff276_b,rf_filter_coeff277_a,
rf_filter_coeff277_b,rf_filter_coeff278_a, rf_filter_coeff278_b,rf_filter_coeff279_a,
rf_filter_coeff279_b,rf_filter_coeff280_a, rf_filter_coeff280_b,rf_filter_coeff281_a,
rf_filter_coeff281_b,rf_filter_coeff282_a, rf_filter_coeff282_b,rf_filter_coeff283_a,
rf_filter_coeff283_b,rf_filter_coeff284_a, rf_filter_coeff284_b,rf_filter_coeff285_a,
rf_filter_coeff285_b,rf_filter_coeff286_a, rf_filter_coeff286_b,rf_filter_coeff287_a,

rf_filter_coeff287_b,rf_filter_coeff288_a, rf_filter_coeff288_b,rf_filter_coeff289_a,
rf_filter_coeff289_b,rf_filter_coeff290_a, rf_filter_coeff290_b,rf_filter_coeff291_a,
rf_filter_coeff291_b,rf_filter_coeff292_a, rf_filter_coeff292_b,rf_filter_coeff293_a,
rf_filter_coeff293_b,rf_filter_coeff294_a, rf_filter_coeff294_b,rf_filter_coeff295_a,
rf_filter_coeff295_b,rf_filter_coeff296_a, rf_filter_coeff296_b,rf_filter_coeff297_a,
rf_filter_coeff297_b,rf_filter_coeff298_a, rf_filter_coeff298_b,rf_filter_coeff299_a,
rf_filter_coeff299_b,rf_filter_coeff300_a, rf_filter_coeff300_b,rf_filter_coeff301_a,
rf_filter_coeff301_b,rf_filter_coeff302_a, rf_filter_coeff302_b,rf_filter_coeff303_a,
rf_filter_coeff303_b,rf_filter_coeff304_a, rf_filter_coeff304_b,rf_filter_coeff305_a,
rf_filter_coeff305_b,rf_filter_coeff306_a, rf_filter_coeff306_b,rf_filter_coeff307_a,
rf_filter_coeff307_b,rf_filter_coeff308_a, rf_filter_coeff308_b,rf_filter_coeff309_a,
rf_filter_coeff309_b,rf_filter_coeff310_a, rf_filter_coeff310_b,rf_filter_coeff311_a,
rf_filter_coeff311_b,rf_filter_coeff312_a, rf_filter_coeff312_b,rf_filter_coeff313_a,
rf_filter_coeff313_b,rf_filter_coeff314_a, rf_filter_coeff314_b,rf_filter_coeff315_a,
rf_filter_coeff315_b,rf_filter_coeff316_a, rf_filter_coeff316_b,rf_filter_coeff317_a,
rf_filter_coeff317_b,rf_filter_coeff318_a, rf_filter_coeff318_b,rf_filter_coeff319_a,
rf_filter_coeff319_b,rf_filter_coeff320_a, rf_filter_coeff320_b,rf_filter_coeff321_a,
rf_filter_coeff321_b,rf_filter_coeff322_a, rf_filter_coeff322_b,rf_filter_coeff323_a,
rf_filter_coeff323_b,rf_filter_coeff324_a, rf_filter_coeff324_b,rf_filter_coeff325_a,
rf_filter_coeff325_b,rf_filter_coeff326_a, rf_filter_coeff326_b,rf_filter_coeff327_a,
rf_filter_coeff327_b,rf_filter_coeff328_a, rf_filter_coeff328_b,rf_filter_coeff329_a,
rf_filter_coeff329_b,rf_filter_coeff330_a, rf_filter_coeff330_b,rf_filter_coeff331_a,
rf_filter_coeff331_b,rf_filter_coeff332_a, rf_filter_coeff332_b,rf_filter_coeff333_a,
rf_filter_coeff333_b,rf_filter_coeff334_a, rf_filter_coeff334_b,rf_filter_coeff335_a,
rf_filter_coeff335_b,rf_filter_coeff336_a, rf_filter_coeff336_b,rf_filter_coeff337_a,
rf_filter_coeff337_b,rf_filter_coeff338_a, rf_filter_coeff338_b,rf_filter_coeff339_a,
rf_filter_coeff339_b,rf_filter_coeff340_a, rf_filter_coeff340_b,rf_filter_coeff341_a,
rf_filter_coeff341_b,rf_filter_coeff342_a, rf_filter_coeff342_b,rf_filter_coeff343_a,
rf_filter_coeff343_b,rf_filter_coeff344_a, rf_filter_coeff344_b,rf_filter_coeff345_a,
rf_filter_coeff345_b,rf_filter_coeff346_a, rf_filter_coeff346_b,rf_filter_coeff347_a,
rf_filter_coeff347_b,rf_filter_coeff348_a, rf_filter_coeff348_b,rf_filter_coeff349_a,
rf_filter_coeff349_b,rf_filter_coeff350_a, rf_filter_coeff350_b,rf_filter_coeff351_a,
rf_filter_coeff351_b,rf_filter_coeff352_a, rf_filter_coeff352_b,rf_filter_coeff353_a,
rf_filter_coeff353_b,rf_filter_coeff354_a, rf_filter_coeff354_b,rf_filter_coeff355_a,
rf_filter_coeff355_b,rf_filter_coeff356_a, rf_filter_coeff356_b,rf_filter_coeff357_a,
rf_filter_coeff357_b,rf_filter_coeff358_a, rf_filter_coeff358_b,rf_filter_coeff359_a,
rf_filter_coeff359_b,rf_filter_coeff360_a, rf_filter_coeff360_b,rf_filter_coeff361_a,
rf_filter_coeff361_b,rf_filter_coeff362_a, rf_filter_coeff362_b,rf_filter_coeff363_a,
rf_filter_coeff363_b,rf_filter_coeff364_a, rf_filter_coeff364_b,rf_filter_coeff365_a,
rf_filter_coeff365_b,rf_filter_coeff366_a, rf_filter_coeff366_b,rf_filter_coeff367_a,
rf_filter_coeff367_b,rf_filter_coeff368_a, rf_filter_coeff368_b,rf_filter_coeff369_a,
rf_filter_coeff369_b,rf_filter_coeff370_a, rf_filter_coeff370_b,rf_filter_coeff371_a,
rf_filter_coeff371_b,rf_filter_coeff372_a, rf_filter_coeff372_b,rf_filter_coeff373_a,
rf_filter_coeff373_b,rf_filter_coeff374_a, rf_filter_coeff374_b,rf_filter_coeff375_a,
rf_filter_coeff375_b,rf_filter_coeff376_a, rf_filter_coeff376_b,rf_filter_coeff377_a,
rf_filter_coeff377_b,rf_filter_coeff378_a, rf_filter_coeff378_b,rf_filter_coeff379_a,
rf_filter_coeff379_b,rf_filter_coeff380_a, rf_filter_coeff380_b,rf_filter_coeff381_a,
rf_filter_coeff381_b,rf_filter_coeff382_a, rf_filter_coeff382_b,rf_filter_coeff383_a,
rf_filter_coeff383_b,rf_filter_coeff384_a, rf_filter_coeff384_b,rf_filter_coeff385_a,
rf_filter_coeff385_b,rf_filter_coeff386_a, rf_filter_coeff386_b,rf_filter_coeff387_a,
rf_filter_coeff387_b,rf_filter_coeff388_a, rf_filter_coeff388_b,rf_filter_coeff389_a,
rf_filter_coeff389_b,rf_filter_coeff390_a, rf_filter_coeff390_b,rf_filter_coeff391_a,
rf_filter_coeff391_b,rf_filter_coeff392_a, rf_filter_coeff392_b,rf_filter_coeff393_a,
rf_filter_coeff393_b,rf_filter_coeff394_a, rf_filter_coeff394_b,rf_filter_coeff395_a,
rf_filter_coeff395_b,rf_filter_coeff396_a, rf_filter_coeff396_b,rf_filter_coeff397_a,
rf_filter_coeff397_b,rf_filter_coeff398_a, rf_filter_coeff398_b,rf_filter_coeff399_a,
rf_filter_coeff399_b,rf_filter_coeff400_a, rf_filter_coeff400_b,rf_filter_coeff401_a,
rf_filter_coeff401_b,rf_filter_coeff402_a, rf_filter_coeff402_b,rf_filter_coeff403_a,
rf_filter_coeff403_b,rf_filter_coeff404_a, rf_filter_coeff404_b,rf_filter_coeff405_a,
rf_filter_coeff405_b,rf_filter_coeff406_a, rf_filter_coeff406_b,rf_filter_coeff407_a,
rf_filter_coeff407_b,rf_filter_coeff408_a, rf_filter_coeff408_b,rf_filter_coeff409_a,
rf_filter_coeff409_b,rf_filter_coeff410_a, rf_filter_coeff410_b,rf_filter_coeff411_a,
rf_filter_coeff411_b,rf_filter_coeff412_a, rf_filter_coeff412_b,rf_filter_coeff413_a,
rf_filter_coeff413_b,rf_filter_coeff414_a, rf_filter_coeff414_b,rf_filter_coeff415_a,
rf_filter_coeff415_b,rf_filter_coeff416_a, rf_filter_coeff416_b,rf_filter_coeff417_a,
rf_filter_coeff417_b,rf_filter_coeff418_a, rf_filter_coeff418_b,rf_filter_coeff419_a,
rf_filter_coeff419_b,rf_filter_coeff420_a, rf_filter_coeff420_b,rf_filter_coeff421_a,
rf_filter_coeff421_b,rf_filter_coeff422_a, rf_filter_coeff422_b,rf_filter_coeff423_a,
rf_filter_coeff423_b,rf_filter_coeff424_a, rf_filter_coeff424_b,rf_filter_coeff425_a,
rf_filter_coeff425_b,rf_filter_coeff426_a, rf_filter_coeff426_b,rf_filter_coeff427_a,
rf_filter_coeff427_b,rf_filter_coeff428_a, rf_filter_coeff428_b,rf_filter_coeff429_a,

```
rf_filter_coeff429_b,rf_filter_coeff430_a, rf_filter_coeff430_b,rf_filter_coeff431_a,
rf_filter_coeff431_b,rf_filter_coeff432_a, rf_filter_coeff432_b,rf_filter_coeff433_a,
rf_filter_coeff433_b,rf_filter_coeff434_a, rf_filter_coeff434_b,rf_filter_coeff435_a,
rf_filter_coeff435_b,rf_filter_coeff436_a, rf_filter_coeff436_b,rf_filter_coeff437_a,
rf_filter_coeff437_b,rf_filter_coeff438_a, rf_filter_coeff438_b,rf_filter_coeff439_a,
rf_filter_coeff439_b,rf_filter_coeff440_a, rf_filter_coeff440_b,rf_filter_coeff441_a,
rf_filter_coeff441_b,rf_filter_coeff442_a, rf_filter_coeff442_b,rf_filter_coeff443_a,
rf_filter_coeff443_b,rf_filter_coeff444_a, rf_filter_coeff444_b,rf_filter_coeff445_a,
rf_filter_coeff445_b,rf_filter_coeff446_a, rf_filter_coeff446_b,rf_filter_coeff447_a,
rf_filter_coeff447_b,rf_filter_coeff448_a, rf_filter_coeff448_b,rf_filter_coeff449_a,
rf_filter_coeff449_b,rf_filter_coeff450_a, rf_filter_coeff450_b,rf_filter_coeff451_a,
rf_filter_coeff451_b,rf_filter_coeff452_a, rf_filter_coeff452_b,rf_filter_coeff453_a,
rf_filter_coeff453_b,rf_filter_coeff454_a, rf_filter_coeff454_b,rf_filter_coeff455_a,
rf_filter_coeff455_b,rf_filter_coeff456_a, rf_filter_coeff456_b,rf_filter_coeff457_a,
rf_filter_coeff457_b,rf_filter_coeff458_a, rf_filter_coeff458_b,rf_filter_coeff459_a,
rf_filter_coeff459_b,rf_filter_coeff460_a, rf_filter_coeff460_b,rf_filter_coeff461_a,
rf_filter_coeff461_b,rf_filter_coeff462_a, rf_filter_coeff462_b,rf_filter_coeff463_a,
rf_filter_coeff463_b,rf_filter_coeff464_a, rf_filter_coeff464_b,rf_filter_coeff465_a,
rf_filter_coeff465_b,rf_filter_coeff466_a, rf_filter_coeff466_b,rf_filter_coeff467_a,
rf_filter_coeff467_b,rf_filter_coeff468_a, rf_filter_coeff468_b,rf_filter_coeff469_a,
rf_filter_coeff469_b,rf_filter_coeff470_a, rf_filter_coeff470_b,rf_filter_coeff471_a,
rf_filter_coeff471_b,rf_filter_coeff472_a, rf_filter_coeff472_b,rf_filter_coeff473_a,
rf_filter_coeff473_b,rf_filter_coeff474_a, rf_filter_coeff474_b,rf_filter_coeff475_a,
rf_filter_coeff475_b,rf_filter_coeff476_a, rf_filter_coeff476_b,rf_filter_coeff477_a,
rf_filter_coeff477_b,rf_filter_coeff478_a, rf_filter_coeff478_b,rf_filter_coeff479_a,
rf_filter_coeff479_b,rf_filter_coeff480_a, rf_filter_coeff480_b,rf_filter_coeff481_a,
rf_filter_coeff481_b,rf_filter_coeff482_a, rf_filter_coeff482_b,rf_filter_coeff483_a,
rf_filter_coeff483_b,rf_filter_coeff484_a, rf_filter_coeff484_b,rf_filter_coeff485_a,
rf_filter_coeff485_b,rf_filter_coeff486_a, rf_filter_coeff486_b,rf_filter_coeff487_a,
rf_filter_coeff487_b,rf_filter_coeff488_a, rf_filter_coeff488_b,rf_filter_coeff489_a,
rf_filter_coeff489_b,rf_filter_coeff490_a, rf_filter_coeff490_b,rf_filter_coeff491_a,
rf_filter_coeff491_b,rf_filter_coeff492_a, rf_filter_coeff492_b,rf_filter_coeff493_a,
rf_filter_coeff493_b,rf_filter_coeff494_a, rf_filter_coeff494_b,rf_filter_coeff495_a,
rf_filter_coeff495_b,rf_filter_coeff496_a, rf_filter_coeff496_b,rf_filter_coeff497_a,
rf_filter_coeff497_b,rf_filter_coeff498_a, rf_filter_coeff498_b,rf_filter_coeff499_a,
rf_filter_coeff499_b,rf_filter_coeff500_a, rf_filter_coeff500_b,rf_filter_coeff501_a,
rf_filter_coeff501_b,rf_filter_coeff502_a, rf_filter_coeff502_b,rf_filter_coeff503_a,
rf_filter_coeff503_b,rf_filter_coeff504_a, rf_filter_coeff504_b,rf_filter_coeff505_a,
rf_filter_coeff505_b,rf_filter_coeff506_a, rf_filter_coeff506_b,rf_filter_coeff507_a,
rf_filter_coeff507_b,rf_filter_coeff508_a, rf_filter_coeff508_b,rf_filter_coeff509_a,
rf_filter_coeff509_b,rf_filter_coeff510_a, rf_filter_coeff510_b,rf_filter_coeff511_a,
rf_filter_coeff511_b;
    wire trig_filter_ovf_flag_clear;        // signal to reset ro_filter_ovf_flag_clear

    // Inputs to Register Block
    wire [10:0] i2c_wraddr;                 // register address
    wire [7:0] i2c_wdata;                   // data to be written for a write op
    wire i2c_xfc_write;                     // write data transfer complete
    wire i2c_op;                            // 1- write, 0- read
    wire ro_fifo_overrun;                   // when the I2S input FIFO is full
    wire ro_fifo_underrun;                  // FIFO buffer is not full and no more data is
available
    wire ro_filter_ovf_flag;                // filter overflow flag

    // Inputs to I2C Block
    wire [2:0] i2c_addr_bits;               // 3 LSB i2c address select
    wire [7:0] i2c_rdata;                   // read return data
    wire i2c_xfc_read;                      // read data transfer complete

    // Unknown
    wire rf_soft_reset;
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------

    // INTERNAL VARIABLES
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------
    wire [7:0] rf_i2si_bist_start_val_a;     // BIST start value
    wire [3:0] rf_i2si_bist_start_val_b;     // BIST start value
    wire [7:0] rf_bist_upper_limit_a;        // BIST upper limit value
    wire [3:0] rf_bist_upper_limit_b;        // BIST upper limit value
```

```
    assign rf_bist_start_val = {rf_i2si_bist_start_val_a,rf_i2si_bist_start_val_b};
    assign rf_bist_up_limit = {rf_bist_upper_limit_a,rf_bist_upper_limit_b};
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------


    // MODULE INSTANTIATION
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------
    i2s_in I2S_Input(
        .clk                        (clk),                          // input: master clock
        .rst_n                      (rst_n),                        // input: reset not
        .inp_sck                    (i2si_sck),                     // input: serial clock
        .inp_ws                     (i2si_ws),                      // input: word select
        .inp_sd                     (i2si_sd),                      // input: serial data
        .rf_i2si_en                 (rf_i2si_en),                   // input: enable bit for
deserializer
        .rf_bist_start_val          (rf_bist_start_val),            // input: BIST start value
        .rf_bist_inc                (rf_bist_inc),                  // input: BIST increment
value
        .rf_bist_up_limit           (rf_bist_up_limit),             // input: BIST upper limit
value
        .rf_mux_en                  (rf_mux_en),                    // input: multiplexer select
bit
        .i2si_rtr                   (i2si_rtr),                     // input: ready to receive
        .i2si_data                  (filt_input_data),              // output: audio data
        .i2si_rts                   (filt_rts),                     // output: ready to send
        .trig_fifo_overrun_clr      (trig_fifo_overrun_clr),        // input: signal to reset
ro_fifo_overrun
        .ro_fifo_overrun            (ro_fifo_overrun),              // output: when the I2S input
FIFO is full
        .sync_sck                   (i2si_sync_sck),                // output: synchronized
serial clock
        .sync_sck_transition        (i2si_sck_transition)           // output: synchronized
serial clock transition
    );

    i2s_out I2S_Output(
        .clk                        (clk),                          // input: master clock
        .rst_n                      (rst_n),                        // input: reset not
        .i2so_sync_sck              (i2si_sync_sck),                // input: synchronized serial
clock
        .i2so_sck_transition        (i2si_sck_transition),          // input: serial clock
transition
        .filt_rts                   (aud_out_rts),                  // input: ready to send
        .filt_data                  (filt_out_data),                // input: audio data
        .filt_rtr                   (filt_rtr),                     // output: ready to receive
        .i2so_ws                    (i2so_ws),                      // output: word select
        .i2so_sd                    (i2so_sd),                      // output: serial data
        .i2so_sck                   (i2so_sck),                     // output: serial clock
        .trig_fifo_underrun         (trig_fifo_underrun),           // output: signal to reset
ro_fifo_underrun
        .ro_fifo_underrun           (ro_fifo_underrun)              // output: FIFO buffer is not
full and no more data is available
    );

    filter Filter(
        .clk                        (clk),                          // input: master clock
        .rst_n                      (rst_n),                        // input: reset not
        .filt_input_data            (filt_input_data),              // input: input parallel
digital audio
        .aud_in_rts                 (filt_rts),                     // input: ready to send (for
input FIFO)
        .aud_in_rtr                 (i2si_rtr),                     // output: ready to receive
(for input FIFO)
        .filt_out_data              (filt_out_data),                // output: output parallel
digital audio
        .aud_out_rts                (aud_out_rts),                  // output: ready to send (for
output FIFO)
```

```
        .aud_out_rtr                (filt_rtr),                     // input: ready to receive
(for output FIFO)
        .trig_filter_ovf_flag_clear (trig_filter_ovf_flag_clear),   // input: signal to reset
ro_filter_ovf_flag_clear
        .ro_filter_ovf_flag         (ro_filter_ovf_flag),           // output: filter overflow
flag
        .rf_filter_shift            (rf_filter_shift),              // input: # of bits to shift
after accumulator
        .rf_filter_clip_en          (rf_filter_clip_en),            // input: select bit (1-
perform clipping, 0- no clipping)
        .rf_filter_coeff0_a(rf_filter_coeff0_a),
        .rf_filter_coeff0_b(rf_filter_coeff0_b),
        .rf_filter_coeff1_a(rf_filter_coeff1_a),
        .rf_filter_coeff1_b(rf_filter_coeff1_b),
        .rf_filter_coeff2_a(rf_filter_coeff2_a),
        .rf_filter_coeff2_b(rf_filter_coeff2_b),
        .rf_filter_coeff3_a(rf_filter_coeff3_a),
        .rf_filter_coeff3_b(rf_filter_coeff3_b),
        .rf_filter_coeff4_a(rf_filter_coeff4_a),
        .rf_filter_coeff4_b(rf_filter_coeff4_b),
        .rf_filter_coeff5_a(rf_filter_coeff5_a),
        .rf_filter_coeff5_b(rf_filter_coeff5_b),
        .rf_filter_coeff6_a(rf_filter_coeff6_a),
        .rf_filter_coeff6_b(rf_filter_coeff6_b),
        .rf_filter_coeff7_a(rf_filter_coeff7_a),
        .rf_filter_coeff7_b(rf_filter_coeff7_b),
        .rf_filter_coeff8_a(rf_filter_coeff8_a),
        .rf_filter_coeff8_b(rf_filter_coeff8_b),
        .rf_filter_coeff9_a(rf_filter_coeff9_a),
        .rf_filter_coeff9_b(rf_filter_coeff9_b),
        .rf_filter_coeff10_a(rf_filter_coeff10_a),
        .rf_filter_coeff10_b(rf_filter_coeff10_b),
        .rf_filter_coeff11_a(rf_filter_coeff11_a),
        .rf_filter_coeff11_b(rf_filter_coeff11_b),
        .rf_filter_coeff12_a(rf_filter_coeff12_a),
        .rf_filter_coeff12_b(rf_filter_coeff12_b),
        .rf_filter_coeff13_a(rf_filter_coeff13_a),
        .rf_filter_coeff13_b(rf_filter_coeff13_b),
        .rf_filter_coeff14_a(rf_filter_coeff14_a),
        .rf_filter_coeff14_b(rf_filter_coeff14_b),
        .rf_filter_coeff15_a(rf_filter_coeff15_a),
        .rf_filter_coeff15_b(rf_filter_coeff15_b),
        .rf_filter_coeff16_a(rf_filter_coeff16_a),
        .rf_filter_coeff16_b(rf_filter_coeff16_b),
        .rf_filter_coeff17_a(rf_filter_coeff17_a),
        .rf_filter_coeff17_b(rf_filter_coeff17_b),
        .rf_filter_coeff18_a(rf_filter_coeff18_a),
        .rf_filter_coeff18_b(rf_filter_coeff18_b),
        .rf_filter_coeff19_a(rf_filter_coeff19_a),
        .rf_filter_coeff19_b(rf_filter_coeff19_b),
        .rf_filter_coeff20_a(rf_filter_coeff20_a),
        .rf_filter_coeff20_b(rf_filter_coeff20_b),
        .rf_filter_coeff21_a(rf_filter_coeff21_a),
        .rf_filter_coeff21_b(rf_filter_coeff21_b),
        .rf_filter_coeff22_a(rf_filter_coeff22_a),
        .rf_filter_coeff22_b(rf_filter_coeff22_b),
        .rf_filter_coeff23_a(rf_filter_coeff23_a),
        .rf_filter_coeff23_b(rf_filter_coeff23_b),
        .rf_filter_coeff24_a(rf_filter_coeff24_a),
        .rf_filter_coeff24_b(rf_filter_coeff24_b),
        .rf_filter_coeff25_a(rf_filter_coeff25_a),
        .rf_filter_coeff25_b(rf_filter_coeff25_b),
        .rf_filter_coeff26_a(rf_filter_coeff26_a),
        .rf_filter_coeff26_b(rf_filter_coeff26_b),
        .rf_filter_coeff27_a(rf_filter_coeff27_a),
        .rf_filter_coeff27_b(rf_filter_coeff27_b),
        .rf_filter_coeff28_a(rf_filter_coeff28_a),
        .rf_filter_coeff28_b(rf_filter_coeff28_b),
        .rf_filter_coeff29_a(rf_filter_coeff29_a),
        .rf_filter_coeff29_b(rf_filter_coeff29_b),
        .rf_filter_coeff30_a(rf_filter_coeff30_a),
```

```
.rf_filter_coeff30_b(rf_filter_coeff30_b),
.rf_filter_coeff31_a(rf_filter_coeff31_a),
.rf_filter_coeff31_b(rf_filter_coeff31_b),
.rf_filter_coeff32_a(rf_filter_coeff32_a),
.rf_filter_coeff32_b(rf_filter_coeff32_b),
.rf_filter_coeff33_a(rf_filter_coeff33_a),
.rf_filter_coeff33_b(rf_filter_coeff33_b),
.rf_filter_coeff34_a(rf_filter_coeff34_a),
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
```

```
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
```

```
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
```

```
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
```

```
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
```

```
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
```

```
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
```

```
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
```

```
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
```

```
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
```

```
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
```

```
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
```

```
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
.rf_filter_coeff479_a(rf_filter_coeff479_a),
.rf_filter_coeff479_b(rf_filter_coeff479_b),
.rf_filter_coeff480_a(rf_filter_coeff480_a),
.rf_filter_coeff480_b(rf_filter_coeff480_b),
.rf_filter_coeff481_a(rf_filter_coeff481_a),
.rf_filter_coeff481_b(rf_filter_coeff481_b),
.rf_filter_coeff482_a(rf_filter_coeff482_a),
.rf_filter_coeff482_b(rf_filter_coeff482_b),
.rf_filter_coeff483_a(rf_filter_coeff483_a),
.rf_filter_coeff483_b(rf_filter_coeff483_b),
.rf_filter_coeff484_a(rf_filter_coeff484_a),
.rf_filter_coeff484_b(rf_filter_coeff484_b),
.rf_filter_coeff485_a(rf_filter_coeff485_a),
.rf_filter_coeff485_b(rf_filter_coeff485_b),
.rf_filter_coeff486_a(rf_filter_coeff486_a),
.rf_filter_coeff486_b(rf_filter_coeff486_b),
.rf_filter_coeff487_a(rf_filter_coeff487_a),
.rf_filter_coeff487_b(rf_filter_coeff487_b),
.rf_filter_coeff488_a(rf_filter_coeff488_a),
.rf_filter_coeff488_b(rf_filter_coeff488_b),
.rf_filter_coeff489_a(rf_filter_coeff489_a),
.rf_filter_coeff489_b(rf_filter_coeff489_b),
.rf_filter_coeff490_a(rf_filter_coeff490_a),
.rf_filter_coeff490_b(rf_filter_coeff490_b),
.rf_filter_coeff491_a(rf_filter_coeff491_a),
.rf_filter_coeff491_b(rf_filter_coeff491_b),
```

```
        .rf_filter_coeff492_a(rf_filter_coeff492_a),
        .rf_filter_coeff492_b(rf_filter_coeff492_b),
        .rf_filter_coeff493_a(rf_filter_coeff493_a),
        .rf_filter_coeff493_b(rf_filter_coeff493_b),
        .rf_filter_coeff494_a(rf_filter_coeff494_a),
        .rf_filter_coeff494_b(rf_filter_coeff494_b),
        .rf_filter_coeff495_a(rf_filter_coeff495_a),
        .rf_filter_coeff495_b(rf_filter_coeff495_b),
        .rf_filter_coeff496_a(rf_filter_coeff496_a),
        .rf_filter_coeff496_b(rf_filter_coeff496_b),
        .rf_filter_coeff497_a(rf_filter_coeff497_a),
        .rf_filter_coeff497_b(rf_filter_coeff497_b),
        .rf_filter_coeff498_a(rf_filter_coeff498_a),
        .rf_filter_coeff498_b(rf_filter_coeff498_b),
        .rf_filter_coeff499_a(rf_filter_coeff499_a),
        .rf_filter_coeff499_b(rf_filter_coeff499_b),
        .rf_filter_coeff500_a(rf_filter_coeff500_a),
        .rf_filter_coeff500_b(rf_filter_coeff500_b),
        .rf_filter_coeff501_a(rf_filter_coeff501_a),
        .rf_filter_coeff501_b(rf_filter_coeff501_b),
        .rf_filter_coeff502_a(rf_filter_coeff502_a),
        .rf_filter_coeff502_b(rf_filter_coeff502_b),
        .rf_filter_coeff503_a(rf_filter_coeff503_a),
        .rf_filter_coeff503_b(rf_filter_coeff503_b),
        .rf_filter_coeff504_a(rf_filter_coeff504_a),
        .rf_filter_coeff504_b(rf_filter_coeff504_b),
        .rf_filter_coeff505_a(rf_filter_coeff505_a),
        .rf_filter_coeff505_b(rf_filter_coeff505_b),
        .rf_filter_coeff506_a(rf_filter_coeff506_a),
        .rf_filter_coeff506_b(rf_filter_coeff506_b),
        .rf_filter_coeff507_a(rf_filter_coeff507_a),
        .rf_filter_coeff507_b(rf_filter_coeff507_b),
        .rf_filter_coeff508_a(rf_filter_coeff508_a),
        .rf_filter_coeff508_b(rf_filter_coeff508_b),
        .rf_filter_coeff509_a(rf_filter_coeff509_a),
        .rf_filter_coeff509_b(rf_filter_coeff509_b),
        .rf_filter_coeff510_a(rf_filter_coeff510_a),
        .rf_filter_coeff510_b(rf_filter_coeff510_b),
        .rf_filter_coeff511_a(rf_filter_coeff511_a),
        .rf_filter_coeff511_b(rf_filter_coeff511_b)
    );


    chip_reg Register(
        .clk                        (clk),                          // input: master clock
        .rst_n                      (rst_n),                        // input: reset not
        .i2c_addr                   (i2c_wraddr),                   // input: register address
        .i2c_wdata                  (i2c_wdata),                    // input: data to be written
for a write op
        .i2c_xfc_write              (i2c_xfc_write),                // input: write data transfer
complete
        .i2c_op                     (i2c_op),                       // input: 1- write, 0- read
        .ro_fifo_overrun            (ro_fifo_overrun),              // input: when the I2S input
FIFO is full
        .ro_fifo_underrun           (ro_fifo_underrun),             // input: FIFO buffer is not
full and no more data is available
        .ro_filter_ovf_flag         (ro_filter_ovf_flag),           // input: filter overflow
flag
        .i2c_rdata                  (i2c_rdata),                    // output: read return data
        .i2c_xfc_read               (i2c_xfc_read),                 // output: read data transfer
complete
        .rf_i2si_bist_en            (rf_mux_en),                    // output: I2S input
multiplexer select bit
        .rf_filter_shift            (rf_filter_shift),              // output: number of bit
positions to shift after filter accumulator
        .rf_filter_clip_en          (rf_filter_clip_en),            // output: 0- no clipping 1-
performs clipping
        .rf_i2si_bist_start_val_a   (rf_i2si_bist_start_val_a),     // output: BIST start value
        .rf_i2si_bist_start_val_b   (rf_i2si_bist_start_val_b),     // output: BIST start value
        .rf_i2si_bist_incr          (rf_bist_inc),                  // output: BIST increment
value
```

```
        .rf_i2si_bist_upper_limit_a (rf_bist_upper_limit_a),        // output: BIST upper limit
value
        .rf_i2si_bist_upper_limit_b (rf_bist_upper_limit_b),        // output: BIST upper limit
value
        .rf_i2si_en                 (rf_i2si_en),                   // output: enable bit for
deserializer
        .rf_filter_coeff0_a(rf_filter_coeff0_a),
        .rf_filter_coeff0_b(rf_filter_coeff0_b),
        .rf_filter_coeff1_a(rf_filter_coeff1_a),
        .rf_filter_coeff1_b(rf_filter_coeff1_b),
        .rf_filter_coeff2_a(rf_filter_coeff2_a),
        .rf_filter_coeff2_b(rf_filter_coeff2_b),
        .rf_filter_coeff3_a(rf_filter_coeff3_a),
        .rf_filter_coeff3_b(rf_filter_coeff3_b),
        .rf_filter_coeff4_a(rf_filter_coeff4_a),
        .rf_filter_coeff4_b(rf_filter_coeff4_b),
        .rf_filter_coeff5_a(rf_filter_coeff5_a),
        .rf_filter_coeff5_b(rf_filter_coeff5_b),
        .rf_filter_coeff6_a(rf_filter_coeff6_a),
        .rf_filter_coeff6_b(rf_filter_coeff6_b),
        .rf_filter_coeff7_a(rf_filter_coeff7_a),
        .rf_filter_coeff7_b(rf_filter_coeff7_b),
        .rf_filter_coeff8_a(rf_filter_coeff8_a),
        .rf_filter_coeff8_b(rf_filter_coeff8_b),
        .rf_filter_coeff9_a(rf_filter_coeff9_a),
        .rf_filter_coeff9_b(rf_filter_coeff9_b),
        .rf_filter_coeff10_a(rf_filter_coeff10_a),
        .rf_filter_coeff10_b(rf_filter_coeff10_b),
        .rf_filter_coeff11_a(rf_filter_coeff11_a),
        .rf_filter_coeff11_b(rf_filter_coeff11_b),
        .rf_filter_coeff12_a(rf_filter_coeff12_a),
        .rf_filter_coeff12_b(rf_filter_coeff12_b),
        .rf_filter_coeff13_a(rf_filter_coeff13_a),
        .rf_filter_coeff13_b(rf_filter_coeff13_b),
        .rf_filter_coeff14_a(rf_filter_coeff14_a),
        .rf_filter_coeff14_b(rf_filter_coeff14_b),
        .rf_filter_coeff15_a(rf_filter_coeff15_a),
        .rf_filter_coeff15_b(rf_filter_coeff15_b),
        .rf_filter_coeff16_a(rf_filter_coeff16_a),
        .rf_filter_coeff16_b(rf_filter_coeff16_b),
        .rf_filter_coeff17_a(rf_filter_coeff17_a),
        .rf_filter_coeff17_b(rf_filter_coeff17_b),
        .rf_filter_coeff18_a(rf_filter_coeff18_a),
        .rf_filter_coeff18_b(rf_filter_coeff18_b),
        .rf_filter_coeff19_a(rf_filter_coeff19_a),
        .rf_filter_coeff19_b(rf_filter_coeff19_b),
        .rf_filter_coeff20_a(rf_filter_coeff20_a),
        .rf_filter_coeff20_b(rf_filter_coeff20_b),
        .rf_filter_coeff21_a(rf_filter_coeff21_a),
        .rf_filter_coeff21_b(rf_filter_coeff21_b),
        .rf_filter_coeff22_a(rf_filter_coeff22_a),
        .rf_filter_coeff22_b(rf_filter_coeff22_b),
        .rf_filter_coeff23_a(rf_filter_coeff23_a),
        .rf_filter_coeff23_b(rf_filter_coeff23_b),
        .rf_filter_coeff24_a(rf_filter_coeff24_a),
        .rf_filter_coeff24_b(rf_filter_coeff24_b),
        .rf_filter_coeff25_a(rf_filter_coeff25_a),
        .rf_filter_coeff25_b(rf_filter_coeff25_b),
        .rf_filter_coeff26_a(rf_filter_coeff26_a),
        .rf_filter_coeff26_b(rf_filter_coeff26_b),
        .rf_filter_coeff27_a(rf_filter_coeff27_a),
        .rf_filter_coeff27_b(rf_filter_coeff27_b),
        .rf_filter_coeff28_a(rf_filter_coeff28_a),
        .rf_filter_coeff28_b(rf_filter_coeff28_b),
        .rf_filter_coeff29_a(rf_filter_coeff29_a),
        .rf_filter_coeff29_b(rf_filter_coeff29_b),
        .rf_filter_coeff30_a(rf_filter_coeff30_a),
        .rf_filter_coeff30_b(rf_filter_coeff30_b),
        .rf_filter_coeff31_a(rf_filter_coeff31_a),
        .rf_filter_coeff31_b(rf_filter_coeff31_b),
        .rf_filter_coeff32_a(rf_filter_coeff32_a),
```

```
.rf_filter_coeff32_b(rf_filter_coeff32_b),
.rf_filter_coeff33_a(rf_filter_coeff33_a),
.rf_filter_coeff33_b(rf_filter_coeff33_b),
.rf_filter_coeff34_a(rf_filter_coeff34_a),
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
```

```
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
```

```
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
```

```
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
```

```
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
```

```
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
```

```
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
```

```
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
```

```
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
```

```
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
```

```
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
```

```
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
```

```
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
.rf_filter_coeff479_a(rf_filter_coeff479_a),
.rf_filter_coeff479_b(rf_filter_coeff479_b),
.rf_filter_coeff480_a(rf_filter_coeff480_a),
.rf_filter_coeff480_b(rf_filter_coeff480_b),
.rf_filter_coeff481_a(rf_filter_coeff481_a),
.rf_filter_coeff481_b(rf_filter_coeff481_b),
.rf_filter_coeff482_a(rf_filter_coeff482_a),
.rf_filter_coeff482_b(rf_filter_coeff482_b),
.rf_filter_coeff483_a(rf_filter_coeff483_a),
.rf_filter_coeff483_b(rf_filter_coeff483_b),
.rf_filter_coeff484_a(rf_filter_coeff484_a),
.rf_filter_coeff484_b(rf_filter_coeff484_b),
.rf_filter_coeff485_a(rf_filter_coeff485_a),
.rf_filter_coeff485_b(rf_filter_coeff485_b),
.rf_filter_coeff486_a(rf_filter_coeff486_a),
.rf_filter_coeff486_b(rf_filter_coeff486_b),
.rf_filter_coeff487_a(rf_filter_coeff487_a),
.rf_filter_coeff487_b(rf_filter_coeff487_b),
.rf_filter_coeff488_a(rf_filter_coeff488_a),
.rf_filter_coeff488_b(rf_filter_coeff488_b),
.rf_filter_coeff489_a(rf_filter_coeff489_a),
.rf_filter_coeff489_b(rf_filter_coeff489_b),
.rf_filter_coeff490_a(rf_filter_coeff490_a),
.rf_filter_coeff490_b(rf_filter_coeff490_b),
.rf_filter_coeff491_a(rf_filter_coeff491_a),
.rf_filter_coeff491_b(rf_filter_coeff491_b),
.rf_filter_coeff492_a(rf_filter_coeff492_a),
.rf_filter_coeff492_b(rf_filter_coeff492_b),
.rf_filter_coeff493_a(rf_filter_coeff493_a),
.rf_filter_coeff493_b(rf_filter_coeff493_b),
```

```
        .rf_filter_coeff494_a(rf_filter_coeff494_a),
        .rf_filter_coeff494_b(rf_filter_coeff494_b),
        .rf_filter_coeff495_a(rf_filter_coeff495_a),
        .rf_filter_coeff495_b(rf_filter_coeff495_b),
        .rf_filter_coeff496_a(rf_filter_coeff496_a),
        .rf_filter_coeff496_b(rf_filter_coeff496_b),
        .rf_filter_coeff497_a(rf_filter_coeff497_a),
        .rf_filter_coeff497_b(rf_filter_coeff497_b),
        .rf_filter_coeff498_a(rf_filter_coeff498_a),
        .rf_filter_coeff498_b(rf_filter_coeff498_b),
        .rf_filter_coeff499_a(rf_filter_coeff499_a),
        .rf_filter_coeff499_b(rf_filter_coeff499_b),
        .rf_filter_coeff500_a(rf_filter_coeff500_a),
        .rf_filter_coeff500_b(rf_filter_coeff500_b),
        .rf_filter_coeff501_a(rf_filter_coeff501_a),
        .rf_filter_coeff501_b(rf_filter_coeff501_b),
        .rf_filter_coeff502_a(rf_filter_coeff502_a),
        .rf_filter_coeff502_b(rf_filter_coeff502_b),
        .rf_filter_coeff503_a(rf_filter_coeff503_a),
        .rf_filter_coeff503_b(rf_filter_coeff503_b),
        .rf_filter_coeff504_a(rf_filter_coeff504_a),
        .rf_filter_coeff504_b(rf_filter_coeff504_b),
        .rf_filter_coeff505_a(rf_filter_coeff505_a),
        .rf_filter_coeff505_b(rf_filter_coeff505_b),
        .rf_filter_coeff506_a(rf_filter_coeff506_a),
        .rf_filter_coeff506_b(rf_filter_coeff506_b),
        .rf_filter_coeff507_a(rf_filter_coeff507_a),
        .rf_filter_coeff507_b(rf_filter_coeff507_b),
        .rf_filter_coeff508_a(rf_filter_coeff508_a),
        .rf_filter_coeff508_b(rf_filter_coeff508_b),
        .rf_filter_coeff509_a(rf_filter_coeff509_a),
        .rf_filter_coeff509_b(rf_filter_coeff509_b),
        .rf_filter_coeff510_a(rf_filter_coeff510_a),
        .rf_filter_coeff510_b(rf_filter_coeff510_b),
        .rf_filter_coeff511_a(rf_filter_coeff511_a),
        .rf_filter_coeff511_b(rf_filter_coeff511_b),
        .trig_fifo_overrun          (trig_fifo_overrun_clr),        // output: signal to reset
I2S input FIFO overrun
        .trig_fifo_underrun         (trig_fifo_underrun),           // output: signal to reset
ro_fifo_underrun
        .trig_filter_ovf_flag_clear (trig_filter_ovf_flag_clear)    // output: signal to reset
ro_filter_ovf_flag_clear
    );

    i2c I2C(
        .i2c_addr_bits              (i2c_addr_bits),                // input: 3 LSB I2C address
select
        .i2c_sda_in                 (i2c_sda_in),                   // input: serial data input
        .i2c_scl                    (i2c_scl),                      // input: serial clock
        .i2c_op                     (i2c_op),                       // output: 1- write, 0- read
        .i2c_xfc_write              (i2c_xfc_write),                // output: write data
transfer complete
        .i2c_wraddr                 (i2c_wraddr),                   // output: write address
        .i2c_wdata                  (i2c_wdata),                    // output: data to be written
for a write op
        .i2c_rdata                  (i2c_rdata),                    // input: read return data
        .i2c_xfc_read               (i2c_xfc_read),                 // input: read data transfer
complete
        .i2c_sda_out                (i2c_sda_out),                  // output: serial data output
        .clk                        (clk),                          // input: master clock
        .reset                      (rst_n)                         // input: reset not
    );

    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------

endmodule
```

filter.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter.v
// Create Date:             10/15/2016
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             Filters the I2S input stream and send it to the I2S output module
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module filter(clk, rst_n, filt_input_data, aud_in_rts, aud_in_rtr, filt_out_data,
aud_out_rts,aud_out_rtr, rf_filter_shift, rf_filter_clip_en, trig_filter_ovf_flag_clear,
ro_filter_ovf_flag,
rf_filter_coeff0_a, rf_filter_coeff0_b,rf_filter_coeff1_a, rf_filter_coeff1_b,rf_filter_coeff2_a,
rf_filter_coeff2_b,rf_filter_coeff3_a, rf_filter_coeff3_b,rf_filter_coeff4_a,
rf_filter_coeff4_b,rf_filter_coeff5_a, rf_filter_coeff5_b,rf_filter_coeff6_a,
rf_filter_coeff6_b,rf_filter_coeff7_a, rf_filter_coeff7_b,rf_filter_coeff8_a,
rf_filter_coeff8_b,rf_filter_coeff9_a, rf_filter_coeff9_b,rf_filter_coeff10_a,
rf_filter_coeff10_b,rf_filter_coeff11_a, rf_filter_coeff11_b,rf_filter_coeff12_a,
rf_filter_coeff12_b,rf_filter_coeff13_a, rf_filter_coeff13_b,rf_filter_coeff14_a,
rf_filter_coeff14_b,rf_filter_coeff15_a, rf_filter_coeff15_b,rf_filter_coeff16_a,
rf_filter_coeff16_b,rf_filter_coeff17_a, rf_filter_coeff17_b,rf_filter_coeff18_a,
rf_filter_coeff18_b,rf_filter_coeff19_a, rf_filter_coeff19_b,rf_filter_coeff20_a,
rf_filter_coeff20_b,rf_filter_coeff21_a, rf_filter_coeff21_b,rf_filter_coeff22_a,
rf_filter_coeff22_b,rf_filter_coeff23_a, rf_filter_coeff23_b,rf_filter_coeff24_a,
rf_filter_coeff24_b,rf_filter_coeff25_a, rf_filter_coeff25_b,rf_filter_coeff26_a,
rf_filter_coeff26_b,rf_filter_coeff27_a, rf_filter_coeff27_b,rf_filter_coeff28_a,
rf_filter_coeff28_b,rf_filter_coeff29_a, rf_filter_coeff29_b,rf_filter_coeff30_a,
rf_filter_coeff30_b,rf_filter_coeff31_a, rf_filter_coeff31_b,rf_filter_coeff32_a,
rf_filter_coeff32_b,rf_filter_coeff33_a, rf_filter_coeff33_b,rf_filter_coeff34_a,
rf_filter_coeff34_b,rf_filter_coeff35_a, rf_filter_coeff35_b,rf_filter_coeff36_a,
rf_filter_coeff36_b,rf_filter_coeff37_a, rf_filter_coeff37_b,rf_filter_coeff38_a,
rf_filter_coeff38_b,rf_filter_coeff39_a, rf_filter_coeff39_b,rf_filter_coeff40_a,
rf_filter_coeff40_b,rf_filter_coeff41_a, rf_filter_coeff41_b,rf_filter_coeff42_a,
rf_filter_coeff42_b,rf_filter_coeff43_a, rf_filter_coeff43_b,rf_filter_coeff44_a,
rf_filter_coeff44_b,rf_filter_coeff45_a, rf_filter_coeff45_b,rf_filter_coeff46_a,
rf_filter_coeff46_b,rf_filter_coeff47_a, rf_filter_coeff47_b,rf_filter_coeff48_a,
rf_filter_coeff48_b,rf_filter_coeff49_a, rf_filter_coeff49_b,rf_filter_coeff50_a,
rf_filter_coeff50_b,rf_filter_coeff51_a, rf_filter_coeff51_b,rf_filter_coeff52_a,
rf_filter_coeff52_b,rf_filter_coeff53_a, rf_filter_coeff53_b,rf_filter_coeff54_a,
rf_filter_coeff54_b,rf_filter_coeff55_a, rf_filter_coeff55_b,rf_filter_coeff56_a,
rf_filter_coeff56_b,rf_filter_coeff57_a, rf_filter_coeff57_b,rf_filter_coeff58_a,
rf_filter_coeff58_b,rf_filter_coeff59_a, rf_filter_coeff59_b,rf_filter_coeff60_a,
rf_filter_coeff60_b,rf_filter_coeff61_a, rf_filter_coeff61_b,rf_filter_coeff62_a,
rf_filter_coeff62_b,rf_filter_coeff63_a, rf_filter_coeff63_b,rf_filter_coeff64_a,
rf_filter_coeff64_b,rf_filter_coeff65_a, rf_filter_coeff65_b,rf_filter_coeff66_a,
rf_filter_coeff66_b,rf_filter_coeff67_a, rf_filter_coeff67_b,rf_filter_coeff68_a,
rf_filter_coeff68_b,rf_filter_coeff69_a, rf_filter_coeff69_b,rf_filter_coeff70_a,
rf_filter_coeff70_b,rf_filter_coeff71_a, rf_filter_coeff71_b,rf_filter_coeff72_a,
rf_filter_coeff72_b,rf_filter_coeff73_a, rf_filter_coeff73_b,rf_filter_coeff74_a,
rf_filter_coeff74_b,rf_filter_coeff75_a, rf_filter_coeff75_b,rf_filter_coeff76_a,
rf_filter_coeff76_b,rf_filter_coeff77_a, rf_filter_coeff77_b,rf_filter_coeff78_a,
rf_filter_coeff78_b,rf_filter_coeff79_a, rf_filter_coeff79_b,rf_filter_coeff80_a,
rf_filter_coeff80_b,rf_filter_coeff81_a, rf_filter_coeff81_b,rf_filter_coeff82_a,
rf_filter_coeff82_b,rf_filter_coeff83_a, rf_filter_coeff83_b,rf_filter_coeff84_a,
rf_filter_coeff84_b,rf_filter_coeff85_a, rf_filter_coeff85_b,rf_filter_coeff86_a,
rf_filter_coeff86_b,rf_filter_coeff87_a, rf_filter_coeff87_b,rf_filter_coeff88_a,
rf_filter_coeff88_b,rf_filter_coeff89_a, rf_filter_coeff89_b,rf_filter_coeff90_a,
rf_filter_coeff90_b,rf_filter_coeff91_a, rf_filter_coeff91_b,rf_filter_coeff92_a,
rf_filter_coeff92_b,rf_filter_coeff93_a, rf_filter_coeff93_b,rf_filter_coeff94_a,
rf_filter_coeff94_b,rf_filter_coeff95_a, rf_filter_coeff95_b,rf_filter_coeff96_a,
rf_filter_coeff96_b,rf_filter_coeff97_a, rf_filter_coeff97_b,rf_filter_coeff98_a,
rf_filter_coeff98_b,rf_filter_coeff99_a, rf_filter_coeff99_b,rf_filter_coeff100_a,
rf_filter_coeff100_b,rf_filter_coeff101_a, rf_filter_coeff101_b,rf_filter_coeff102_a,
rf_filter_coeff102_b,rf_filter_coeff103_a, rf_filter_coeff103_b,rf_filter_coeff104_a,
rf_filter_coeff104_b,rf_filter_coeff105_a, rf_filter_coeff105_b,rf_filter_coeff106_a,
rf_filter_coeff106_b,rf_filter_coeff107_a, rf_filter_coeff107_b,rf_filter_coeff108_a,
rf_filter_coeff108_b,rf_filter_coeff109_a, rf_filter_coeff109_b,rf_filter_coeff110_a,
```

rf_filter_coeff110_b,rf_filter_coeff111_a, rf_filter_coeff111_b,rf_filter_coeff112_a,
rf_filter_coeff112_b,rf_filter_coeff113_a, rf_filter_coeff113_b,rf_filter_coeff114_a,
rf_filter_coeff114_b,rf_filter_coeff115_a, rf_filter_coeff115_b,rf_filter_coeff116_a,
rf_filter_coeff116_b,rf_filter_coeff117_a, rf_filter_coeff117_b,rf_filter_coeff118_a,
rf_filter_coeff118_b,rf_filter_coeff119_a, rf_filter_coeff119_b,rf_filter_coeff120_a,
rf_filter_coeff120_b,rf_filter_coeff121_a, rf_filter_coeff121_b,rf_filter_coeff122_a,
rf_filter_coeff122_b,rf_filter_coeff123_a, rf_filter_coeff123_b,rf_filter_coeff124_a,
rf_filter_coeff124_b,rf_filter_coeff125_a, rf_filter_coeff125_b,rf_filter_coeff126_a,
rf_filter_coeff126_b,rf_filter_coeff127_a, rf_filter_coeff127_b,rf_filter_coeff128_a,
rf_filter_coeff128_b,rf_filter_coeff129_a, rf_filter_coeff129_b,rf_filter_coeff130_a,
rf_filter_coeff130_b,rf_filter_coeff131_a, rf_filter_coeff131_b,rf_filter_coeff132_a,
rf_filter_coeff132_b,rf_filter_coeff133_a, rf_filter_coeff133_b,rf_filter_coeff134_a,
rf_filter_coeff134_b,rf_filter_coeff135_a, rf_filter_coeff135_b,rf_filter_coeff136_a,
rf_filter_coeff136_b,rf_filter_coeff137_a, rf_filter_coeff137_b,rf_filter_coeff138_a,
rf_filter_coeff138_b,rf_filter_coeff139_a, rf_filter_coeff139_b,rf_filter_coeff140_a,
rf_filter_coeff140_b,rf_filter_coeff141_a, rf_filter_coeff141_b,rf_filter_coeff142_a,
rf_filter_coeff142_b,rf_filter_coeff143_a, rf_filter_coeff143_b,rf_filter_coeff144_a,
rf_filter_coeff144_b,rf_filter_coeff145_a, rf_filter_coeff145_b,rf_filter_coeff146_a,
rf_filter_coeff146_b,rf_filter_coeff147_a, rf_filter_coeff147_b,rf_filter_coeff148_a,
rf_filter_coeff148_b,rf_filter_coeff149_a, rf_filter_coeff149_b,rf_filter_coeff150_a,
rf_filter_coeff150_b,rf_filter_coeff151_a, rf_filter_coeff151_b,rf_filter_coeff152_a,
rf_filter_coeff152_b,rf_filter_coeff153_a, rf_filter_coeff153_b,rf_filter_coeff154_a,
rf_filter_coeff154_b,rf_filter_coeff155_a, rf_filter_coeff155_b,rf_filter_coeff156_a,
rf_filter_coeff156_b,rf_filter_coeff157_a, rf_filter_coeff157_b,rf_filter_coeff158_a,
rf_filter_coeff158_b,rf_filter_coeff159_a, rf_filter_coeff159_b,rf_filter_coeff160_a,
rf_filter_coeff160_b,rf_filter_coeff161_a, rf_filter_coeff161_b,rf_filter_coeff162_a,
rf_filter_coeff162_b,rf_filter_coeff163_a, rf_filter_coeff163_b,rf_filter_coeff164_a,
rf_filter_coeff164_b,rf_filter_coeff165_a, rf_filter_coeff165_b,rf_filter_coeff166_a,
rf_filter_coeff166_b,rf_filter_coeff167_a, rf_filter_coeff167_b,rf_filter_coeff168_a,
rf_filter_coeff168_b,rf_filter_coeff169_a, rf_filter_coeff169_b,rf_filter_coeff170_a,
rf_filter_coeff170_b,rf_filter_coeff171_a, rf_filter_coeff171_b,rf_filter_coeff172_a,
rf_filter_coeff172_b,rf_filter_coeff173_a, rf_filter_coeff173_b,rf_filter_coeff174_a,
rf_filter_coeff174_b,rf_filter_coeff175_a, rf_filter_coeff175_b,rf_filter_coeff176_a,
rf_filter_coeff176_b,rf_filter_coeff177_a, rf_filter_coeff177_b,rf_filter_coeff178_a,
rf_filter_coeff178_b,rf_filter_coeff179_a, rf_filter_coeff179_b,rf_filter_coeff180_a,
rf_filter_coeff180_b,rf_filter_coeff181_a, rf_filter_coeff181_b,rf_filter_coeff182_a,
rf_filter_coeff182_b,rf_filter_coeff183_a, rf_filter_coeff183_b,rf_filter_coeff184_a,
rf_filter_coeff184_b,rf_filter_coeff185_a, rf_filter_coeff185_b,rf_filter_coeff186_a,
rf_filter_coeff186_b,rf_filter_coeff187_a, rf_filter_coeff187_b,rf_filter_coeff188_a,
rf_filter_coeff188_b,rf_filter_coeff189_a, rf_filter_coeff189_b,rf_filter_coeff190_a,
rf_filter_coeff190_b,rf_filter_coeff191_a, rf_filter_coeff191_b,rf_filter_coeff192_a,
rf_filter_coeff192_b,rf_filter_coeff193_a, rf_filter_coeff193_b,rf_filter_coeff194_a,
rf_filter_coeff194_b,rf_filter_coeff195_a, rf_filter_coeff195_b,rf_filter_coeff196_a,
rf_filter_coeff196_b,rf_filter_coeff197_a, rf_filter_coeff197_b,rf_filter_coeff198_a,
rf_filter_coeff198_b,rf_filter_coeff199_a, rf_filter_coeff199_b,rf_filter_coeff200_a,
rf_filter_coeff200_b,rf_filter_coeff201_a, rf_filter_coeff201_b,rf_filter_coeff202_a,
rf_filter_coeff202_b,rf_filter_coeff203_a, rf_filter_coeff203_b,rf_filter_coeff204_a,
rf_filter_coeff204_b,rf_filter_coeff205_a, rf_filter_coeff205_b,rf_filter_coeff206_a,
rf_filter_coeff206_b,rf_filter_coeff207_a, rf_filter_coeff207_b,rf_filter_coeff208_a,
rf_filter_coeff208_b,rf_filter_coeff209_a, rf_filter_coeff209_b,rf_filter_coeff210_a,
rf_filter_coeff210_b,rf_filter_coeff211_a, rf_filter_coeff211_b,rf_filter_coeff212_a,
rf_filter_coeff212_b,rf_filter_coeff213_a, rf_filter_coeff213_b,rf_filter_coeff214_a,
rf_filter_coeff214_b,rf_filter_coeff215_a, rf_filter_coeff215_b,rf_filter_coeff216_a,
rf_filter_coeff216_b,rf_filter_coeff217_a, rf_filter_coeff217_b,rf_filter_coeff218_a,
rf_filter_coeff218_b,rf_filter_coeff219_a, rf_filter_coeff219_b,rf_filter_coeff220_a,
rf_filter_coeff220_b,rf_filter_coeff221_a, rf_filter_coeff221_b,rf_filter_coeff222_a,
rf_filter_coeff222_b,rf_filter_coeff223_a, rf_filter_coeff223_b,rf_filter_coeff224_a,
rf_filter_coeff224_b,rf_filter_coeff225_a, rf_filter_coeff225_b,rf_filter_coeff226_a,
rf_filter_coeff226_b,rf_filter_coeff227_a, rf_filter_coeff227_b,rf_filter_coeff228_a,
rf_filter_coeff228_b,rf_filter_coeff229_a, rf_filter_coeff229_b,rf_filter_coeff230_a,
rf_filter_coeff230_b,rf_filter_coeff231_a, rf_filter_coeff231_b,rf_filter_coeff232_a,
rf_filter_coeff232_b,rf_filter_coeff233_a, rf_filter_coeff233_b,rf_filter_coeff234_a,
rf_filter_coeff234_b,rf_filter_coeff235_a, rf_filter_coeff235_b,rf_filter_coeff236_a,
rf_filter_coeff236_b,rf_filter_coeff237_a, rf_filter_coeff237_b,rf_filter_coeff238_a,
rf_filter_coeff238_b,rf_filter_coeff239_a, rf_filter_coeff239_b,rf_filter_coeff240_a,
rf_filter_coeff240_b,rf_filter_coeff241_a, rf_filter_coeff241_b,rf_filter_coeff242_a,
rf_filter_coeff242_b,rf_filter_coeff243_a, rf_filter_coeff243_b,rf_filter_coeff244_a,
rf_filter_coeff244_b,rf_filter_coeff245_a, rf_filter_coeff245_b,rf_filter_coeff246_a,
rf_filter_coeff246_b,rf_filter_coeff247_a, rf_filter_coeff247_b,rf_filter_coeff248_a,
rf_filter_coeff248_b,rf_filter_coeff249_a, rf_filter_coeff249_b,rf_filter_coeff250_a,
rf_filter_coeff250_b,rf_filter_coeff251_a, rf_filter_coeff251_b,rf_filter_coeff252_a,

rf_filter_coeff252_b,rf_filter_coeff253_a, rf_filter_coeff253_b,rf_filter_coeff254_a,
rf_filter_coeff254_b,rf_filter_coeff255_a, rf_filter_coeff255_b,rf_filter_coeff256_a,
rf_filter_coeff256_b,rf_filter_coeff257_a, rf_filter_coeff257_b,rf_filter_coeff258_a,
rf_filter_coeff258_b,rf_filter_coeff259_a, rf_filter_coeff259_b,rf_filter_coeff260_a,
rf_filter_coeff260_b,rf_filter_coeff261_a, rf_filter_coeff261_b,rf_filter_coeff262_a,
rf_filter_coeff262_b,rf_filter_coeff263_a, rf_filter_coeff263_b,rf_filter_coeff264_a,
rf_filter_coeff264_b,rf_filter_coeff265_a, rf_filter_coeff265_b,rf_filter_coeff266_a,
rf_filter_coeff266_b,rf_filter_coeff267_a, rf_filter_coeff267_b,rf_filter_coeff268_a,
rf_filter_coeff268_b,rf_filter_coeff269_a, rf_filter_coeff269_b,rf_filter_coeff270_a,
rf_filter_coeff270_b,rf_filter_coeff271_a, rf_filter_coeff271_b,rf_filter_coeff272_a,
rf_filter_coeff272_b,rf_filter_coeff273_a, rf_filter_coeff273_b,rf_filter_coeff274_a,
rf_filter_coeff274_b,rf_filter_coeff275_a, rf_filter_coeff275_b,rf_filter_coeff276_a,
rf_filter_coeff276_b,rf_filter_coeff277_a, rf_filter_coeff277_b,rf_filter_coeff278_a,
rf_filter_coeff278_b,rf_filter_coeff279_a, rf_filter_coeff279_b,rf_filter_coeff280_a,
rf_filter_coeff280_b,rf_filter_coeff281_a, rf_filter_coeff281_b,rf_filter_coeff282_a,
rf_filter_coeff282_b,rf_filter_coeff283_a, rf_filter_coeff283_b,rf_filter_coeff284_a,
rf_filter_coeff284_b,rf_filter_coeff285_a, rf_filter_coeff285_b,rf_filter_coeff286_a,
rf_filter_coeff286_b,rf_filter_coeff287_a, rf_filter_coeff287_b,rf_filter_coeff288_a,
rf_filter_coeff288_b,rf_filter_coeff289_a, rf_filter_coeff289_b,rf_filter_coeff290_a,
rf_filter_coeff290_b,rf_filter_coeff291_a, rf_filter_coeff291_b,rf_filter_coeff292_a,
rf_filter_coeff292_b,rf_filter_coeff293_a, rf_filter_coeff293_b,rf_filter_coeff294_a,
rf_filter_coeff294_b,rf_filter_coeff295_a, rf_filter_coeff295_b,rf_filter_coeff296_a,
rf_filter_coeff296_b,rf_filter_coeff297_a, rf_filter_coeff297_b,rf_filter_coeff298_a,
rf_filter_coeff298_b,rf_filter_coeff299_a, rf_filter_coeff299_b,rf_filter_coeff300_a,
rf_filter_coeff300_b,rf_filter_coeff301_a, rf_filter_coeff301_b,rf_filter_coeff302_a,
rf_filter_coeff302_b,rf_filter_coeff303_a, rf_filter_coeff303_b,rf_filter_coeff304_a,
rf_filter_coeff304_b,rf_filter_coeff305_a, rf_filter_coeff305_b,rf_filter_coeff306_a,
rf_filter_coeff306_b,rf_filter_coeff307_a, rf_filter_coeff307_b,rf_filter_coeff308_a,
rf_filter_coeff308_b,rf_filter_coeff309_a, rf_filter_coeff309_b,rf_filter_coeff310_a,
rf_filter_coeff310_b,rf_filter_coeff311_a, rf_filter_coeff311_b,rf_filter_coeff312_a,
rf_filter_coeff312_b,rf_filter_coeff313_a, rf_filter_coeff313_b,rf_filter_coeff314_a,
rf_filter_coeff314_b,rf_filter_coeff315_a, rf_filter_coeff315_b,rf_filter_coeff316_a,
rf_filter_coeff316_b,rf_filter_coeff317_a, rf_filter_coeff317_b,rf_filter_coeff318_a,
rf_filter_coeff318_b,rf_filter_coeff319_a, rf_filter_coeff319_b,rf_filter_coeff320_a,
rf_filter_coeff320_b,rf_filter_coeff321_a, rf_filter_coeff321_b,rf_filter_coeff322_a,
rf_filter_coeff322_b,rf_filter_coeff323_a, rf_filter_coeff323_b,rf_filter_coeff324_a,
rf_filter_coeff324_b,rf_filter_coeff325_a, rf_filter_coeff325_b,rf_filter_coeff326_a,
rf_filter_coeff326_b,rf_filter_coeff327_a, rf_filter_coeff327_b,rf_filter_coeff328_a,
rf_filter_coeff328_b,rf_filter_coeff329_a, rf_filter_coeff329_b,rf_filter_coeff330_a,
rf_filter_coeff330_b,rf_filter_coeff331_a, rf_filter_coeff331_b,rf_filter_coeff332_a,
rf_filter_coeff332_b,rf_filter_coeff333_a, rf_filter_coeff333_b,rf_filter_coeff334_a,
rf_filter_coeff334_b,rf_filter_coeff335_a, rf_filter_coeff335_b,rf_filter_coeff336_a,
rf_filter_coeff336_b,rf_filter_coeff337_a, rf_filter_coeff337_b,rf_filter_coeff338_a,
rf_filter_coeff338_b,rf_filter_coeff339_a, rf_filter_coeff339_b,rf_filter_coeff340_a,
rf_filter_coeff340_b,rf_filter_coeff341_a, rf_filter_coeff341_b,rf_filter_coeff342_a,
rf_filter_coeff342_b,rf_filter_coeff343_a, rf_filter_coeff343_b,rf_filter_coeff344_a,
rf_filter_coeff344_b,rf_filter_coeff345_a, rf_filter_coeff345_b,rf_filter_coeff346_a,
rf_filter_coeff346_b,rf_filter_coeff347_a, rf_filter_coeff347_b,rf_filter_coeff348_a,
rf_filter_coeff348_b,rf_filter_coeff349_a, rf_filter_coeff349_b,rf_filter_coeff350_a,
rf_filter_coeff350_b,rf_filter_coeff351_a, rf_filter_coeff351_b,rf_filter_coeff352_a,
rf_filter_coeff352_b,rf_filter_coeff353_a, rf_filter_coeff353_b,rf_filter_coeff354_a,
rf_filter_coeff354_b,rf_filter_coeff355_a, rf_filter_coeff355_b,rf_filter_coeff356_a,
rf_filter_coeff356_b,rf_filter_coeff357_a, rf_filter_coeff357_b,rf_filter_coeff358_a,
rf_filter_coeff358_b,rf_filter_coeff359_a, rf_filter_coeff359_b,rf_filter_coeff360_a,
rf_filter_coeff360_b,rf_filter_coeff361_a, rf_filter_coeff361_b,rf_filter_coeff362_a,
rf_filter_coeff362_b,rf_filter_coeff363_a, rf_filter_coeff363_b,rf_filter_coeff364_a,
rf_filter_coeff364_b,rf_filter_coeff365_a, rf_filter_coeff365_b,rf_filter_coeff366_a,
rf_filter_coeff366_b,rf_filter_coeff367_a, rf_filter_coeff367_b,rf_filter_coeff368_a,
rf_filter_coeff368_b,rf_filter_coeff369_a, rf_filter_coeff369_b,rf_filter_coeff370_a,
rf_filter_coeff370_b,rf_filter_coeff371_a, rf_filter_coeff371_b,rf_filter_coeff372_a,
rf_filter_coeff372_b,rf_filter_coeff373_a, rf_filter_coeff373_b,rf_filter_coeff374_a,
rf_filter_coeff374_b,rf_filter_coeff375_a, rf_filter_coeff375_b,rf_filter_coeff376_a,
rf_filter_coeff376_b,rf_filter_coeff377_a, rf_filter_coeff377_b,rf_filter_coeff378_a,
rf_filter_coeff378_b,rf_filter_coeff379_a, rf_filter_coeff379_b,rf_filter_coeff380_a,
rf_filter_coeff380_b,rf_filter_coeff381_a, rf_filter_coeff381_b,rf_filter_coeff382_a,
rf_filter_coeff382_b,rf_filter_coeff383_a, rf_filter_coeff383_b,rf_filter_coeff384_a,
rf_filter_coeff384_b,rf_filter_coeff385_a, rf_filter_coeff385_b,rf_filter_coeff386_a,
rf_filter_coeff386_b,rf_filter_coeff387_a, rf_filter_coeff387_b,rf_filter_coeff388_a,
rf_filter_coeff388_b,rf_filter_coeff389_a, rf_filter_coeff389_b,rf_filter_coeff390_a,
rf_filter_coeff390_b,rf_filter_coeff391_a, rf_filter_coeff391_b,rf_filter_coeff392_a,
rf_filter_coeff392_b,rf_filter_coeff393_a, rf_filter_coeff393_b,rf_filter_coeff394_a,

```
rf_filter_coeff394_b,rf_filter_coeff395_a, rf_filter_coeff395_b,rf_filter_coeff396_a,
rf_filter_coeff396_b,rf_filter_coeff397_a, rf_filter_coeff397_b,rf_filter_coeff398_a,
rf_filter_coeff398_b,rf_filter_coeff399_a, rf_filter_coeff399_b,rf_filter_coeff400_a,
rf_filter_coeff400_b,rf_filter_coeff401_a, rf_filter_coeff401_b,rf_filter_coeff402_a,
rf_filter_coeff402_b,rf_filter_coeff403_a, rf_filter_coeff403_b,rf_filter_coeff404_a,
rf_filter_coeff404_b,rf_filter_coeff405_a, rf_filter_coeff405_b,rf_filter_coeff406_a,
rf_filter_coeff406_b,rf_filter_coeff407_a, rf_filter_coeff407_b,rf_filter_coeff408_a,
rf_filter_coeff408_b,rf_filter_coeff409_a, rf_filter_coeff409_b,rf_filter_coeff410_a,
rf_filter_coeff410_b,rf_filter_coeff411_a, rf_filter_coeff411_b,rf_filter_coeff412_a,
rf_filter_coeff412_b,rf_filter_coeff413_a, rf_filter_coeff413_b,rf_filter_coeff414_a,
rf_filter_coeff414_b,rf_filter_coeff415_a, rf_filter_coeff415_b,rf_filter_coeff416_a,
rf_filter_coeff416_b,rf_filter_coeff417_a, rf_filter_coeff417_b,rf_filter_coeff418_a,
rf_filter_coeff418_b,rf_filter_coeff419_a, rf_filter_coeff419_b,rf_filter_coeff420_a,
rf_filter_coeff420_b,rf_filter_coeff421_a, rf_filter_coeff421_b,rf_filter_coeff422_a,
rf_filter_coeff422_b,rf_filter_coeff423_a, rf_filter_coeff423_b,rf_filter_coeff424_a,
rf_filter_coeff424_b,rf_filter_coeff425_a, rf_filter_coeff425_b,rf_filter_coeff426_a,
rf_filter_coeff426_b,rf_filter_coeff427_a, rf_filter_coeff427_b,rf_filter_coeff428_a,
rf_filter_coeff428_b,rf_filter_coeff429_a, rf_filter_coeff429_b,rf_filter_coeff430_a,
rf_filter_coeff430_b,rf_filter_coeff431_a, rf_filter_coeff431_b,rf_filter_coeff432_a,
rf_filter_coeff432_b,rf_filter_coeff433_a, rf_filter_coeff433_b,rf_filter_coeff434_a,
rf_filter_coeff434_b,rf_filter_coeff435_a, rf_filter_coeff435_b,rf_filter_coeff436_a,
rf_filter_coeff436_b,rf_filter_coeff437_a, rf_filter_coeff437_b,rf_filter_coeff438_a,
rf_filter_coeff438_b,rf_filter_coeff439_a, rf_filter_coeff439_b,rf_filter_coeff440_a,
rf_filter_coeff440_b,rf_filter_coeff441_a, rf_filter_coeff441_b,rf_filter_coeff442_a,
rf_filter_coeff442_b,rf_filter_coeff443_a, rf_filter_coeff443_b,rf_filter_coeff444_a,
rf_filter_coeff444_b,rf_filter_coeff445_a, rf_filter_coeff445_b,rf_filter_coeff446_a,
rf_filter_coeff446_b,rf_filter_coeff447_a, rf_filter_coeff447_b,rf_filter_coeff448_a,
rf_filter_coeff448_b,rf_filter_coeff449_a, rf_filter_coeff449_b,rf_filter_coeff450_a,
rf_filter_coeff450_b,rf_filter_coeff451_a, rf_filter_coeff451_b,rf_filter_coeff452_a,
rf_filter_coeff452_b,rf_filter_coeff453_a, rf_filter_coeff453_b,rf_filter_coeff454_a,
rf_filter_coeff454_b,rf_filter_coeff455_a, rf_filter_coeff455_b,rf_filter_coeff456_a,
rf_filter_coeff456_b,rf_filter_coeff457_a, rf_filter_coeff457_b,rf_filter_coeff458_a,
rf_filter_coeff458_b,rf_filter_coeff459_a, rf_filter_coeff459_b,rf_filter_coeff460_a,
rf_filter_coeff460_b,rf_filter_coeff461_a, rf_filter_coeff461_b,rf_filter_coeff462_a,
rf_filter_coeff462_b,rf_filter_coeff463_a, rf_filter_coeff463_b,rf_filter_coeff464_a,
rf_filter_coeff464_b,rf_filter_coeff465_a, rf_filter_coeff465_b,rf_filter_coeff466_a,
rf_filter_coeff466_b,rf_filter_coeff467_a, rf_filter_coeff467_b,rf_filter_coeff468_a,
rf_filter_coeff468_b,rf_filter_coeff469_a, rf_filter_coeff469_b,rf_filter_coeff470_a,
rf_filter_coeff470_b,rf_filter_coeff471_a, rf_filter_coeff471_b,rf_filter_coeff472_a,
rf_filter_coeff472_b,rf_filter_coeff473_a, rf_filter_coeff473_b,rf_filter_coeff474_a,
rf_filter_coeff474_b,rf_filter_coeff475_a, rf_filter_coeff475_b,rf_filter_coeff476_a,
rf_filter_coeff476_b,rf_filter_coeff477_a, rf_filter_coeff477_b,rf_filter_coeff478_a,
rf_filter_coeff478_b,rf_filter_coeff479_a, rf_filter_coeff479_b,rf_filter_coeff480_a,
rf_filter_coeff480_b,rf_filter_coeff481_a, rf_filter_coeff481_b,rf_filter_coeff482_a,
rf_filter_coeff482_b,rf_filter_coeff483_a, rf_filter_coeff483_b,rf_filter_coeff484_a,
rf_filter_coeff484_b,rf_filter_coeff485_a, rf_filter_coeff485_b,rf_filter_coeff486_a,
rf_filter_coeff486_b,rf_filter_coeff487_a, rf_filter_coeff487_b,rf_filter_coeff488_a,
rf_filter_coeff488_b,rf_filter_coeff489_a, rf_filter_coeff489_b,rf_filter_coeff490_a,
rf_filter_coeff490_b,rf_filter_coeff491_a, rf_filter_coeff491_b,rf_filter_coeff492_a,
rf_filter_coeff492_b,rf_filter_coeff493_a, rf_filter_coeff493_b,rf_filter_coeff494_a,
rf_filter_coeff494_b,rf_filter_coeff495_a, rf_filter_coeff495_b,rf_filter_coeff496_a,
rf_filter_coeff496_b,rf_filter_coeff497_a, rf_filter_coeff497_b,rf_filter_coeff498_a,
rf_filter_coeff498_b,rf_filter_coeff499_a, rf_filter_coeff499_b,rf_filter_coeff500_a,
rf_filter_coeff500_b,rf_filter_coeff501_a, rf_filter_coeff501_b,rf_filter_coeff502_a,
rf_filter_coeff502_b,rf_filter_coeff503_a, rf_filter_coeff503_b,rf_filter_coeff504_a,
rf_filter_coeff504_b,rf_filter_coeff505_a, rf_filter_coeff505_b,rf_filter_coeff506_a,
rf_filter_coeff506_b,rf_filter_coeff507_a, rf_filter_coeff507_b,rf_filter_coeff508_a,
rf_filter_coeff508_b,rf_filter_coeff509_a, rf_filter_coeff509_b,rf_filter_coeff510_a,
rf_filter_coeff510_b,rf_filter_coeff511_a, rf_filter_coeff511_b
     );

    // FILTER INTERFACES
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------
    // Clock and Reset
    input                                       clk;
    input                                       rst_n;

    // Handshaking Signals (Input)
    input                                       aud_in_rts;
    output                                      aud_in_rtr;
```

```
    // Audio Data
    input                   [31:0]                  filt_input_data;
    output                  [31:0]                  filt_out_data;

    // Handshaking Signals (Output)
    output                                          aud_out_rts;
    input                                           aud_out_rtr;

    // Control Signals
    input                                           trig_filter_ovf_flag_clear;
    output                                          ro_filter_ovf_flag;
    input               [2:0]                   rf_filter_shift;
    input                                       rf_filter_clip_en;

    // Filter Coeffs
    input           [7:0]                   rf_filter_coeff0_a, rf_filter_coeff0_b,
    rf_filter_coeff1_a, rf_filter_coeff1_b,rf_filter_coeff2_a,
rf_filter_coeff2_b,rf_filter_coeff3_a, rf_filter_coeff3_b,rf_filter_coeff4_a, rf_filter_coeff4_b,
    rf_filter_coeff5_a, rf_filter_coeff5_b,rf_filter_coeff6_a,
rf_filter_coeff6_b,rf_filter_coeff7_a, rf_filter_coeff7_b,rf_filter_coeff8_a, rf_filter_coeff8_b,
    rf_filter_coeff9_a, rf_filter_coeff9_b,rf_filter_coeff10_a,
rf_filter_coeff10_b,rf_filter_coeff11_a, rf_filter_coeff11_b,rf_filter_coeff12_a,
rf_filter_coeff12_b,
    rf_filter_coeff13_a, rf_filter_coeff13_b,rf_filter_coeff14_a,
rf_filter_coeff14_b,rf_filter_coeff15_a, rf_filter_coeff15_b,rf_filter_coeff16_a,
rf_filter_coeff16_b,
    rf_filter_coeff17_a, rf_filter_coeff17_b,rf_filter_coeff18_a,
rf_filter_coeff18_b,rf_filter_coeff19_a, rf_filter_coeff19_b,rf_filter_coeff20_a,
rf_filter_coeff20_b,
    rf_filter_coeff21_a, rf_filter_coeff21_b,rf_filter_coeff22_a,
rf_filter_coeff22_b,rf_filter_coeff23_a, rf_filter_coeff23_b,rf_filter_coeff24_a,
rf_filter_coeff24_b,
    rf_filter_coeff25_a, rf_filter_coeff25_b,rf_filter_coeff26_a,
rf_filter_coeff26_b,rf_filter_coeff27_a, rf_filter_coeff27_b,rf_filter_coeff28_a,
rf_filter_coeff28_b,
    rf_filter_coeff29_a, rf_filter_coeff29_b,rf_filter_coeff30_a,
rf_filter_coeff30_b,rf_filter_coeff31_a, rf_filter_coeff31_b,rf_filter_coeff32_a,
rf_filter_coeff32_b,
    rf_filter_coeff33_a, rf_filter_coeff33_b,rf_filter_coeff34_a,
rf_filter_coeff34_b,rf_filter_coeff35_a, rf_filter_coeff35_b,rf_filter_coeff36_a,
rf_filter_coeff36_b,
    rf_filter_coeff37_a, rf_filter_coeff37_b,rf_filter_coeff38_a,
rf_filter_coeff38_b,rf_filter_coeff39_a, rf_filter_coeff39_b,rf_filter_coeff40_a,
rf_filter_coeff40_b,
    rf_filter_coeff41_a, rf_filter_coeff41_b,rf_filter_coeff42_a,
rf_filter_coeff42_b,rf_filter_coeff43_a, rf_filter_coeff43_b,rf_filter_coeff44_a,
rf_filter_coeff44_b,
    rf_filter_coeff45_a, rf_filter_coeff45_b,rf_filter_coeff46_a,
rf_filter_coeff46_b,rf_filter_coeff47_a, rf_filter_coeff47_b,rf_filter_coeff48_a,
rf_filter_coeff48_b,
    rf_filter_coeff49_a, rf_filter_coeff49_b,rf_filter_coeff50_a,
rf_filter_coeff50_b,rf_filter_coeff51_a, rf_filter_coeff51_b,rf_filter_coeff52_a,
rf_filter_coeff52_b,
    rf_filter_coeff53_a, rf_filter_coeff53_b,rf_filter_coeff54_a,
rf_filter_coeff54_b,rf_filter_coeff55_a, rf_filter_coeff55_b,rf_filter_coeff56_a,
rf_filter_coeff56_b,
    rf_filter_coeff57_a, rf_filter_coeff57_b,rf_filter_coeff58_a,
rf_filter_coeff58_b,rf_filter_coeff59_a, rf_filter_coeff59_b,rf_filter_coeff60_a,
rf_filter_coeff60_b,
    rf_filter_coeff61_a, rf_filter_coeff61_b,rf_filter_coeff62_a,
rf_filter_coeff62_b,rf_filter_coeff63_a, rf_filter_coeff63_b,rf_filter_coeff64_a,
rf_filter_coeff64_b,
    rf_filter_coeff65_a, rf_filter_coeff65_b,rf_filter_coeff66_a,
rf_filter_coeff66_b,rf_filter_coeff67_a, rf_filter_coeff67_b,rf_filter_coeff68_a,
rf_filter_coeff68_b,
    rf_filter_coeff69_a, rf_filter_coeff69_b,rf_filter_coeff70_a,
rf_filter_coeff70_b,rf_filter_coeff71_a, rf_filter_coeff71_b,rf_filter_coeff72_a,
rf_filter_coeff72_b,
```

```
    rf_filter_coeff73_a, rf_filter_coeff73_b,rf_filter_coeff74_a,
rf_filter_coeff74_b,rf_filter_coeff75_a, rf_filter_coeff75_b,rf_filter_coeff76_a,
rf_filter_coeff76_b,
    rf_filter_coeff77_a, rf_filter_coeff77_b,rf_filter_coeff78_a,
rf_filter_coeff78_b,rf_filter_coeff79_a, rf_filter_coeff79_b,rf_filter_coeff80_a,
rf_filter_coeff80_b,
    rf_filter_coeff81_a, rf_filter_coeff81_b,rf_filter_coeff82_a,
rf_filter_coeff82_b,rf_filter_coeff83_a, rf_filter_coeff83_b,rf_filter_coeff84_a,
rf_filter_coeff84_b,
    rf_filter_coeff85_a, rf_filter_coeff85_b,rf_filter_coeff86_a,
rf_filter_coeff86_b,rf_filter_coeff87_a, rf_filter_coeff87_b,rf_filter_coeff88_a,
rf_filter_coeff88_b,
    rf_filter_coeff89_a, rf_filter_coeff89_b,rf_filter_coeff90_a,
rf_filter_coeff90_b,rf_filter_coeff91_a, rf_filter_coeff91_b,rf_filter_coeff92_a,
rf_filter_coeff92_b,
    rf_filter_coeff93_a, rf_filter_coeff93_b,rf_filter_coeff94_a,
rf_filter_coeff94_b,rf_filter_coeff95_a, rf_filter_coeff95_b,rf_filter_coeff96_a,
rf_filter_coeff96_b,
    rf_filter_coeff97_a, rf_filter_coeff97_b,rf_filter_coeff98_a,
rf_filter_coeff98_b,rf_filter_coeff99_a, rf_filter_coeff99_b,rf_filter_coeff100_a,
rf_filter_coeff100_b,
    rf_filter_coeff101_a, rf_filter_coeff101_b,rf_filter_coeff102_a,
rf_filter_coeff102_b,rf_filter_coeff103_a, rf_filter_coeff103_b,rf_filter_coeff104_a,
rf_filter_coeff104_b,
    rf_filter_coeff105_a, rf_filter_coeff105_b,rf_filter_coeff106_a,
rf_filter_coeff106_b,rf_filter_coeff107_a, rf_filter_coeff107_b,rf_filter_coeff108_a,
rf_filter_coeff108_b,
    rf_filter_coeff109_a, rf_filter_coeff109_b,rf_filter_coeff110_a,
rf_filter_coeff110_b,rf_filter_coeff111_a, rf_filter_coeff111_b,rf_filter_coeff112_a,
rf_filter_coeff112_b,
    rf_filter_coeff113_a, rf_filter_coeff113_b,rf_filter_coeff114_a,
rf_filter_coeff114_b,rf_filter_coeff115_a, rf_filter_coeff115_b,rf_filter_coeff116_a,
rf_filter_coeff116_b,
    rf_filter_coeff117_a, rf_filter_coeff117_b,rf_filter_coeff118_a,
rf_filter_coeff118_b,rf_filter_coeff119_a, rf_filter_coeff119_b,rf_filter_coeff120_a,
rf_filter_coeff120_b,
    rf_filter_coeff121_a, rf_filter_coeff121_b,rf_filter_coeff122_a,
rf_filter_coeff122_b,rf_filter_coeff123_a, rf_filter_coeff123_b,rf_filter_coeff124_a,
rf_filter_coeff124_b,
    rf_filter_coeff125_a, rf_filter_coeff125_b,rf_filter_coeff126_a,
rf_filter_coeff126_b,rf_filter_coeff127_a, rf_filter_coeff127_b,rf_filter_coeff128_a,
rf_filter_coeff128_b,
    rf_filter_coeff129_a, rf_filter_coeff129_b,rf_filter_coeff130_a,
rf_filter_coeff130_b,rf_filter_coeff131_a, rf_filter_coeff131_b,rf_filter_coeff132_a,
rf_filter_coeff132_b,
    rf_filter_coeff133_a, rf_filter_coeff133_b,rf_filter_coeff134_a,
rf_filter_coeff134_b,rf_filter_coeff135_a, rf_filter_coeff135_b,rf_filter_coeff136_a,
rf_filter_coeff136_b,
    rf_filter_coeff137_a, rf_filter_coeff137_b,rf_filter_coeff138_a,
rf_filter_coeff138_b,rf_filter_coeff139_a, rf_filter_coeff139_b,rf_filter_coeff140_a,
rf_filter_coeff140_b,
    rf_filter_coeff141_a, rf_filter_coeff141_b,rf_filter_coeff142_a,
rf_filter_coeff142_b,rf_filter_coeff143_a, rf_filter_coeff143_b,rf_filter_coeff144_a,
rf_filter_coeff144_b,
    rf_filter_coeff145_a, rf_filter_coeff145_b,rf_filter_coeff146_a,
rf_filter_coeff146_b,rf_filter_coeff147_a, rf_filter_coeff147_b,rf_filter_coeff148_a,
rf_filter_coeff148_b,
    rf_filter_coeff149_a, rf_filter_coeff149_b,rf_filter_coeff150_a,
rf_filter_coeff150_b,rf_filter_coeff151_a, rf_filter_coeff151_b,rf_filter_coeff152_a,
rf_filter_coeff152_b,
    rf_filter_coeff153_a, rf_filter_coeff153_b,rf_filter_coeff154_a,
rf_filter_coeff154_b,rf_filter_coeff155_a, rf_filter_coeff155_b,rf_filter_coeff156_a,
rf_filter_coeff156_b,
    rf_filter_coeff157_a, rf_filter_coeff157_b,rf_filter_coeff158_a,
rf_filter_coeff158_b,rf_filter_coeff159_a, rf_filter_coeff159_b,rf_filter_coeff160_a,
rf_filter_coeff160_b,
    rf_filter_coeff161_a, rf_filter_coeff161_b,rf_filter_coeff162_a,
rf_filter_coeff162_b,rf_filter_coeff163_a, rf_filter_coeff163_b,rf_filter_coeff164_a,
rf_filter_coeff164_b,
```

```
    rf_filter_coeff165_a, rf_filter_coeff165_b,rf_filter_coeff166_a,
rf_filter_coeff166_b,rf_filter_coeff167_a, rf_filter_coeff167_b,rf_filter_coeff168_a,
rf_filter_coeff168_b,
    rf_filter_coeff169_a, rf_filter_coeff169_b,rf_filter_coeff170_a,
rf_filter_coeff170_b,rf_filter_coeff171_a, rf_filter_coeff171_b,rf_filter_coeff172_a,
rf_filter_coeff172_b,
    rf_filter_coeff173_a, rf_filter_coeff173_b,rf_filter_coeff174_a,
rf_filter_coeff174_b,rf_filter_coeff175_a, rf_filter_coeff175_b,rf_filter_coeff176_a,
rf_filter_coeff176_b,
    rf_filter_coeff177_a, rf_filter_coeff177_b,rf_filter_coeff178_a,
rf_filter_coeff178_b,rf_filter_coeff179_a, rf_filter_coeff179_b,rf_filter_coeff180_a,
rf_filter_coeff180_b,
    rf_filter_coeff181_a, rf_filter_coeff181_b,rf_filter_coeff182_a,
rf_filter_coeff182_b,rf_filter_coeff183_a, rf_filter_coeff183_b,rf_filter_coeff184_a,
rf_filter_coeff184_b,
    rf_filter_coeff185_a, rf_filter_coeff185_b,rf_filter_coeff186_a,
rf_filter_coeff186_b,rf_filter_coeff187_a, rf_filter_coeff187_b,rf_filter_coeff188_a,
rf_filter_coeff188_b,
    rf_filter_coeff189_a, rf_filter_coeff189_b,rf_filter_coeff190_a,
rf_filter_coeff190_b,rf_filter_coeff191_a, rf_filter_coeff191_b,rf_filter_coeff192_a,
rf_filter_coeff192_b,
    rf_filter_coeff193_a, rf_filter_coeff193_b,rf_filter_coeff194_a,
rf_filter_coeff194_b,rf_filter_coeff195_a, rf_filter_coeff195_b,rf_filter_coeff196_a,
rf_filter_coeff196_b,
    rf_filter_coeff197_a, rf_filter_coeff197_b,rf_filter_coeff198_a,
rf_filter_coeff198_b,rf_filter_coeff199_a, rf_filter_coeff199_b,rf_filter_coeff200_a,
rf_filter_coeff200_b,
    rf_filter_coeff201_a, rf_filter_coeff201_b,rf_filter_coeff202_a,
rf_filter_coeff202_b,rf_filter_coeff203_a, rf_filter_coeff203_b,rf_filter_coeff204_a,
rf_filter_coeff204_b,
    rf_filter_coeff205_a, rf_filter_coeff205_b,rf_filter_coeff206_a,
rf_filter_coeff206_b,rf_filter_coeff207_a, rf_filter_coeff207_b,rf_filter_coeff208_a,
rf_filter_coeff208_b,
    rf_filter_coeff209_a, rf_filter_coeff209_b,rf_filter_coeff210_a,
rf_filter_coeff210_b,rf_filter_coeff211_a, rf_filter_coeff211_b,rf_filter_coeff212_a,
rf_filter_coeff212_b,
    rf_filter_coeff213_a, rf_filter_coeff213_b,rf_filter_coeff214_a,
rf_filter_coeff214_b,rf_filter_coeff215_a, rf_filter_coeff215_b,rf_filter_coeff216_a,
rf_filter_coeff216_b,
    rf_filter_coeff217_a, rf_filter_coeff217_b,rf_filter_coeff218_a,
rf_filter_coeff218_b,rf_filter_coeff219_a, rf_filter_coeff219_b,rf_filter_coeff220_a,
rf_filter_coeff220_b,
    rf_filter_coeff221_a, rf_filter_coeff221_b,rf_filter_coeff222_a,
rf_filter_coeff222_b,rf_filter_coeff223_a, rf_filter_coeff223_b,rf_filter_coeff224_a,
rf_filter_coeff224_b,
    rf_filter_coeff225_a, rf_filter_coeff225_b,rf_filter_coeff226_a,
rf_filter_coeff226_b,rf_filter_coeff227_a, rf_filter_coeff227_b,rf_filter_coeff228_a,
rf_filter_coeff228_b,
    rf_filter_coeff229_a, rf_filter_coeff229_b,rf_filter_coeff230_a,
rf_filter_coeff230_b,rf_filter_coeff231_a, rf_filter_coeff231_b,rf_filter_coeff232_a,
rf_filter_coeff232_b,
    rf_filter_coeff233_a, rf_filter_coeff233_b,rf_filter_coeff234_a,
rf_filter_coeff234_b,rf_filter_coeff235_a, rf_filter_coeff235_b,rf_filter_coeff236_a,
rf_filter_coeff236_b,
    rf_filter_coeff237_a, rf_filter_coeff237_b,rf_filter_coeff238_a,
rf_filter_coeff238_b,rf_filter_coeff239_a, rf_filter_coeff239_b,rf_filter_coeff240_a,
rf_filter_coeff240_b,
    rf_filter_coeff241_a, rf_filter_coeff241_b,rf_filter_coeff242_a,
rf_filter_coeff242_b,rf_filter_coeff243_a, rf_filter_coeff243_b,rf_filter_coeff244_a,
rf_filter_coeff244_b,
    rf_filter_coeff245_a, rf_filter_coeff245_b,rf_filter_coeff246_a,
rf_filter_coeff246_b,rf_filter_coeff247_a, rf_filter_coeff247_b,rf_filter_coeff248_a,
rf_filter_coeff248_b,
    rf_filter_coeff249_a, rf_filter_coeff249_b,rf_filter_coeff250_a,
rf_filter_coeff250_b,rf_filter_coeff251_a, rf_filter_coeff251_b,rf_filter_coeff252_a,
rf_filter_coeff252_b,
    rf_filter_coeff253_a, rf_filter_coeff253_b,rf_filter_coeff254_a,
rf_filter_coeff254_b,rf_filter_coeff255_a, rf_filter_coeff255_b,rf_filter_coeff256_a,
rf_filter_coeff256_b,
```

```
    rf_filter_coeff257_a, rf_filter_coeff257_b,rf_filter_coeff258_a,
rf_filter_coeff258_b,rf_filter_coeff259_a, rf_filter_coeff259_b,rf_filter_coeff260_a,
rf_filter_coeff260_b,
    rf_filter_coeff261_a, rf_filter_coeff261_b,rf_filter_coeff262_a,
rf_filter_coeff262_b,rf_filter_coeff263_a, rf_filter_coeff263_b,rf_filter_coeff264_a,
rf_filter_coeff264_b,
    rf_filter_coeff265_a, rf_filter_coeff265_b,rf_filter_coeff266_a,
rf_filter_coeff266_b,rf_filter_coeff267_a, rf_filter_coeff267_b,rf_filter_coeff268_a,
rf_filter_coeff268_b,
    rf_filter_coeff269_a, rf_filter_coeff269_b,rf_filter_coeff270_a,
rf_filter_coeff270_b,rf_filter_coeff271_a, rf_filter_coeff271_b,rf_filter_coeff272_a,
rf_filter_coeff272_b,
    rf_filter_coeff273_a, rf_filter_coeff273_b,rf_filter_coeff274_a,
rf_filter_coeff274_b,rf_filter_coeff275_a, rf_filter_coeff275_b,rf_filter_coeff276_a,
rf_filter_coeff276_b,
    rf_filter_coeff277_a, rf_filter_coeff277_b,rf_filter_coeff278_a,
rf_filter_coeff278_b,rf_filter_coeff279_a, rf_filter_coeff279_b,rf_filter_coeff280_a,
rf_filter_coeff280_b,
    rf_filter_coeff281_a, rf_filter_coeff281_b,rf_filter_coeff282_a,
rf_filter_coeff282_b,rf_filter_coeff283_a, rf_filter_coeff283_b,rf_filter_coeff284_a,
rf_filter_coeff284_b,
    rf_filter_coeff285_a, rf_filter_coeff285_b,rf_filter_coeff286_a,
rf_filter_coeff286_b,rf_filter_coeff287_a, rf_filter_coeff287_b,rf_filter_coeff288_a,
rf_filter_coeff288_b,
    rf_filter_coeff289_a, rf_filter_coeff289_b,rf_filter_coeff290_a,
rf_filter_coeff290_b,rf_filter_coeff291_a, rf_filter_coeff291_b,rf_filter_coeff292_a,
rf_filter_coeff292_b,
    rf_filter_coeff293_a, rf_filter_coeff293_b,rf_filter_coeff294_a,
rf_filter_coeff294_b,rf_filter_coeff295_a, rf_filter_coeff295_b,rf_filter_coeff296_a,
rf_filter_coeff296_b,
    rf_filter_coeff297_a, rf_filter_coeff297_b,rf_filter_coeff298_a,
rf_filter_coeff298_b,rf_filter_coeff299_a, rf_filter_coeff299_b,rf_filter_coeff300_a,
rf_filter_coeff300_b,
    rf_filter_coeff301_a, rf_filter_coeff301_b,rf_filter_coeff302_a,
rf_filter_coeff302_b,rf_filter_coeff303_a, rf_filter_coeff303_b,rf_filter_coeff304_a,
rf_filter_coeff304_b,
    rf_filter_coeff305_a, rf_filter_coeff305_b,rf_filter_coeff306_a,
rf_filter_coeff306_b,rf_filter_coeff307_a, rf_filter_coeff307_b,rf_filter_coeff308_a,
rf_filter_coeff308_b,
    rf_filter_coeff309_a, rf_filter_coeff309_b,rf_filter_coeff310_a,
rf_filter_coeff310_b,rf_filter_coeff311_a, rf_filter_coeff311_b,rf_filter_coeff312_a,
rf_filter_coeff312_b,
    rf_filter_coeff313_a, rf_filter_coeff313_b,rf_filter_coeff314_a,
rf_filter_coeff314_b,rf_filter_coeff315_a, rf_filter_coeff315_b,rf_filter_coeff316_a,
rf_filter_coeff316_b,
    rf_filter_coeff317_a, rf_filter_coeff317_b,rf_filter_coeff318_a,
rf_filter_coeff318_b,rf_filter_coeff319_a, rf_filter_coeff319_b,rf_filter_coeff320_a,
rf_filter_coeff320_b,
    rf_filter_coeff321_a, rf_filter_coeff321_b,rf_filter_coeff322_a,
rf_filter_coeff322_b,rf_filter_coeff323_a, rf_filter_coeff323_b,rf_filter_coeff324_a,
rf_filter_coeff324_b,
    rf_filter_coeff325_a, rf_filter_coeff325_b,rf_filter_coeff326_a,
rf_filter_coeff326_b,rf_filter_coeff327_a, rf_filter_coeff327_b,rf_filter_coeff328_a,
rf_filter_coeff328_b,
    rf_filter_coeff329_a, rf_filter_coeff329_b,rf_filter_coeff330_a,
rf_filter_coeff330_b,rf_filter_coeff331_a, rf_filter_coeff331_b,rf_filter_coeff332_a,
rf_filter_coeff332_b,
    rf_filter_coeff333_a, rf_filter_coeff333_b,rf_filter_coeff334_a,
rf_filter_coeff334_b,rf_filter_coeff335_a, rf_filter_coeff335_b,rf_filter_coeff336_a,
rf_filter_coeff336_b,
    rf_filter_coeff337_a, rf_filter_coeff337_b,rf_filter_coeff338_a,
rf_filter_coeff338_b,rf_filter_coeff339_a, rf_filter_coeff339_b,rf_filter_coeff340_a,
rf_filter_coeff340_b,
    rf_filter_coeff341_a, rf_filter_coeff341_b,rf_filter_coeff342_a,
rf_filter_coeff342_b,rf_filter_coeff343_a, rf_filter_coeff343_b,rf_filter_coeff344_a,
rf_filter_coeff344_b,
    rf_filter_coeff345_a, rf_filter_coeff345_b,rf_filter_coeff346_a,
rf_filter_coeff346_b,rf_filter_coeff347_a, rf_filter_coeff347_b,rf_filter_coeff348_a,
rf_filter_coeff348_b,
```

```
    rf_filter_coeff349_a, rf_filter_coeff349_b,rf_filter_coeff350_a,
rf_filter_coeff350_b,rf_filter_coeff351_a, rf_filter_coeff351_b,rf_filter_coeff352_a,
rf_filter_coeff352_b,
    rf_filter_coeff353_a, rf_filter_coeff353_b,rf_filter_coeff354_a,
rf_filter_coeff354_b,rf_filter_coeff355_a, rf_filter_coeff355_b,rf_filter_coeff356_a,
rf_filter_coeff356_b,
    rf_filter_coeff357_a, rf_filter_coeff357_b,rf_filter_coeff358_a,
rf_filter_coeff358_b,rf_filter_coeff359_a, rf_filter_coeff359_b,rf_filter_coeff360_a,
rf_filter_coeff360_b,
    rf_filter_coeff361_a, rf_filter_coeff361_b,rf_filter_coeff362_a,
rf_filter_coeff362_b,rf_filter_coeff363_a, rf_filter_coeff363_b,rf_filter_coeff364_a,
rf_filter_coeff364_b,
    rf_filter_coeff365_a, rf_filter_coeff365_b,rf_filter_coeff366_a,
rf_filter_coeff366_b,rf_filter_coeff367_a, rf_filter_coeff367_b,rf_filter_coeff368_a,
rf_filter_coeff368_b,
    rf_filter_coeff369_a, rf_filter_coeff369_b,rf_filter_coeff370_a,
rf_filter_coeff370_b,rf_filter_coeff371_a, rf_filter_coeff371_b,rf_filter_coeff372_a,
rf_filter_coeff372_b,
    rf_filter_coeff373_a, rf_filter_coeff373_b,rf_filter_coeff374_a,
rf_filter_coeff374_b,rf_filter_coeff375_a, rf_filter_coeff375_b,rf_filter_coeff376_a,
rf_filter_coeff376_b,
    rf_filter_coeff377_a, rf_filter_coeff377_b,rf_filter_coeff378_a,
rf_filter_coeff378_b,rf_filter_coeff379_a, rf_filter_coeff379_b,rf_filter_coeff380_a,
rf_filter_coeff380_b,
    rf_filter_coeff381_a, rf_filter_coeff381_b,rf_filter_coeff382_a,
rf_filter_coeff382_b,rf_filter_coeff383_a, rf_filter_coeff383_b,rf_filter_coeff384_a,
rf_filter_coeff384_b,
    rf_filter_coeff385_a, rf_filter_coeff385_b,rf_filter_coeff386_a,
rf_filter_coeff386_b,rf_filter_coeff387_a, rf_filter_coeff387_b,rf_filter_coeff388_a,
rf_filter_coeff388_b,
    rf_filter_coeff389_a, rf_filter_coeff389_b,rf_filter_coeff390_a,
rf_filter_coeff390_b,rf_filter_coeff391_a, rf_filter_coeff391_b,rf_filter_coeff392_a,
rf_filter_coeff392_b,
    rf_filter_coeff393_a, rf_filter_coeff393_b,rf_filter_coeff394_a,
rf_filter_coeff394_b,rf_filter_coeff395_a, rf_filter_coeff395_b,rf_filter_coeff396_a,
rf_filter_coeff396_b,
    rf_filter_coeff397_a, rf_filter_coeff397_b,rf_filter_coeff398_a,
rf_filter_coeff398_b,rf_filter_coeff399_a, rf_filter_coeff399_b,rf_filter_coeff400_a,
rf_filter_coeff400_b,
    rf_filter_coeff401_a, rf_filter_coeff401_b,rf_filter_coeff402_a,
rf_filter_coeff402_b,rf_filter_coeff403_a, rf_filter_coeff403_b,rf_filter_coeff404_a,
rf_filter_coeff404_b,
    rf_filter_coeff405_a, rf_filter_coeff405_b,rf_filter_coeff406_a,
rf_filter_coeff406_b,rf_filter_coeff407_a, rf_filter_coeff407_b,rf_filter_coeff408_a,
rf_filter_coeff408_b,
    rf_filter_coeff409_a, rf_filter_coeff409_b,rf_filter_coeff410_a,
rf_filter_coeff410_b,rf_filter_coeff411_a, rf_filter_coeff411_b,rf_filter_coeff412_a,
rf_filter_coeff412_b,
    rf_filter_coeff413_a, rf_filter_coeff413_b,rf_filter_coeff414_a,
rf_filter_coeff414_b,rf_filter_coeff415_a, rf_filter_coeff415_b,rf_filter_coeff416_a,
rf_filter_coeff416_b,
    rf_filter_coeff417_a, rf_filter_coeff417_b,rf_filter_coeff418_a,
rf_filter_coeff418_b,rf_filter_coeff419_a, rf_filter_coeff419_b,rf_filter_coeff420_a,
rf_filter_coeff420_b,
    rf_filter_coeff421_a, rf_filter_coeff421_b,rf_filter_coeff422_a,
rf_filter_coeff422_b,rf_filter_coeff423_a, rf_filter_coeff423_b,rf_filter_coeff424_a,
rf_filter_coeff424_b,
    rf_filter_coeff425_a, rf_filter_coeff425_b,rf_filter_coeff426_a,
rf_filter_coeff426_b,rf_filter_coeff427_a, rf_filter_coeff427_b,rf_filter_coeff428_a,
rf_filter_coeff428_b,
    rf_filter_coeff429_a, rf_filter_coeff429_b,rf_filter_coeff430_a,
rf_filter_coeff430_b,rf_filter_coeff431_a, rf_filter_coeff431_b,rf_filter_coeff432_a,
rf_filter_coeff432_b,
    rf_filter_coeff433_a, rf_filter_coeff433_b,rf_filter_coeff434_a,
rf_filter_coeff434_b,rf_filter_coeff435_a, rf_filter_coeff435_b,rf_filter_coeff436_a,
rf_filter_coeff436_b,
    rf_filter_coeff437_a, rf_filter_coeff437_b,rf_filter_coeff438_a,
rf_filter_coeff438_b,rf_filter_coeff439_a, rf_filter_coeff439_b,rf_filter_coeff440_a,
rf_filter_coeff440_b,
```

```
    rf_filter_coeff441_a, rf_filter_coeff441_b,rf_filter_coeff442_a,
rf_filter_coeff442_b,rf_filter_coeff443_a, rf_filter_coeff443_b,rf_filter_coeff444_a,
rf_filter_coeff444_b,
    rf_filter_coeff445_a, rf_filter_coeff445_b,rf_filter_coeff446_a,
rf_filter_coeff446_b,rf_filter_coeff447_a, rf_filter_coeff447_b,rf_filter_coeff448_a,
rf_filter_coeff448_b,
    rf_filter_coeff449_a, rf_filter_coeff449_b,rf_filter_coeff450_a,
rf_filter_coeff450_b,rf_filter_coeff451_a, rf_filter_coeff451_b,rf_filter_coeff452_a,
rf_filter_coeff452_b,
    rf_filter_coeff453_a, rf_filter_coeff453_b,rf_filter_coeff454_a,
rf_filter_coeff454_b,rf_filter_coeff455_a, rf_filter_coeff455_b,rf_filter_coeff456_a,
rf_filter_coeff456_b,
    rf_filter_coeff457_a, rf_filter_coeff457_b,rf_filter_coeff458_a,
rf_filter_coeff458_b,rf_filter_coeff459_a, rf_filter_coeff459_b,rf_filter_coeff460_a,
rf_filter_coeff460_b,
    rf_filter_coeff461_a, rf_filter_coeff461_b,rf_filter_coeff462_a,
rf_filter_coeff462_b,rf_filter_coeff463_a, rf_filter_coeff463_b,rf_filter_coeff464_a,
rf_filter_coeff464_b,
    rf_filter_coeff465_a, rf_filter_coeff465_b,rf_filter_coeff466_a,
rf_filter_coeff466_b,rf_filter_coeff467_a, rf_filter_coeff467_b,rf_filter_coeff468_a,
rf_filter_coeff468_b,
    rf_filter_coeff469_a, rf_filter_coeff469_b,rf_filter_coeff470_a,
rf_filter_coeff470_b,rf_filter_coeff471_a, rf_filter_coeff471_b,rf_filter_coeff472_a,
rf_filter_coeff472_b,
    rf_filter_coeff473_a, rf_filter_coeff473_b,rf_filter_coeff474_a,
rf_filter_coeff474_b,rf_filter_coeff475_a, rf_filter_coeff475_b,rf_filter_coeff476_a,
rf_filter_coeff476_b,
    rf_filter_coeff477_a, rf_filter_coeff477_b,rf_filter_coeff478_a,
rf_filter_coeff478_b,rf_filter_coeff479_a, rf_filter_coeff479_b,rf_filter_coeff480_a,
rf_filter_coeff480_b,
    rf_filter_coeff481_a, rf_filter_coeff481_b,rf_filter_coeff482_a,
rf_filter_coeff482_b,rf_filter_coeff483_a, rf_filter_coeff483_b,rf_filter_coeff484_a,
rf_filter_coeff484_b,
    rf_filter_coeff485_a, rf_filter_coeff485_b,rf_filter_coeff486_a,
rf_filter_coeff486_b,rf_filter_coeff487_a, rf_filter_coeff487_b,rf_filter_coeff488_a,
rf_filter_coeff488_b,
    rf_filter_coeff489_a, rf_filter_coeff489_b,rf_filter_coeff490_a,
rf_filter_coeff490_b,rf_filter_coeff491_a, rf_filter_coeff491_b,rf_filter_coeff492_a,
rf_filter_coeff492_b,
    rf_filter_coeff493_a, rf_filter_coeff493_b,rf_filter_coeff494_a,
rf_filter_coeff494_b,rf_filter_coeff495_a, rf_filter_coeff495_b,rf_filter_coeff496_a,
rf_filter_coeff496_b,
    rf_filter_coeff497_a, rf_filter_coeff497_b,rf_filter_coeff498_a,
rf_filter_coeff498_b,rf_filter_coeff499_a, rf_filter_coeff499_b,rf_filter_coeff500_a,
rf_filter_coeff500_b,
    rf_filter_coeff501_a, rf_filter_coeff501_b,rf_filter_coeff502_a,
rf_filter_coeff502_b,rf_filter_coeff503_a, rf_filter_coeff503_b,rf_filter_coeff504_a,
rf_filter_coeff504_b,
    rf_filter_coeff505_a, rf_filter_coeff505_b,rf_filter_coeff506_a,
rf_filter_coeff506_b,rf_filter_coeff507_a, rf_filter_coeff507_b,rf_filter_coeff508_a,
rf_filter_coeff508_b,
    rf_filter_coeff509_a, rf_filter_coeff509_b,rf_filter_coeff510_a,
rf_filter_coeff510_b,rf_filter_coeff511_a, rf_filter_coeff511_b;
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------

    // SUBMODULE CONNECTIONS
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------
    wire                                do_transfer;
    wire                                do_multiply_1st;
    wire                                do_multiply;
    wire            [15:0]      rf_filter_coeff;
    wire            [8:0]       mux_rdptr;
    wire                                mux_re;

    wire signed             [31:0]      accumulator_in_left;
    wire signed             [31:0]      accumulator_in_right;
    wire                                      accumulator_load;
    wire                                      accumulator_enable;
```

```
    wire signed              [39:0]      accumulator_out_left;
    wire signed              [39:0]      accumulator_out_right;

    wire signed              [15:0]      filter_out_left;
    wire signed              [15:0]      filter_out_right;
    wire                                       ro_filter_ovf_flag_left;
    wire                                       ro_filter_ovf_flag_right;
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------

    assign ro_filter_ovf_flag                   = ro_filter_ovf_flag_left ||
ro_filter_ovf_flag_right;
    assign filt_out_data                        = {filter_out_left,filter_out_right};


    //*****************************************************************************
    //                                FILTER STM
    //*****************************************************************************
    filter_stm TOP_FILTER_STM (
        .clk(clk),
        .rst_n(rst_n),
        .filter_aud_in_rts(aud_in_rts),
        .filter_aud_in_rtr(aud_in_rtr),
        .filter_aud_out_rts(aud_out_rts),
        .filter_aud_out_rtr(aud_out_rtr),
        .filter_aud_in(filt_input_data),
        .accumulator_load(accumulator_load),
        .accumulator_enable(accumulator_enable),
        .accumulator_in_left(accumulator_in_left),
        .accumulator_in_right(accumulator_in_right),
        .rf_filter_coeff(rf_filter_coeff),
        .mux_re(mux_re),
        .mux_rdptr(mux_rdptr)
        );

    //*****************************************************************************


    //*****************************************************************************
    //                               FILTER ACCUMULATOR LEFT
    //*****************************************************************************
    filter_accumulator TOP_FILTER_ACCUMULATOR_LEFT (
        .clk(clk),
        .rst_n(rst_n),
        .enable(accumulator_enable),
        .load(accumulator_load),
        .D(accumulator_in_left),
        .Q(accumulator_out_left)
        );

    //*****************************************************************************

    //*****************************************************************************
    //                               FILTER ACCUMULATOR RIGHT
    //*****************************************************************************
    filter_accumulator TOP_FILTER_ACCUMULATOR_RIGHT (
        .clk(clk),
        .rst_n(rst_n),
        .enable(accumulator_enable),
        .load(accumulator_load),
        .D(accumulator_in_right),
        .Q(accumulator_out_right)
        );

    //*****************************************************************************

    //*****************************************************************************
    //                               FILTER ROUND TRUNCATE LEFT
    //*****************************************************************************
    filter_round_truncate TOP_FILTER_ROUND_TRUNCATE_LEFT (
        .clk(clk),
```

```
    .rst_n(rst_n),
    .acc_in(accumulator_out_left),
    .rf_sat(rf_filter_clip_en),
    .rf_shift(rf_filter_shift),
    .trig_filter_ovf_flag_clear(trig_filter_ovf_flag_clear),
    .filter_out(filter_out_left),
    .ro_filter_ovf_flag(ro_filter_ovf_flag_left)
    );

//*******************************************************************************

//*******************************************************************************
//                                FILTER ROUND TRUNCATE RIGHT
//*******************************************************************************
filter_round_truncate TOP_FILTER_ROUND_TRUNCATE_RIGHT (
    .clk(clk),
    .rst_n(rst_n),
    .acc_in(accumulator_out_right),
    .rf_sat(rf_filter_clip_en),
    .rf_shift(rf_filter_shift),
    .trig_filter_ovf_flag_clear(trig_filter_ovf_flag_clear),
    .filter_out(filter_out_right),
    .ro_filter_ovf_flag(ro_filter_ovf_flag_right)
    );

//*******************************************************************************


//*******************************************************************************
//                                FILTER MUX
//*******************************************************************************
filter_mux TOP_FILTER_MUX (
    .clk(clk),
    .rden(mux_re),
    .rdptr(mux_rdptr),
    .rddata(rf_filter_coeff),
    .rf_filter_coeff0_a(rf_filter_coeff0_a),
    .rf_filter_coeff0_b(rf_filter_coeff0_b),
    .rf_filter_coeff1_a(rf_filter_coeff1_a),
    .rf_filter_coeff1_b(rf_filter_coeff1_b),
    .rf_filter_coeff2_a(rf_filter_coeff2_a),
    .rf_filter_coeff2_b(rf_filter_coeff2_b),
    .rf_filter_coeff3_a(rf_filter_coeff3_a),
    .rf_filter_coeff3_b(rf_filter_coeff3_b),
    .rf_filter_coeff4_a(rf_filter_coeff4_a),
    .rf_filter_coeff4_b(rf_filter_coeff4_b),
    .rf_filter_coeff5_a(rf_filter_coeff5_a),
    .rf_filter_coeff5_b(rf_filter_coeff5_b),
    .rf_filter_coeff6_a(rf_filter_coeff6_a),
    .rf_filter_coeff6_b(rf_filter_coeff6_b),
    .rf_filter_coeff7_a(rf_filter_coeff7_a),
    .rf_filter_coeff7_b(rf_filter_coeff7_b),
    .rf_filter_coeff8_a(rf_filter_coeff8_a),
    .rf_filter_coeff8_b(rf_filter_coeff8_b),
    .rf_filter_coeff9_a(rf_filter_coeff9_a),
    .rf_filter_coeff9_b(rf_filter_coeff9_b),
    .rf_filter_coeff10_a(rf_filter_coeff10_a),
    .rf_filter_coeff10_b(rf_filter_coeff10_b),
    .rf_filter_coeff11_a(rf_filter_coeff11_a),
    .rf_filter_coeff11_b(rf_filter_coeff11_b),
    .rf_filter_coeff12_a(rf_filter_coeff12_a),
    .rf_filter_coeff12_b(rf_filter_coeff12_b),
    .rf_filter_coeff13_a(rf_filter_coeff13_a),
    .rf_filter_coeff13_b(rf_filter_coeff13_b),
    .rf_filter_coeff14_a(rf_filter_coeff14_a),
    .rf_filter_coeff14_b(rf_filter_coeff14_b),
    .rf_filter_coeff15_a(rf_filter_coeff15_a),
    .rf_filter_coeff15_b(rf_filter_coeff15_b),
    .rf_filter_coeff16_a(rf_filter_coeff16_a),
    .rf_filter_coeff16_b(rf_filter_coeff16_b),
    .rf_filter_coeff17_a(rf_filter_coeff17_a),
```

```
.rf_filter_coeff17_b(rf_filter_coeff17_b),
.rf_filter_coeff18_a(rf_filter_coeff18_a),
.rf_filter_coeff18_b(rf_filter_coeff18_b),
.rf_filter_coeff19_a(rf_filter_coeff19_a),
.rf_filter_coeff19_b(rf_filter_coeff19_b),
.rf_filter_coeff20_a(rf_filter_coeff20_a),
.rf_filter_coeff20_b(rf_filter_coeff20_b),
.rf_filter_coeff21_a(rf_filter_coeff21_a),
.rf_filter_coeff21_b(rf_filter_coeff21_b),
.rf_filter_coeff22_a(rf_filter_coeff22_a),
.rf_filter_coeff22_b(rf_filter_coeff22_b),
.rf_filter_coeff23_a(rf_filter_coeff23_a),
.rf_filter_coeff23_b(rf_filter_coeff23_b),
.rf_filter_coeff24_a(rf_filter_coeff24_a),
.rf_filter_coeff24_b(rf_filter_coeff24_b),
.rf_filter_coeff25_a(rf_filter_coeff25_a),
.rf_filter_coeff25_b(rf_filter_coeff25_b),
.rf_filter_coeff26_a(rf_filter_coeff26_a),
.rf_filter_coeff26_b(rf_filter_coeff26_b),
.rf_filter_coeff27_a(rf_filter_coeff27_a),
.rf_filter_coeff27_b(rf_filter_coeff27_b),
.rf_filter_coeff28_a(rf_filter_coeff28_a),
.rf_filter_coeff28_b(rf_filter_coeff28_b),
.rf_filter_coeff29_a(rf_filter_coeff29_a),
.rf_filter_coeff29_b(rf_filter_coeff29_b),
.rf_filter_coeff30_a(rf_filter_coeff30_a),
.rf_filter_coeff30_b(rf_filter_coeff30_b),
.rf_filter_coeff31_a(rf_filter_coeff31_a),
.rf_filter_coeff31_b(rf_filter_coeff31_b),
.rf_filter_coeff32_a(rf_filter_coeff32_a),
.rf_filter_coeff32_b(rf_filter_coeff32_b),
.rf_filter_coeff33_a(rf_filter_coeff33_a),
.rf_filter_coeff33_b(rf_filter_coeff33_b),
.rf_filter_coeff34_a(rf_filter_coeff34_a),
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
```

```
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
```

```
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
```

```
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
```

```
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
```

```
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
```

```
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
```

```
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
```

```
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
```

```
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
```

```
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
```

```
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
```

```
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
```

```
        .rf_filter_coeff479_a(rf_filter_coeff479_a),
        .rf_filter_coeff479_b(rf_filter_coeff479_b),
        .rf_filter_coeff480_a(rf_filter_coeff480_a),
        .rf_filter_coeff480_b(rf_filter_coeff480_b),
        .rf_filter_coeff481_a(rf_filter_coeff481_a),
        .rf_filter_coeff481_b(rf_filter_coeff481_b),
        .rf_filter_coeff482_a(rf_filter_coeff482_a),
        .rf_filter_coeff482_b(rf_filter_coeff482_b),
        .rf_filter_coeff483_a(rf_filter_coeff483_a),
        .rf_filter_coeff483_b(rf_filter_coeff483_b),
        .rf_filter_coeff484_a(rf_filter_coeff484_a),
        .rf_filter_coeff484_b(rf_filter_coeff484_b),
        .rf_filter_coeff485_a(rf_filter_coeff485_a),
        .rf_filter_coeff485_b(rf_filter_coeff485_b),
        .rf_filter_coeff486_a(rf_filter_coeff486_a),
        .rf_filter_coeff486_b(rf_filter_coeff486_b),
        .rf_filter_coeff487_a(rf_filter_coeff487_a),
        .rf_filter_coeff487_b(rf_filter_coeff487_b),
        .rf_filter_coeff488_a(rf_filter_coeff488_a),
        .rf_filter_coeff488_b(rf_filter_coeff488_b),
        .rf_filter_coeff489_a(rf_filter_coeff489_a),
        .rf_filter_coeff489_b(rf_filter_coeff489_b),
        .rf_filter_coeff490_a(rf_filter_coeff490_a),
        .rf_filter_coeff490_b(rf_filter_coeff490_b),
        .rf_filter_coeff491_a(rf_filter_coeff491_a),
        .rf_filter_coeff491_b(rf_filter_coeff491_b),
        .rf_filter_coeff492_a(rf_filter_coeff492_a),
        .rf_filter_coeff492_b(rf_filter_coeff492_b),
        .rf_filter_coeff493_a(rf_filter_coeff493_a),
        .rf_filter_coeff493_b(rf_filter_coeff493_b),
        .rf_filter_coeff494_a(rf_filter_coeff494_a),
        .rf_filter_coeff494_b(rf_filter_coeff494_b),
        .rf_filter_coeff495_a(rf_filter_coeff495_a),
        .rf_filter_coeff495_b(rf_filter_coeff495_b),
        .rf_filter_coeff496_a(rf_filter_coeff496_a),
        .rf_filter_coeff496_b(rf_filter_coeff496_b),
        .rf_filter_coeff497_a(rf_filter_coeff497_a),
        .rf_filter_coeff497_b(rf_filter_coeff497_b),
        .rf_filter_coeff498_a(rf_filter_coeff498_a),
        .rf_filter_coeff498_b(rf_filter_coeff498_b),
        .rf_filter_coeff499_a(rf_filter_coeff499_a),
        .rf_filter_coeff499_b(rf_filter_coeff499_b),
        .rf_filter_coeff500_a(rf_filter_coeff500_a),
        .rf_filter_coeff500_b(rf_filter_coeff500_b),
        .rf_filter_coeff501_a(rf_filter_coeff501_a),
        .rf_filter_coeff501_b(rf_filter_coeff501_b),
        .rf_filter_coeff502_a(rf_filter_coeff502_a),
        .rf_filter_coeff502_b(rf_filter_coeff502_b),
        .rf_filter_coeff503_a(rf_filter_coeff503_a),
        .rf_filter_coeff503_b(rf_filter_coeff503_b),
        .rf_filter_coeff504_a(rf_filter_coeff504_a),
        .rf_filter_coeff504_b(rf_filter_coeff504_b),
        .rf_filter_coeff505_a(rf_filter_coeff505_a),
        .rf_filter_coeff505_b(rf_filter_coeff505_b),
        .rf_filter_coeff506_a(rf_filter_coeff506_a),
        .rf_filter_coeff506_b(rf_filter_coeff506_b),
        .rf_filter_coeff507_a(rf_filter_coeff507_a),
        .rf_filter_coeff507_b(rf_filter_coeff507_b),
        .rf_filter_coeff508_a(rf_filter_coeff508_a),
        .rf_filter_coeff508_b(rf_filter_coeff508_b),
        .rf_filter_coeff509_a(rf_filter_coeff509_a),
        .rf_filter_coeff509_b(rf_filter_coeff509_b),
        .rf_filter_coeff510_a(rf_filter_coeff510_a),
        .rf_filter_coeff510_b(rf_filter_coeff510_b),
        .rf_filter_coeff511_a(rf_filter_coeff511_a),
        .rf_filter_coeff511_b(rf_filter_coeff511_b)
        );
    //*****************************************************************************************


endmodule
```

filter_accumulator.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_accumulator.v
// Create Date:             10/15/2016
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module filter_accumulator(clk, rst_n, enable,load, D, Q);

    // MODULE INTERFACES
    //-------------------------------------------------------------------------
    //-------------------------------------------------------------------------
    input signed                  clk;     // input: these need descriptions
    input   signed                  rst_n;   // input:
    input   signed                  enable;  // input:
    input   signed                  load;    // input:
    input          signed  [31:0]          D;        // input:
    output  signed  [39:0]          Q;                // output:
    //-------------------------------------------------------------------------
    //-------------------------------------------------------------------------

    reg     signed  [39:0]          tmp;     // ????

    assign Q = tmp;                          //

    always @(posedge clk or negedge rst_n)
    begin
        if (!rst_n)
            tmp <= 1'b0;

        else if (enable)
        begin
            if (load)
                tmp <= D;
            else
                tmp <= tmp + D;
        end
    end
endmodule
```

filter_mux.v:

```
`timescale 1ns / 1ps
module filter_mux(clk, rden, rdptr, rddata,rf_filter_coeff0_a,
rf_filter_coeff0_b,rf_filter_coeff1_a, rf_filter_coeff1_b,rf_filter_coeff2_a,
rf_filter_coeff2_b,rf_filter_coeff3_a, rf_filter_coeff3_b,rf_filter_coeff4_a,
rf_filter_coeff4_b,rf_filter_coeff5_a, rf_filter_coeff5_b,rf_filter_coeff6_a,
rf_filter_coeff6_b,rf_filter_coeff7_a, rf_filter_coeff7_b,rf_filter_coeff8_a,
rf_filter_coeff8_b,rf_filter_coeff9_a, rf_filter_coeff9_b,rf_filter_coeff10_a,
rf_filter_coeff10_b,rf_filter_coeff11_a, rf_filter_coeff11_b,rf_filter_coeff12_a,
rf_filter_coeff12_b,rf_filter_coeff13_a, rf_filter_coeff13_b,rf_filter_coeff14_a,
rf_filter_coeff14_b,rf_filter_coeff15_a, rf_filter_coeff15_b,rf_filter_coeff16_a,
rf_filter_coeff16_b,rf_filter_coeff17_a, rf_filter_coeff17_b,rf_filter_coeff18_a,
rf_filter_coeff18_b,rf_filter_coeff19_a, rf_filter_coeff19_b,rf_filter_coeff20_a,
rf_filter_coeff20_b,rf_filter_coeff21_a, rf_filter_coeff21_b,rf_filter_coeff22_a,
rf_filter_coeff22_b,rf_filter_coeff23_a, rf_filter_coeff23_b,rf_filter_coeff24_a,
rf_filter_coeff24_b,rf_filter_coeff25_a, rf_filter_coeff25_b,rf_filter_coeff26_a,
rf_filter_coeff26_b,rf_filter_coeff27_a, rf_filter_coeff27_b,rf_filter_coeff28_a,
rf_filter_coeff28_b,rf_filter_coeff29_a, rf_filter_coeff29_b,rf_filter_coeff30_a,
rf_filter_coeff30_b,rf_filter_coeff31_a, rf_filter_coeff31_b,rf_filter_coeff32_a,
rf_filter_coeff32_b,rf_filter_coeff33_a, rf_filter_coeff33_b,rf_filter_coeff34_a,
rf_filter_coeff34_b,rf_filter_coeff35_a, rf_filter_coeff35_b,rf_filter_coeff36_a,
rf_filter_coeff36_b,rf_filter_coeff37_a, rf_filter_coeff37_b,rf_filter_coeff38_a,
rf_filter_coeff38_b,rf_filter_coeff39_a, rf_filter_coeff39_b,rf_filter_coeff40_a,
rf_filter_coeff40_b,rf_filter_coeff41_a, rf_filter_coeff41_b,rf_filter_coeff42_a,
rf_filter_coeff42_b,rf_filter_coeff43_a, rf_filter_coeff43_b,rf_filter_coeff44_a,
rf_filter_coeff44_b,rf_filter_coeff45_a, rf_filter_coeff45_b,rf_filter_coeff46_a,
rf_filter_coeff46_b,rf_filter_coeff47_a, rf_filter_coeff47_b,rf_filter_coeff48_a,
rf_filter_coeff48_b,rf_filter_coeff49_a, rf_filter_coeff49_b,rf_filter_coeff50_a,
rf_filter_coeff50_b,rf_filter_coeff51_a, rf_filter_coeff51_b,rf_filter_coeff52_a,
rf_filter_coeff52_b,rf_filter_coeff53_a, rf_filter_coeff53_b,rf_filter_coeff54_a,
rf_filter_coeff54_b,rf_filter_coeff55_a, rf_filter_coeff55_b,rf_filter_coeff56_a,
rf_filter_coeff56_b,rf_filter_coeff57_a, rf_filter_coeff57_b,rf_filter_coeff58_a,
rf_filter_coeff58_b,rf_filter_coeff59_a, rf_filter_coeff59_b,rf_filter_coeff60_a,
rf_filter_coeff60_b,rf_filter_coeff61_a, rf_filter_coeff61_b,rf_filter_coeff62_a,
rf_filter_coeff62_b,rf_filter_coeff63_a, rf_filter_coeff63_b,rf_filter_coeff64_a,
rf_filter_coeff64_b,rf_filter_coeff65_a, rf_filter_coeff65_b,rf_filter_coeff66_a,
rf_filter_coeff66_b,rf_filter_coeff67_a, rf_filter_coeff67_b,rf_filter_coeff68_a,
rf_filter_coeff68_b,rf_filter_coeff69_a, rf_filter_coeff69_b,rf_filter_coeff70_a,
rf_filter_coeff70_b,rf_filter_coeff71_a, rf_filter_coeff71_b,rf_filter_coeff72_a,
rf_filter_coeff72_b,rf_filter_coeff73_a, rf_filter_coeff73_b,rf_filter_coeff74_a,
rf_filter_coeff74_b,rf_filter_coeff75_a, rf_filter_coeff75_b,rf_filter_coeff76_a,
rf_filter_coeff76_b,rf_filter_coeff77_a, rf_filter_coeff77_b,rf_filter_coeff78_a,
rf_filter_coeff78_b,rf_filter_coeff79_a, rf_filter_coeff79_b,rf_filter_coeff80_a,
rf_filter_coeff80_b,rf_filter_coeff81_a, rf_filter_coeff81_b,rf_filter_coeff82_a,
rf_filter_coeff82_b,rf_filter_coeff83_a, rf_filter_coeff83_b,rf_filter_coeff84_a,
rf_filter_coeff84_b,rf_filter_coeff85_a, rf_filter_coeff85_b,rf_filter_coeff86_a,
rf_filter_coeff86_b,rf_filter_coeff87_a, rf_filter_coeff87_b,rf_filter_coeff88_a,
rf_filter_coeff88_b,rf_filter_coeff89_a, rf_filter_coeff89_b,rf_filter_coeff90_a,
rf_filter_coeff90_b,rf_filter_coeff91_a, rf_filter_coeff91_b,rf_filter_coeff92_a,
rf_filter_coeff92_b,rf_filter_coeff93_a, rf_filter_coeff93_b,rf_filter_coeff94_a,
rf_filter_coeff94_b,rf_filter_coeff95_a, rf_filter_coeff95_b,rf_filter_coeff96_a,
rf_filter_coeff96_b,rf_filter_coeff97_a, rf_filter_coeff97_b,rf_filter_coeff98_a,
rf_filter_coeff98_b,rf_filter_coeff99_a, rf_filter_coeff99_b,rf_filter_coeff100_a,
rf_filter_coeff100_b,rf_filter_coeff101_a, rf_filter_coeff101_b,rf_filter_coeff102_a,
rf_filter_coeff102_b,rf_filter_coeff103_a, rf_filter_coeff103_b,rf_filter_coeff104_a,
rf_filter_coeff104_b,rf_filter_coeff105_a, rf_filter_coeff105_b,rf_filter_coeff106_a,
rf_filter_coeff106_b,rf_filter_coeff107_a, rf_filter_coeff107_b,rf_filter_coeff108_a,
rf_filter_coeff108_b,rf_filter_coeff109_a, rf_filter_coeff109_b,rf_filter_coeff110_a,
rf_filter_coeff110_b,rf_filter_coeff111_a, rf_filter_coeff111_b,rf_filter_coeff112_a,
rf_filter_coeff112_b,rf_filter_coeff113_a, rf_filter_coeff113_b,rf_filter_coeff114_a,
rf_filter_coeff114_b,rf_filter_coeff115_a, rf_filter_coeff115_b,rf_filter_coeff116_a,
rf_filter_coeff116_b,rf_filter_coeff117_a, rf_filter_coeff117_b,rf_filter_coeff118_a,
rf_filter_coeff118_b,rf_filter_coeff119_a, rf_filter_coeff119_b,rf_filter_coeff120_a,
rf_filter_coeff120_b,rf_filter_coeff121_a, rf_filter_coeff121_b,rf_filter_coeff122_a,
rf_filter_coeff122_b,rf_filter_coeff123_a, rf_filter_coeff123_b,rf_filter_coeff124_a,
rf_filter_coeff124_b,rf_filter_coeff125_a, rf_filter_coeff125_b,rf_filter_coeff126_a,
rf_filter_coeff126_b,rf_filter_coeff127_a, rf_filter_coeff127_b,rf_filter_coeff128_a,
rf_filter_coeff128_b,rf_filter_coeff129_a, rf_filter_coeff129_b,rf_filter_coeff130_a,
rf_filter_coeff130_b,rf_filter_coeff131_a, rf_filter_coeff131_b,rf_filter_coeff132_a,
```

rf_filter_coeff132_b,rf_filter_coeff133_a, rf_filter_coeff133_b,rf_filter_coeff134_a,
rf_filter_coeff134_b,rf_filter_coeff135_a, rf_filter_coeff135_b,rf_filter_coeff136_a,
rf_filter_coeff136_b,rf_filter_coeff137_a, rf_filter_coeff137_b,rf_filter_coeff138_a,
rf_filter_coeff138_b,rf_filter_coeff139_a, rf_filter_coeff139_b,rf_filter_coeff140_a,
rf_filter_coeff140_b,rf_filter_coeff141_a, rf_filter_coeff141_b,rf_filter_coeff142_a,
rf_filter_coeff142_b,rf_filter_coeff143_a, rf_filter_coeff143_b,rf_filter_coeff144_a,
rf_filter_coeff144_b,rf_filter_coeff145_a, rf_filter_coeff145_b,rf_filter_coeff146_a,
rf_filter_coeff146_b,rf_filter_coeff147_a, rf_filter_coeff147_b,rf_filter_coeff148_a,
rf_filter_coeff148_b,rf_filter_coeff149_a, rf_filter_coeff149_b,rf_filter_coeff150_a,
rf_filter_coeff150_b,rf_filter_coeff151_a, rf_filter_coeff151_b,rf_filter_coeff152_a,
rf_filter_coeff152_b,rf_filter_coeff153_a, rf_filter_coeff153_b,rf_filter_coeff154_a,
rf_filter_coeff154_b,rf_filter_coeff155_a, rf_filter_coeff155_b,rf_filter_coeff156_a,
rf_filter_coeff156_b,rf_filter_coeff157_a, rf_filter_coeff157_b,rf_filter_coeff158_a,
rf_filter_coeff158_b,rf_filter_coeff159_a, rf_filter_coeff159_b,rf_filter_coeff160_a,
rf_filter_coeff160_b,rf_filter_coeff161_a, rf_filter_coeff161_b,rf_filter_coeff162_a,
rf_filter_coeff162_b,rf_filter_coeff163_a, rf_filter_coeff163_b,rf_filter_coeff164_a,
rf_filter_coeff164_b,rf_filter_coeff165_a, rf_filter_coeff165_b,rf_filter_coeff166_a,
rf_filter_coeff166_b,rf_filter_coeff167_a, rf_filter_coeff167_b,rf_filter_coeff168_a,
rf_filter_coeff168_b,rf_filter_coeff169_a, rf_filter_coeff169_b,rf_filter_coeff170_a,
rf_filter_coeff170_b,rf_filter_coeff171_a, rf_filter_coeff171_b,rf_filter_coeff172_a,
rf_filter_coeff172_b,rf_filter_coeff173_a, rf_filter_coeff173_b,rf_filter_coeff174_a,
rf_filter_coeff174_b,rf_filter_coeff175_a, rf_filter_coeff175_b,rf_filter_coeff176_a,
rf_filter_coeff176_b,rf_filter_coeff177_a, rf_filter_coeff177_b,rf_filter_coeff178_a,
rf_filter_coeff178_b,rf_filter_coeff179_a, rf_filter_coeff179_b,rf_filter_coeff180_a,
rf_filter_coeff180_b,rf_filter_coeff181_a, rf_filter_coeff181_b,rf_filter_coeff182_a,
rf_filter_coeff182_b,rf_filter_coeff183_a, rf_filter_coeff183_b,rf_filter_coeff184_a,
rf_filter_coeff184_b,rf_filter_coeff185_a, rf_filter_coeff185_b,rf_filter_coeff186_a,
rf_filter_coeff186_b,rf_filter_coeff187_a, rf_filter_coeff187_b,rf_filter_coeff188_a,
rf_filter_coeff188_b,rf_filter_coeff189_a, rf_filter_coeff189_b,rf_filter_coeff190_a,
rf_filter_coeff190_b,rf_filter_coeff191_a, rf_filter_coeff191_b,rf_filter_coeff192_a,
rf_filter_coeff192_b,rf_filter_coeff193_a, rf_filter_coeff193_b,rf_filter_coeff194_a,
rf_filter_coeff194_b,rf_filter_coeff195_a, rf_filter_coeff195_b,rf_filter_coeff196_a,
rf_filter_coeff196_b,rf_filter_coeff197_a, rf_filter_coeff197_b,rf_filter_coeff198_a,
rf_filter_coeff198_b,rf_filter_coeff199_a, rf_filter_coeff199_b,rf_filter_coeff200_a,
rf_filter_coeff200_b,rf_filter_coeff201_a, rf_filter_coeff201_b,rf_filter_coeff202_a,
rf_filter_coeff202_b,rf_filter_coeff203_a, rf_filter_coeff203_b,rf_filter_coeff204_a,
rf_filter_coeff204_b,rf_filter_coeff205_a, rf_filter_coeff205_b,rf_filter_coeff206_a,
rf_filter_coeff206_b,rf_filter_coeff207_a, rf_filter_coeff207_b,rf_filter_coeff208_a,
rf_filter_coeff208_b,rf_filter_coeff209_a, rf_filter_coeff209_b,rf_filter_coeff210_a,
rf_filter_coeff210_b,rf_filter_coeff211_a, rf_filter_coeff211_b,rf_filter_coeff212_a,
rf_filter_coeff212_b,rf_filter_coeff213_a, rf_filter_coeff213_b,rf_filter_coeff214_a,
rf_filter_coeff214_b,rf_filter_coeff215_a, rf_filter_coeff215_b,rf_filter_coeff216_a,
rf_filter_coeff216_b,rf_filter_coeff217_a, rf_filter_coeff217_b,rf_filter_coeff218_a,
rf_filter_coeff218_b,rf_filter_coeff219_a, rf_filter_coeff219_b,rf_filter_coeff220_a,
rf_filter_coeff220_b,rf_filter_coeff221_a, rf_filter_coeff221_b,rf_filter_coeff222_a,
rf_filter_coeff222_b,rf_filter_coeff223_a, rf_filter_coeff223_b,rf_filter_coeff224_a,
rf_filter_coeff224_b,rf_filter_coeff225_a, rf_filter_coeff225_b,rf_filter_coeff226_a,
rf_filter_coeff226_b,rf_filter_coeff227_a, rf_filter_coeff227_b,rf_filter_coeff228_a,
rf_filter_coeff228_b,rf_filter_coeff229_a, rf_filter_coeff229_b,rf_filter_coeff230_a,
rf_filter_coeff230_b,rf_filter_coeff231_a, rf_filter_coeff231_b,rf_filter_coeff232_a,
rf_filter_coeff232_b,rf_filter_coeff233_a, rf_filter_coeff233_b,rf_filter_coeff234_a,
rf_filter_coeff234_b,rf_filter_coeff235_a, rf_filter_coeff235_b,rf_filter_coeff236_a,
rf_filter_coeff236_b,rf_filter_coeff237_a, rf_filter_coeff237_b,rf_filter_coeff238_a,
rf_filter_coeff238_b,rf_filter_coeff239_a, rf_filter_coeff239_b,rf_filter_coeff240_a,
rf_filter_coeff240_b,rf_filter_coeff241_a, rf_filter_coeff241_b,rf_filter_coeff242_a,
rf_filter_coeff242_b,rf_filter_coeff243_a, rf_filter_coeff243_b,rf_filter_coeff244_a,
rf_filter_coeff244_b,rf_filter_coeff245_a, rf_filter_coeff245_b,rf_filter_coeff246_a,
rf_filter_coeff246_b,rf_filter_coeff247_a, rf_filter_coeff247_b,rf_filter_coeff248_a,
rf_filter_coeff248_b,rf_filter_coeff249_a, rf_filter_coeff249_b,rf_filter_coeff250_a,
rf_filter_coeff250_b,rf_filter_coeff251_a, rf_filter_coeff251_b,rf_filter_coeff252_a,
rf_filter_coeff252_b,rf_filter_coeff253_a, rf_filter_coeff253_b,rf_filter_coeff254_a,
rf_filter_coeff254_b,rf_filter_coeff255_a, rf_filter_coeff255_b,rf_filter_coeff256_a,
rf_filter_coeff256_b,rf_filter_coeff257_a, rf_filter_coeff257_b,rf_filter_coeff258_a,
rf_filter_coeff258_b,rf_filter_coeff259_a, rf_filter_coeff259_b,rf_filter_coeff260_a,
rf_filter_coeff260_b,rf_filter_coeff261_a, rf_filter_coeff261_b,rf_filter_coeff262_a,
rf_filter_coeff262_b,rf_filter_coeff263_a, rf_filter_coeff263_b,rf_filter_coeff264_a,
rf_filter_coeff264_b,rf_filter_coeff265_a, rf_filter_coeff265_b,rf_filter_coeff266_a,
rf_filter_coeff266_b,rf_filter_coeff267_a, rf_filter_coeff267_b,rf_filter_coeff268_a,
rf_filter_coeff268_b,rf_filter_coeff269_a, rf_filter_coeff269_b,rf_filter_coeff270_a,
rf_filter_coeff270_b,rf_filter_coeff271_a, rf_filter_coeff271_b,rf_filter_coeff272_a,
rf_filter_coeff272_b,rf_filter_coeff273_a, rf_filter_coeff273_b,rf_filter_coeff274_a,

rf_filter_coeff274_b,rf_filter_coeff275_a, rf_filter_coeff275_b,rf_filter_coeff276_a,
rf_filter_coeff276_b,rf_filter_coeff277_a, rf_filter_coeff277_b,rf_filter_coeff278_a,
rf_filter_coeff278_b,rf_filter_coeff279_a, rf_filter_coeff279_b,rf_filter_coeff280_a,
rf_filter_coeff280_b,rf_filter_coeff281_a, rf_filter_coeff281_b,rf_filter_coeff282_a,
rf_filter_coeff282_b,rf_filter_coeff283_a, rf_filter_coeff283_b,rf_filter_coeff284_a,
rf_filter_coeff284_b,rf_filter_coeff285_a, rf_filter_coeff285_b,rf_filter_coeff286_a,
rf_filter_coeff286_b,rf_filter_coeff287_a, rf_filter_coeff287_b,rf_filter_coeff288_a,
rf_filter_coeff288_b,rf_filter_coeff289_a, rf_filter_coeff289_b,rf_filter_coeff290_a,
rf_filter_coeff290_b,rf_filter_coeff291_a, rf_filter_coeff291_b,rf_filter_coeff292_a,
rf_filter_coeff292_b,rf_filter_coeff293_a, rf_filter_coeff293_b,rf_filter_coeff294_a,
rf_filter_coeff294_b,rf_filter_coeff295_a, rf_filter_coeff295_b,rf_filter_coeff296_a,
rf_filter_coeff296_b,rf_filter_coeff297_a, rf_filter_coeff297_b,rf_filter_coeff298_a,
rf_filter_coeff298_b,rf_filter_coeff299_a, rf_filter_coeff299_b,rf_filter_coeff300_a,
rf_filter_coeff300_b,rf_filter_coeff301_a, rf_filter_coeff301_b,rf_filter_coeff302_a,
rf_filter_coeff302_b,rf_filter_coeff303_a, rf_filter_coeff303_b,rf_filter_coeff304_a,
rf_filter_coeff304_b,rf_filter_coeff305_a, rf_filter_coeff305_b,rf_filter_coeff306_a,
rf_filter_coeff306_b,rf_filter_coeff307_a, rf_filter_coeff307_b,rf_filter_coeff308_a,
rf_filter_coeff308_b,rf_filter_coeff309_a, rf_filter_coeff309_b,rf_filter_coeff310_a,
rf_filter_coeff310_b,rf_filter_coeff311_a, rf_filter_coeff311_b,rf_filter_coeff312_a,
rf_filter_coeff312_b,rf_filter_coeff313_a, rf_filter_coeff313_b,rf_filter_coeff314_a,
rf_filter_coeff314_b,rf_filter_coeff315_a, rf_filter_coeff315_b,rf_filter_coeff316_a,
rf_filter_coeff316_b,rf_filter_coeff317_a, rf_filter_coeff317_b,rf_filter_coeff318_a,
rf_filter_coeff318_b,rf_filter_coeff319_a, rf_filter_coeff319_b,rf_filter_coeff320_a,
rf_filter_coeff320_b,rf_filter_coeff321_a, rf_filter_coeff321_b,rf_filter_coeff322_a,
rf_filter_coeff322_b,rf_filter_coeff323_a, rf_filter_coeff323_b,rf_filter_coeff324_a,
rf_filter_coeff324_b,rf_filter_coeff325_a, rf_filter_coeff325_b,rf_filter_coeff326_a,
rf_filter_coeff326_b,rf_filter_coeff327_a, rf_filter_coeff327_b,rf_filter_coeff328_a,
rf_filter_coeff328_b,rf_filter_coeff329_a, rf_filter_coeff329_b,rf_filter_coeff330_a,
rf_filter_coeff330_b,rf_filter_coeff331_a, rf_filter_coeff331_b,rf_filter_coeff332_a,
rf_filter_coeff332_b,rf_filter_coeff333_a, rf_filter_coeff333_b,rf_filter_coeff334_a,
rf_filter_coeff334_b,rf_filter_coeff335_a, rf_filter_coeff335_b,rf_filter_coeff336_a,
rf_filter_coeff336_b,rf_filter_coeff337_a, rf_filter_coeff337_b,rf_filter_coeff338_a,
rf_filter_coeff338_b,rf_filter_coeff339_a, rf_filter_coeff339_b,rf_filter_coeff340_a,
rf_filter_coeff340_b,rf_filter_coeff341_a, rf_filter_coeff341_b,rf_filter_coeff342_a,
rf_filter_coeff342_b,rf_filter_coeff343_a, rf_filter_coeff343_b,rf_filter_coeff344_a,
rf_filter_coeff344_b,rf_filter_coeff345_a, rf_filter_coeff345_b,rf_filter_coeff346_a,
rf_filter_coeff346_b,rf_filter_coeff347_a, rf_filter_coeff347_b,rf_filter_coeff348_a,
rf_filter_coeff348_b,rf_filter_coeff349_a, rf_filter_coeff349_b,rf_filter_coeff350_a,
rf_filter_coeff350_b,rf_filter_coeff351_a, rf_filter_coeff351_b,rf_filter_coeff352_a,
rf_filter_coeff352_b,rf_filter_coeff353_a, rf_filter_coeff353_b,rf_filter_coeff354_a,
rf_filter_coeff354_b,rf_filter_coeff355_a, rf_filter_coeff355_b,rf_filter_coeff356_a,
rf_filter_coeff356_b,rf_filter_coeff357_a, rf_filter_coeff357_b,rf_filter_coeff358_a,
rf_filter_coeff358_b,rf_filter_coeff359_a, rf_filter_coeff359_b,rf_filter_coeff360_a,
rf_filter_coeff360_b,rf_filter_coeff361_a, rf_filter_coeff361_b,rf_filter_coeff362_a,
rf_filter_coeff362_b,rf_filter_coeff363_a, rf_filter_coeff363_b,rf_filter_coeff364_a,
rf_filter_coeff364_b,rf_filter_coeff365_a, rf_filter_coeff365_b,rf_filter_coeff366_a,
rf_filter_coeff366_b,rf_filter_coeff367_a, rf_filter_coeff367_b,rf_filter_coeff368_a,
rf_filter_coeff368_b,rf_filter_coeff369_a, rf_filter_coeff369_b,rf_filter_coeff370_a,
rf_filter_coeff370_b,rf_filter_coeff371_a, rf_filter_coeff371_b,rf_filter_coeff372_a,
rf_filter_coeff372_b,rf_filter_coeff373_a, rf_filter_coeff373_b,rf_filter_coeff374_a,
rf_filter_coeff374_b,rf_filter_coeff375_a, rf_filter_coeff375_b,rf_filter_coeff376_a,
rf_filter_coeff376_b,rf_filter_coeff377_a, rf_filter_coeff377_b,rf_filter_coeff378_a,
rf_filter_coeff378_b,rf_filter_coeff379_a, rf_filter_coeff379_b,rf_filter_coeff380_a,
rf_filter_coeff380_b,rf_filter_coeff381_a, rf_filter_coeff381_b,rf_filter_coeff382_a,
rf_filter_coeff382_b,rf_filter_coeff383_a, rf_filter_coeff383_b,rf_filter_coeff384_a,
rf_filter_coeff384_b,rf_filter_coeff385_a, rf_filter_coeff385_b,rf_filter_coeff386_a,
rf_filter_coeff386_b,rf_filter_coeff387_a, rf_filter_coeff387_b,rf_filter_coeff388_a,
rf_filter_coeff388_b,rf_filter_coeff389_a, rf_filter_coeff389_b,rf_filter_coeff390_a,
rf_filter_coeff390_b,rf_filter_coeff391_a, rf_filter_coeff391_b,rf_filter_coeff392_a,
rf_filter_coeff392_b,rf_filter_coeff393_a, rf_filter_coeff393_b,rf_filter_coeff394_a,
rf_filter_coeff394_b,rf_filter_coeff395_a, rf_filter_coeff395_b,rf_filter_coeff396_a,
rf_filter_coeff396_b,rf_filter_coeff397_a, rf_filter_coeff397_b,rf_filter_coeff398_a,
rf_filter_coeff398_b,rf_filter_coeff399_a, rf_filter_coeff399_b,rf_filter_coeff400_a,
rf_filter_coeff400_b,rf_filter_coeff401_a, rf_filter_coeff401_b,rf_filter_coeff402_a,
rf_filter_coeff402_b,rf_filter_coeff403_a, rf_filter_coeff403_b,rf_filter_coeff404_a,
rf_filter_coeff404_b,rf_filter_coeff405_a, rf_filter_coeff405_b,rf_filter_coeff406_a,
rf_filter_coeff406_b,rf_filter_coeff407_a, rf_filter_coeff407_b,rf_filter_coeff408_a,
rf_filter_coeff408_b,rf_filter_coeff409_a, rf_filter_coeff409_b,rf_filter_coeff410_a,
rf_filter_coeff410_b,rf_filter_coeff411_a, rf_filter_coeff411_b,rf_filter_coeff412_a,
rf_filter_coeff412_b,rf_filter_coeff413_a, rf_filter_coeff413_b,rf_filter_coeff414_a,
rf_filter_coeff414_b,rf_filter_coeff415_a, rf_filter_coeff415_b,rf_filter_coeff416_a,

```
rf_filter_coeff416_b,rf_filter_coeff417_a, rf_filter_coeff417_b,rf_filter_coeff418_a,
rf_filter_coeff418_b,rf_filter_coeff419_a, rf_filter_coeff419_b,rf_filter_coeff420_a,
rf_filter_coeff420_b,rf_filter_coeff421_a, rf_filter_coeff421_b,rf_filter_coeff422_a,
rf_filter_coeff422_b,rf_filter_coeff423_a, rf_filter_coeff423_b,rf_filter_coeff424_a,
rf_filter_coeff424_b,rf_filter_coeff425_a, rf_filter_coeff425_b,rf_filter_coeff426_a,
rf_filter_coeff426_b,rf_filter_coeff427_a, rf_filter_coeff427_b,rf_filter_coeff428_a,
rf_filter_coeff428_b,rf_filter_coeff429_a, rf_filter_coeff429_b,rf_filter_coeff430_a,
rf_filter_coeff430_b,rf_filter_coeff431_a, rf_filter_coeff431_b,rf_filter_coeff432_a,
rf_filter_coeff432_b,rf_filter_coeff433_a, rf_filter_coeff433_b,rf_filter_coeff434_a,
rf_filter_coeff434_b,rf_filter_coeff435_a, rf_filter_coeff435_b,rf_filter_coeff436_a,
rf_filter_coeff436_b,rf_filter_coeff437_a, rf_filter_coeff437_b,rf_filter_coeff438_a,
rf_filter_coeff438_b,rf_filter_coeff439_a, rf_filter_coeff439_b,rf_filter_coeff440_a,
rf_filter_coeff440_b,rf_filter_coeff441_a, rf_filter_coeff441_b,rf_filter_coeff442_a,
rf_filter_coeff442_b,rf_filter_coeff443_a, rf_filter_coeff443_b,rf_filter_coeff444_a,
rf_filter_coeff444_b,rf_filter_coeff445_a, rf_filter_coeff445_b,rf_filter_coeff446_a,
rf_filter_coeff446_b,rf_filter_coeff447_a, rf_filter_coeff447_b,rf_filter_coeff448_a,
rf_filter_coeff448_b,rf_filter_coeff449_a, rf_filter_coeff449_b,rf_filter_coeff450_a,
rf_filter_coeff450_b,rf_filter_coeff451_a, rf_filter_coeff451_b,rf_filter_coeff452_a,
rf_filter_coeff452_b,rf_filter_coeff453_a, rf_filter_coeff453_b,rf_filter_coeff454_a,
rf_filter_coeff454_b,rf_filter_coeff455_a, rf_filter_coeff455_b,rf_filter_coeff456_a,
rf_filter_coeff456_b,rf_filter_coeff457_a, rf_filter_coeff457_b,rf_filter_coeff458_a,
rf_filter_coeff458_b,rf_filter_coeff459_a, rf_filter_coeff459_b,rf_filter_coeff460_a,
rf_filter_coeff460_b,rf_filter_coeff461_a, rf_filter_coeff461_b,rf_filter_coeff462_a,
rf_filter_coeff462_b,rf_filter_coeff463_a, rf_filter_coeff463_b,rf_filter_coeff464_a,
rf_filter_coeff464_b,rf_filter_coeff465_a, rf_filter_coeff465_b,rf_filter_coeff466_a,
rf_filter_coeff466_b,rf_filter_coeff467_a, rf_filter_coeff467_b,rf_filter_coeff468_a,
rf_filter_coeff468_b,rf_filter_coeff469_a, rf_filter_coeff469_b,rf_filter_coeff470_a,
rf_filter_coeff470_b,rf_filter_coeff471_a, rf_filter_coeff471_b,rf_filter_coeff472_a,
rf_filter_coeff472_b,rf_filter_coeff473_a, rf_filter_coeff473_b,rf_filter_coeff474_a,
rf_filter_coeff474_b,rf_filter_coeff475_a, rf_filter_coeff475_b,rf_filter_coeff476_a,
rf_filter_coeff476_b,rf_filter_coeff477_a, rf_filter_coeff477_b,rf_filter_coeff478_a,
rf_filter_coeff478_b,rf_filter_coeff479_a, rf_filter_coeff479_b,rf_filter_coeff480_a,
rf_filter_coeff480_b,rf_filter_coeff481_a, rf_filter_coeff481_b,rf_filter_coeff482_a,
rf_filter_coeff482_b,rf_filter_coeff483_a, rf_filter_coeff483_b,rf_filter_coeff484_a,
rf_filter_coeff484_b,rf_filter_coeff485_a, rf_filter_coeff485_b,rf_filter_coeff486_a,
rf_filter_coeff486_b,rf_filter_coeff487_a, rf_filter_coeff487_b,rf_filter_coeff488_a,
rf_filter_coeff488_b,rf_filter_coeff489_a, rf_filter_coeff489_b,rf_filter_coeff490_a,
rf_filter_coeff490_b,rf_filter_coeff491_a, rf_filter_coeff491_b,rf_filter_coeff492_a,
rf_filter_coeff492_b,rf_filter_coeff493_a, rf_filter_coeff493_b,rf_filter_coeff494_a,
rf_filter_coeff494_b,rf_filter_coeff495_a, rf_filter_coeff495_b,rf_filter_coeff496_a,
rf_filter_coeff496_b,rf_filter_coeff497_a, rf_filter_coeff497_b,rf_filter_coeff498_a,
rf_filter_coeff498_b,rf_filter_coeff499_a, rf_filter_coeff499_b,rf_filter_coeff500_a,
rf_filter_coeff500_b,rf_filter_coeff501_a, rf_filter_coeff501_b,rf_filter_coeff502_a,
rf_filter_coeff502_b,rf_filter_coeff503_a, rf_filter_coeff503_b,rf_filter_coeff504_a,
rf_filter_coeff504_b,rf_filter_coeff505_a, rf_filter_coeff505_b,rf_filter_coeff506_a,
rf_filter_coeff506_b,rf_filter_coeff507_a, rf_filter_coeff507_b,rf_filter_coeff508_a,
rf_filter_coeff508_b,rf_filter_coeff509_a, rf_filter_coeff509_b,rf_filter_coeff510_a,
rf_filter_coeff510_b,rf_filter_coeff511_a, rf_filter_coeff511_b
   );

    // MODULE INTERFACE
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------
input [7:0] rf_filter_coeff0_a, rf_filter_coeff0_b,
rf_filter_coeff1_a, rf_filter_coeff1_b,rf_filter_coeff2_a, rf_filter_coeff2_b,rf_filter_coeff3_a,
rf_filter_coeff3_b,rf_filter_coeff4_a, rf_filter_coeff4_b,
rf_filter_coeff5_a, rf_filter_coeff5_b,rf_filter_coeff6_a, rf_filter_coeff6_b,rf_filter_coeff7_a,
rf_filter_coeff7_b,rf_filter_coeff8_a, rf_filter_coeff8_b,
rf_filter_coeff9_a, rf_filter_coeff9_b,rf_filter_coeff10_a,
rf_filter_coeff10_b,rf_filter_coeff11_a, rf_filter_coeff11_b,rf_filter_coeff12_a,
rf_filter_coeff12_b,
rf_filter_coeff13_a, rf_filter_coeff13_b,rf_filter_coeff14_a,
rf_filter_coeff14_b,rf_filter_coeff15_a, rf_filter_coeff15_b,rf_filter_coeff16_a,
rf_filter_coeff16_b,
rf_filter_coeff17_a, rf_filter_coeff17_b,rf_filter_coeff18_a,
rf_filter_coeff18_b,rf_filter_coeff19_a, rf_filter_coeff19_b,rf_filter_coeff20_a,
rf_filter_coeff20_b,
rf_filter_coeff21_a, rf_filter_coeff21_b,rf_filter_coeff22_a,
rf_filter_coeff22_b,rf_filter_coeff23_a, rf_filter_coeff23_b,rf_filter_coeff24_a,
rf_filter_coeff24_b,
```

```
rf_filter_coeff25_a, rf_filter_coeff25_b,rf_filter_coeff26_a,
rf_filter_coeff26_b,rf_filter_coeff27_a, rf_filter_coeff27_b,rf_filter_coeff28_a,
rf_filter_coeff28_b,
rf_filter_coeff29_a, rf_filter_coeff29_b,rf_filter_coeff30_a,
rf_filter_coeff30_b,rf_filter_coeff31_a, rf_filter_coeff31_b,rf_filter_coeff32_a,
rf_filter_coeff32_b,
rf_filter_coeff33_a, rf_filter_coeff33_b,rf_filter_coeff34_a,
rf_filter_coeff34_b,rf_filter_coeff35_a, rf_filter_coeff35_b,rf_filter_coeff36_a,
rf_filter_coeff36_b,
rf_filter_coeff37_a, rf_filter_coeff37_b,rf_filter_coeff38_a,
rf_filter_coeff38_b,rf_filter_coeff39_a, rf_filter_coeff39_b,rf_filter_coeff40_a,
rf_filter_coeff40_b,
rf_filter_coeff41_a, rf_filter_coeff41_b,rf_filter_coeff42_a,
rf_filter_coeff42_b,rf_filter_coeff43_a, rf_filter_coeff43_b,rf_filter_coeff44_a,
rf_filter_coeff44_b,
rf_filter_coeff45_a, rf_filter_coeff45_b,rf_filter_coeff46_a,
rf_filter_coeff46_b,rf_filter_coeff47_a, rf_filter_coeff47_b,rf_filter_coeff48_a,
rf_filter_coeff48_b,
rf_filter_coeff49_a, rf_filter_coeff49_b,rf_filter_coeff50_a,
rf_filter_coeff50_b,rf_filter_coeff51_a, rf_filter_coeff51_b,rf_filter_coeff52_a,
rf_filter_coeff52_b,
rf_filter_coeff53_a, rf_filter_coeff53_b,rf_filter_coeff54_a,
rf_filter_coeff54_b,rf_filter_coeff55_a, rf_filter_coeff55_b,rf_filter_coeff56_a,
rf_filter_coeff56_b,
rf_filter_coeff57_a, rf_filter_coeff57_b,rf_filter_coeff58_a,
rf_filter_coeff58_b,rf_filter_coeff59_a, rf_filter_coeff59_b,rf_filter_coeff60_a,
rf_filter_coeff60_b,
rf_filter_coeff61_a, rf_filter_coeff61_b,rf_filter_coeff62_a,
rf_filter_coeff62_b,rf_filter_coeff63_a, rf_filter_coeff63_b,rf_filter_coeff64_a,
rf_filter_coeff64_b,
rf_filter_coeff65_a, rf_filter_coeff65_b,rf_filter_coeff66_a,
rf_filter_coeff66_b,rf_filter_coeff67_a, rf_filter_coeff67_b,rf_filter_coeff68_a,
rf_filter_coeff68_b,
rf_filter_coeff69_a, rf_filter_coeff69_b,rf_filter_coeff70_a,
rf_filter_coeff70_b,rf_filter_coeff71_a, rf_filter_coeff71_b,rf_filter_coeff72_a,
rf_filter_coeff72_b,
rf_filter_coeff73_a, rf_filter_coeff73_b,rf_filter_coeff74_a,
rf_filter_coeff74_b,rf_filter_coeff75_a, rf_filter_coeff75_b,rf_filter_coeff76_a,
rf_filter_coeff76_b,
rf_filter_coeff77_a, rf_filter_coeff77_b,rf_filter_coeff78_a,
rf_filter_coeff78_b,rf_filter_coeff79_a, rf_filter_coeff79_b,rf_filter_coeff80_a,
rf_filter_coeff80_b,
rf_filter_coeff81_a, rf_filter_coeff81_b,rf_filter_coeff82_a,
rf_filter_coeff82_b,rf_filter_coeff83_a, rf_filter_coeff83_b,rf_filter_coeff84_a,
rf_filter_coeff84_b,
rf_filter_coeff85_a, rf_filter_coeff85_b,rf_filter_coeff86_a,
rf_filter_coeff86_b,rf_filter_coeff87_a, rf_filter_coeff87_b,rf_filter_coeff88_a,
rf_filter_coeff88_b,
rf_filter_coeff89_a, rf_filter_coeff89_b,rf_filter_coeff90_a,
rf_filter_coeff90_b,rf_filter_coeff91_a, rf_filter_coeff91_b,rf_filter_coeff92_a,
rf_filter_coeff92_b,
rf_filter_coeff93_a, rf_filter_coeff93_b,rf_filter_coeff94_a,
rf_filter_coeff94_b,rf_filter_coeff95_a, rf_filter_coeff95_b,rf_filter_coeff96_a,
rf_filter_coeff96_b,
rf_filter_coeff97_a, rf_filter_coeff97_b,rf_filter_coeff98_a,
rf_filter_coeff98_b,rf_filter_coeff99_a, rf_filter_coeff99_b,rf_filter_coeff100_a,
rf_filter_coeff100_b,
rf_filter_coeff101_a, rf_filter_coeff101_b,rf_filter_coeff102_a,
rf_filter_coeff102_b,rf_filter_coeff103_a, rf_filter_coeff103_b,rf_filter_coeff104_a,
rf_filter_coeff104_b,
rf_filter_coeff105_a, rf_filter_coeff105_b,rf_filter_coeff106_a,
rf_filter_coeff106_b,rf_filter_coeff107_a, rf_filter_coeff107_b,rf_filter_coeff108_a,
rf_filter_coeff108_b,
rf_filter_coeff109_a, rf_filter_coeff109_b,rf_filter_coeff110_a,
rf_filter_coeff110_b,rf_filter_coeff111_a, rf_filter_coeff111_b,rf_filter_coeff112_a,
rf_filter_coeff112_b,
rf_filter_coeff113_a, rf_filter_coeff113_b,rf_filter_coeff114_a,
rf_filter_coeff114_b,rf_filter_coeff115_a, rf_filter_coeff115_b,rf_filter_coeff116_a,
rf_filter_coeff116_b,
```

```
rf_filter_coeff117_a, rf_filter_coeff117_b,rf_filter_coeff118_a,
rf_filter_coeff118_b,rf_filter_coeff119_a, rf_filter_coeff119_b,rf_filter_coeff120_a,
rf_filter_coeff120_b,
rf_filter_coeff121_a, rf_filter_coeff121_b,rf_filter_coeff122_a,
rf_filter_coeff122_b,rf_filter_coeff123_a, rf_filter_coeff123_b,rf_filter_coeff124_a,
rf_filter_coeff124_b,
rf_filter_coeff125_a, rf_filter_coeff125_b,rf_filter_coeff126_a,
rf_filter_coeff126_b,rf_filter_coeff127_a, rf_filter_coeff127_b,rf_filter_coeff128_a,
rf_filter_coeff128_b,
rf_filter_coeff129_a, rf_filter_coeff129_b,rf_filter_coeff130_a,
rf_filter_coeff130_b,rf_filter_coeff131_a, rf_filter_coeff131_b,rf_filter_coeff132_a,
rf_filter_coeff132_b,
rf_filter_coeff133_a, rf_filter_coeff133_b,rf_filter_coeff134_a,
rf_filter_coeff134_b,rf_filter_coeff135_a, rf_filter_coeff135_b,rf_filter_coeff136_a,
rf_filter_coeff136_b,
rf_filter_coeff137_a, rf_filter_coeff137_b,rf_filter_coeff138_a,
rf_filter_coeff138_b,rf_filter_coeff139_a, rf_filter_coeff139_b,rf_filter_coeff140_a,
rf_filter_coeff140_b,
rf_filter_coeff141_a, rf_filter_coeff141_b,rf_filter_coeff142_a,
rf_filter_coeff142_b,rf_filter_coeff143_a, rf_filter_coeff143_b,rf_filter_coeff144_a,
rf_filter_coeff144_b,
rf_filter_coeff145_a, rf_filter_coeff145_b,rf_filter_coeff146_a,
rf_filter_coeff146_b,rf_filter_coeff147_a, rf_filter_coeff147_b,rf_filter_coeff148_a,
rf_filter_coeff148_b,
rf_filter_coeff149_a, rf_filter_coeff149_b,rf_filter_coeff150_a,
rf_filter_coeff150_b,rf_filter_coeff151_a, rf_filter_coeff151_b,rf_filter_coeff152_a,
rf_filter_coeff152_b,
rf_filter_coeff153_a, rf_filter_coeff153_b,rf_filter_coeff154_a,
rf_filter_coeff154_b,rf_filter_coeff155_a, rf_filter_coeff155_b,rf_filter_coeff156_a,
rf_filter_coeff156_b,
rf_filter_coeff157_a, rf_filter_coeff157_b,rf_filter_coeff158_a,
rf_filter_coeff158_b,rf_filter_coeff159_a, rf_filter_coeff159_b,rf_filter_coeff160_a,
rf_filter_coeff160_b,
rf_filter_coeff161_a, rf_filter_coeff161_b,rf_filter_coeff162_a,
rf_filter_coeff162_b,rf_filter_coeff163_a, rf_filter_coeff163_b,rf_filter_coeff164_a,
rf_filter_coeff164_b,
rf_filter_coeff165_a, rf_filter_coeff165_b,rf_filter_coeff166_a,
rf_filter_coeff166_b,rf_filter_coeff167_a, rf_filter_coeff167_b,rf_filter_coeff168_a,
rf_filter_coeff168_b,
rf_filter_coeff169_a, rf_filter_coeff169_b,rf_filter_coeff170_a,
rf_filter_coeff170_b,rf_filter_coeff171_a, rf_filter_coeff171_b,rf_filter_coeff172_a,
rf_filter_coeff172_b,
rf_filter_coeff173_a, rf_filter_coeff173_b,rf_filter_coeff174_a,
rf_filter_coeff174_b,rf_filter_coeff175_a, rf_filter_coeff175_b,rf_filter_coeff176_a,
rf_filter_coeff176_b,
rf_filter_coeff177_a, rf_filter_coeff177_b,rf_filter_coeff178_a,
rf_filter_coeff178_b,rf_filter_coeff179_a, rf_filter_coeff179_b,rf_filter_coeff180_a,
rf_filter_coeff180_b,
rf_filter_coeff181_a, rf_filter_coeff181_b,rf_filter_coeff182_a,
rf_filter_coeff182_b,rf_filter_coeff183_a, rf_filter_coeff183_b,rf_filter_coeff184_a,
rf_filter_coeff184_b,
rf_filter_coeff185_a, rf_filter_coeff185_b,rf_filter_coeff186_a,
rf_filter_coeff186_b,rf_filter_coeff187_a, rf_filter_coeff187_b,rf_filter_coeff188_a,
rf_filter_coeff188_b,
rf_filter_coeff189_a, rf_filter_coeff189_b,rf_filter_coeff190_a,
rf_filter_coeff190_b,rf_filter_coeff191_a, rf_filter_coeff191_b,rf_filter_coeff192_a,
rf_filter_coeff192_b,
rf_filter_coeff193_a, rf_filter_coeff193_b,rf_filter_coeff194_a,
rf_filter_coeff194_b,rf_filter_coeff195_a, rf_filter_coeff195_b,rf_filter_coeff196_a,
rf_filter_coeff196_b,
rf_filter_coeff197_a, rf_filter_coeff197_b,rf_filter_coeff198_a,
rf_filter_coeff198_b,rf_filter_coeff199_a, rf_filter_coeff199_b,rf_filter_coeff200_a,
rf_filter_coeff200_b,
rf_filter_coeff201_a, rf_filter_coeff201_b,rf_filter_coeff202_a,
rf_filter_coeff202_b,rf_filter_coeff203_a, rf_filter_coeff203_b,rf_filter_coeff204_a,
rf_filter_coeff204_b,
rf_filter_coeff205_a, rf_filter_coeff205_b,rf_filter_coeff206_a,
rf_filter_coeff206_b,rf_filter_coeff207_a, rf_filter_coeff207_b,rf_filter_coeff208_a,
rf_filter_coeff208_b,
```

```
rf_filter_coeff209_a, rf_filter_coeff209_b,rf_filter_coeff210_a,
rf_filter_coeff210_b,rf_filter_coeff211_a, rf_filter_coeff211_b,rf_filter_coeff212_a,
rf_filter_coeff212_b,
rf_filter_coeff213_a, rf_filter_coeff213_b,rf_filter_coeff214_a,
rf_filter_coeff214_b,rf_filter_coeff215_a, rf_filter_coeff215_b,rf_filter_coeff216_a,
rf_filter_coeff216_b,
rf_filter_coeff217_a, rf_filter_coeff217_b,rf_filter_coeff218_a,
rf_filter_coeff218_b,rf_filter_coeff219_a, rf_filter_coeff219_b,rf_filter_coeff220_a,
rf_filter_coeff220_b,
rf_filter_coeff221_a, rf_filter_coeff221_b,rf_filter_coeff222_a,
rf_filter_coeff222_b,rf_filter_coeff223_a, rf_filter_coeff223_b,rf_filter_coeff224_a,
rf_filter_coeff224_b,
rf_filter_coeff225_a, rf_filter_coeff225_b,rf_filter_coeff226_a,
rf_filter_coeff226_b,rf_filter_coeff227_a, rf_filter_coeff227_b,rf_filter_coeff228_a,
rf_filter_coeff228_b,
rf_filter_coeff229_a, rf_filter_coeff229_b,rf_filter_coeff230_a,
rf_filter_coeff230_b,rf_filter_coeff231_a, rf_filter_coeff231_b,rf_filter_coeff232_a,
rf_filter_coeff232_b,
rf_filter_coeff233_a, rf_filter_coeff233_b,rf_filter_coeff234_a,
rf_filter_coeff234_b,rf_filter_coeff235_a, rf_filter_coeff235_b,rf_filter_coeff236_a,
rf_filter_coeff236_b,
rf_filter_coeff237_a, rf_filter_coeff237_b,rf_filter_coeff238_a,
rf_filter_coeff238_b,rf_filter_coeff239_a, rf_filter_coeff239_b,rf_filter_coeff240_a,
rf_filter_coeff240_b,
rf_filter_coeff241_a, rf_filter_coeff241_b,rf_filter_coeff242_a,
rf_filter_coeff242_b,rf_filter_coeff243_a, rf_filter_coeff243_b,rf_filter_coeff244_a,
rf_filter_coeff244_b,
rf_filter_coeff245_a, rf_filter_coeff245_b,rf_filter_coeff246_a,
rf_filter_coeff246_b,rf_filter_coeff247_a, rf_filter_coeff247_b,rf_filter_coeff248_a,
rf_filter_coeff248_b,
rf_filter_coeff249_a, rf_filter_coeff249_b,rf_filter_coeff250_a,
rf_filter_coeff250_b,rf_filter_coeff251_a, rf_filter_coeff251_b,rf_filter_coeff252_a,
rf_filter_coeff252_b,
rf_filter_coeff253_a, rf_filter_coeff253_b,rf_filter_coeff254_a,
rf_filter_coeff254_b,rf_filter_coeff255_a, rf_filter_coeff255_b,rf_filter_coeff256_a,
rf_filter_coeff256_b,
rf_filter_coeff257_a, rf_filter_coeff257_b,rf_filter_coeff258_a,
rf_filter_coeff258_b,rf_filter_coeff259_a, rf_filter_coeff259_b,rf_filter_coeff260_a,
rf_filter_coeff260_b,
rf_filter_coeff261_a, rf_filter_coeff261_b,rf_filter_coeff262_a,
rf_filter_coeff262_b,rf_filter_coeff263_a, rf_filter_coeff263_b,rf_filter_coeff264_a,
rf_filter_coeff264_b,
rf_filter_coeff265_a, rf_filter_coeff265_b,rf_filter_coeff266_a,
rf_filter_coeff266_b,rf_filter_coeff267_a, rf_filter_coeff267_b,rf_filter_coeff268_a,
rf_filter_coeff268_b,
rf_filter_coeff269_a, rf_filter_coeff269_b,rf_filter_coeff270_a,
rf_filter_coeff270_b,rf_filter_coeff271_a, rf_filter_coeff271_b,rf_filter_coeff272_a,
rf_filter_coeff272_b,
rf_filter_coeff273_a, rf_filter_coeff273_b,rf_filter_coeff274_a,
rf_filter_coeff274_b,rf_filter_coeff275_a, rf_filter_coeff275_b,rf_filter_coeff276_a,
rf_filter_coeff276_b,
rf_filter_coeff277_a, rf_filter_coeff277_b,rf_filter_coeff278_a,
rf_filter_coeff278_b,rf_filter_coeff279_a, rf_filter_coeff279_b,rf_filter_coeff280_a,
rf_filter_coeff280_b,
rf_filter_coeff281_a, rf_filter_coeff281_b,rf_filter_coeff282_a,
rf_filter_coeff282_b,rf_filter_coeff283_a, rf_filter_coeff283_b,rf_filter_coeff284_a,
rf_filter_coeff284_b,
rf_filter_coeff285_a, rf_filter_coeff285_b,rf_filter_coeff286_a,
rf_filter_coeff286_b,rf_filter_coeff287_a, rf_filter_coeff287_b,rf_filter_coeff288_a,
rf_filter_coeff288_b,
rf_filter_coeff289_a, rf_filter_coeff289_b,rf_filter_coeff290_a,
rf_filter_coeff290_b,rf_filter_coeff291_a, rf_filter_coeff291_b,rf_filter_coeff292_a,
rf_filter_coeff292_b,
rf_filter_coeff293_a, rf_filter_coeff293_b,rf_filter_coeff294_a,
rf_filter_coeff294_b,rf_filter_coeff295_a, rf_filter_coeff295_b,rf_filter_coeff296_a,
rf_filter_coeff296_b,
rf_filter_coeff297_a, rf_filter_coeff297_b,rf_filter_coeff298_a,
rf_filter_coeff298_b,rf_filter_coeff299_a, rf_filter_coeff299_b,rf_filter_coeff300_a,
rf_filter_coeff300_b,
```

rf_filter_coeff301_a, rf_filter_coeff301_b,rf_filter_coeff302_a,
rf_filter_coeff302_b,rf_filter_coeff303_a, rf_filter_coeff303_b,rf_filter_coeff304_a,
rf_filter_coeff304_b,
rf_filter_coeff305_a, rf_filter_coeff305_b,rf_filter_coeff306_a,
rf_filter_coeff306_b,rf_filter_coeff307_a, rf_filter_coeff307_b,rf_filter_coeff308_a,
rf_filter_coeff308_b,
rf_filter_coeff309_a, rf_filter_coeff309_b,rf_filter_coeff310_a,
rf_filter_coeff310_b,rf_filter_coeff311_a, rf_filter_coeff311_b,rf_filter_coeff312_a,
rf_filter_coeff312_b,
rf_filter_coeff313_a, rf_filter_coeff313_b,rf_filter_coeff314_a,
rf_filter_coeff314_b,rf_filter_coeff315_a, rf_filter_coeff315_b,rf_filter_coeff316_a,
rf_filter_coeff316_b,
rf_filter_coeff317_a, rf_filter_coeff317_b,rf_filter_coeff318_a,
rf_filter_coeff318_b,rf_filter_coeff319_a, rf_filter_coeff319_b,rf_filter_coeff320_a,
rf_filter_coeff320_b,
rf_filter_coeff321_a, rf_filter_coeff321_b,rf_filter_coeff322_a,
rf_filter_coeff322_b,rf_filter_coeff323_a, rf_filter_coeff323_b,rf_filter_coeff324_a,
rf_filter_coeff324_b,
rf_filter_coeff325_a, rf_filter_coeff325_b,rf_filter_coeff326_a,
rf_filter_coeff326_b,rf_filter_coeff327_a, rf_filter_coeff327_b,rf_filter_coeff328_a,
rf_filter_coeff328_b,
rf_filter_coeff329_a, rf_filter_coeff329_b,rf_filter_coeff330_a,
rf_filter_coeff330_b,rf_filter_coeff331_a, rf_filter_coeff331_b,rf_filter_coeff332_a,
rf_filter_coeff332_b,
rf_filter_coeff333_a, rf_filter_coeff333_b,rf_filter_coeff334_a,
rf_filter_coeff334_b,rf_filter_coeff335_a, rf_filter_coeff335_b,rf_filter_coeff336_a,
rf_filter_coeff336_b,
rf_filter_coeff337_a, rf_filter_coeff337_b,rf_filter_coeff338_a,
rf_filter_coeff338_b,rf_filter_coeff339_a, rf_filter_coeff339_b,rf_filter_coeff340_a,
rf_filter_coeff340_b,
rf_filter_coeff341_a, rf_filter_coeff341_b,rf_filter_coeff342_a,
rf_filter_coeff342_b,rf_filter_coeff343_a, rf_filter_coeff343_b,rf_filter_coeff344_a,
rf_filter_coeff344_b,
rf_filter_coeff345_a, rf_filter_coeff345_b,rf_filter_coeff346_a,
rf_filter_coeff346_b,rf_filter_coeff347_a, rf_filter_coeff347_b,rf_filter_coeff348_a,
rf_filter_coeff348_b,
rf_filter_coeff349_a, rf_filter_coeff349_b,rf_filter_coeff350_a,
rf_filter_coeff350_b,rf_filter_coeff351_a, rf_filter_coeff351_b,rf_filter_coeff352_a,
rf_filter_coeff352_b,
rf_filter_coeff353_a, rf_filter_coeff353_b,rf_filter_coeff354_a,
rf_filter_coeff354_b,rf_filter_coeff355_a, rf_filter_coeff355_b,rf_filter_coeff356_a,
rf_filter_coeff356_b,
rf_filter_coeff357_a, rf_filter_coeff357_b,rf_filter_coeff358_a,
rf_filter_coeff358_b,rf_filter_coeff359_a, rf_filter_coeff359_b,rf_filter_coeff360_a,
rf_filter_coeff360_b,
rf_filter_coeff361_a, rf_filter_coeff361_b,rf_filter_coeff362_a,
rf_filter_coeff362_b,rf_filter_coeff363_a, rf_filter_coeff363_b,rf_filter_coeff364_a,
rf_filter_coeff364_b,
rf_filter_coeff365_a, rf_filter_coeff365_b,rf_filter_coeff366_a,
rf_filter_coeff366_b,rf_filter_coeff367_a, rf_filter_coeff367_b,rf_filter_coeff368_a,
rf_filter_coeff368_b,
rf_filter_coeff369_a, rf_filter_coeff369_b,rf_filter_coeff370_a,
rf_filter_coeff370_b,rf_filter_coeff371_a, rf_filter_coeff371_b,rf_filter_coeff372_a,
rf_filter_coeff372_b,
rf_filter_coeff373_a, rf_filter_coeff373_b,rf_filter_coeff374_a,
rf_filter_coeff374_b,rf_filter_coeff375_a, rf_filter_coeff375_b,rf_filter_coeff376_a,
rf_filter_coeff376_b,
rf_filter_coeff377_a, rf_filter_coeff377_b,rf_filter_coeff378_a,
rf_filter_coeff378_b,rf_filter_coeff379_a, rf_filter_coeff379_b,rf_filter_coeff380_a,
rf_filter_coeff380_b,
rf_filter_coeff381_a, rf_filter_coeff381_b,rf_filter_coeff382_a,
rf_filter_coeff382_b,rf_filter_coeff383_a, rf_filter_coeff383_b,rf_filter_coeff384_a,
rf_filter_coeff384_b,
rf_filter_coeff385_a, rf_filter_coeff385_b,rf_filter_coeff386_a,
rf_filter_coeff386_b,rf_filter_coeff387_a, rf_filter_coeff387_b,rf_filter_coeff388_a,
rf_filter_coeff388_b,
rf_filter_coeff389_a, rf_filter_coeff389_b,rf_filter_coeff390_a,
rf_filter_coeff390_b,rf_filter_coeff391_a, rf_filter_coeff391_b,rf_filter_coeff392_a,
rf_filter_coeff392_b,

rf_filter_coeff393_a, rf_filter_coeff393_b,rf_filter_coeff394_a,
rf_filter_coeff394_b,rf_filter_coeff395_a, rf_filter_coeff395_b,rf_filter_coeff396_a,
rf_filter_coeff396_b,
rf_filter_coeff397_a, rf_filter_coeff397_b,rf_filter_coeff398_a,
rf_filter_coeff398_b,rf_filter_coeff399_a, rf_filter_coeff399_b,rf_filter_coeff400_a,
rf_filter_coeff400_b,
rf_filter_coeff401_a, rf_filter_coeff401_b,rf_filter_coeff402_a,
rf_filter_coeff402_b,rf_filter_coeff403_a, rf_filter_coeff403_b,rf_filter_coeff404_a,
rf_filter_coeff404_b,
rf_filter_coeff405_a, rf_filter_coeff405_b,rf_filter_coeff406_a,
rf_filter_coeff406_b,rf_filter_coeff407_a, rf_filter_coeff407_b,rf_filter_coeff408_a,
rf_filter_coeff408_b,
rf_filter_coeff409_a, rf_filter_coeff409_b,rf_filter_coeff410_a,
rf_filter_coeff410_b,rf_filter_coeff411_a, rf_filter_coeff411_b,rf_filter_coeff412_a,
rf_filter_coeff412_b,
rf_filter_coeff413_a, rf_filter_coeff413_b,rf_filter_coeff414_a,
rf_filter_coeff414_b,rf_filter_coeff415_a, rf_filter_coeff415_b,rf_filter_coeff416_a,
rf_filter_coeff416_b,
rf_filter_coeff417_a, rf_filter_coeff417_b,rf_filter_coeff418_a,
rf_filter_coeff418_b,rf_filter_coeff419_a, rf_filter_coeff419_b,rf_filter_coeff420_a,
rf_filter_coeff420_b,
rf_filter_coeff421_a, rf_filter_coeff421_b,rf_filter_coeff422_a,
rf_filter_coeff422_b,rf_filter_coeff423_a, rf_filter_coeff423_b,rf_filter_coeff424_a,
rf_filter_coeff424_b,
rf_filter_coeff425_a, rf_filter_coeff425_b,rf_filter_coeff426_a,
rf_filter_coeff426_b,rf_filter_coeff427_a, rf_filter_coeff427_b,rf_filter_coeff428_a,
rf_filter_coeff428_b,
rf_filter_coeff429_a, rf_filter_coeff429_b,rf_filter_coeff430_a,
rf_filter_coeff430_b,rf_filter_coeff431_a, rf_filter_coeff431_b,rf_filter_coeff432_a,
rf_filter_coeff432_b,
rf_filter_coeff433_a, rf_filter_coeff433_b,rf_filter_coeff434_a,
rf_filter_coeff434_b,rf_filter_coeff435_a, rf_filter_coeff435_b,rf_filter_coeff436_a,
rf_filter_coeff436_b,
rf_filter_coeff437_a, rf_filter_coeff437_b,rf_filter_coeff438_a,
rf_filter_coeff438_b,rf_filter_coeff439_a, rf_filter_coeff439_b,rf_filter_coeff440_a,
rf_filter_coeff440_b,
rf_filter_coeff441_a, rf_filter_coeff441_b,rf_filter_coeff442_a,
rf_filter_coeff442_b,rf_filter_coeff443_a, rf_filter_coeff443_b,rf_filter_coeff444_a,
rf_filter_coeff444_b,
rf_filter_coeff445_a, rf_filter_coeff445_b,rf_filter_coeff446_a,
rf_filter_coeff446_b,rf_filter_coeff447_a, rf_filter_coeff447_b,rf_filter_coeff448_a,
rf_filter_coeff448_b,
rf_filter_coeff449_a, rf_filter_coeff449_b,rf_filter_coeff450_a,
rf_filter_coeff450_b,rf_filter_coeff451_a, rf_filter_coeff451_b,rf_filter_coeff452_a,
rf_filter_coeff452_b,
rf_filter_coeff453_a, rf_filter_coeff453_b,rf_filter_coeff454_a,
rf_filter_coeff454_b,rf_filter_coeff455_a, rf_filter_coeff455_b,rf_filter_coeff456_a,
rf_filter_coeff456_b,
rf_filter_coeff457_a, rf_filter_coeff457_b,rf_filter_coeff458_a,
rf_filter_coeff458_b,rf_filter_coeff459_a, rf_filter_coeff459_b,rf_filter_coeff460_a,
rf_filter_coeff460_b,
rf_filter_coeff461_a, rf_filter_coeff461_b,rf_filter_coeff462_a,
rf_filter_coeff462_b,rf_filter_coeff463_a, rf_filter_coeff463_b,rf_filter_coeff464_a,
rf_filter_coeff464_b,
rf_filter_coeff465_a, rf_filter_coeff465_b,rf_filter_coeff466_a,
rf_filter_coeff466_b,rf_filter_coeff467_a, rf_filter_coeff467_b,rf_filter_coeff468_a,
rf_filter_coeff468_b,
rf_filter_coeff469_a, rf_filter_coeff469_b,rf_filter_coeff470_a,
rf_filter_coeff470_b,rf_filter_coeff471_a, rf_filter_coeff471_b,rf_filter_coeff472_a,
rf_filter_coeff472_b,
rf_filter_coeff473_a, rf_filter_coeff473_b,rf_filter_coeff474_a,
rf_filter_coeff474_b,rf_filter_coeff475_a, rf_filter_coeff475_b,rf_filter_coeff476_a,
rf_filter_coeff476_b,
rf_filter_coeff477_a, rf_filter_coeff477_b,rf_filter_coeff478_a,
rf_filter_coeff478_b,rf_filter_coeff479_a, rf_filter_coeff479_b,rf_filter_coeff480_a,
rf_filter_coeff480_b,
rf_filter_coeff481_a, rf_filter_coeff481_b,rf_filter_coeff482_a,
rf_filter_coeff482_b,rf_filter_coeff483_a, rf_filter_coeff483_b,rf_filter_coeff484_a,
rf_filter_coeff484_b,

```
rf_filter_coeff485_a, rf_filter_coeff485_b,rf_filter_coeff486_a,
rf_filter_coeff486_b,rf_filter_coeff487_a, rf_filter_coeff487_b,rf_filter_coeff488_a,
rf_filter_coeff488_b,
rf_filter_coeff489_a, rf_filter_coeff489_b,rf_filter_coeff490_a,
rf_filter_coeff490_b,rf_filter_coeff491_a, rf_filter_coeff491_b,rf_filter_coeff492_a,
rf_filter_coeff492_b,
rf_filter_coeff493_a, rf_filter_coeff493_b,rf_filter_coeff494_a,
rf_filter_coeff494_b,rf_filter_coeff495_a, rf_filter_coeff495_b,rf_filter_coeff496_a,
rf_filter_coeff496_b,
rf_filter_coeff497_a, rf_filter_coeff497_b,rf_filter_coeff498_a,
rf_filter_coeff498_b,rf_filter_coeff499_a, rf_filter_coeff499_b,rf_filter_coeff500_a,
rf_filter_coeff500_b,
rf_filter_coeff501_a, rf_filter_coeff501_b,rf_filter_coeff502_a,
rf_filter_coeff502_b,rf_filter_coeff503_a, rf_filter_coeff503_b,rf_filter_coeff504_a,
rf_filter_coeff504_b,
rf_filter_coeff505_a, rf_filter_coeff505_b,rf_filter_coeff506_a,
rf_filter_coeff506_b,rf_filter_coeff507_a, rf_filter_coeff507_b,rf_filter_coeff508_a,
rf_filter_coeff508_b,
rf_filter_coeff509_a, rf_filter_coeff509_b,rf_filter_coeff510_a,
rf_filter_coeff510_b,rf_filter_coeff511_a, rf_filter_coeff511_b;

    input                                                                   clk;
    input                                                                   rden;       // input:
    input   [8:0]                                       rdptr;      // input:
    output          [15:0]                                                  rddata;     //
output:
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------


localparam          DEPTH = 511; //2^9 = 512
localparam          WIDTH = 15;

wire      [WIDTH:0] ram [DEPTH:0];
reg       [WIDTH:0] rddata;


assign ram[0] = {rf_filter_coeff0_a, rf_filter_coeff0_b};
assign ram[1] = {rf_filter_coeff1_a, rf_filter_coeff1_b};
assign ram[2] = {rf_filter_coeff2_a, rf_filter_coeff2_b};
assign ram[3] = {rf_filter_coeff3_a, rf_filter_coeff3_b};
assign ram[4] = {rf_filter_coeff4_a, rf_filter_coeff4_b};
assign ram[5] = {rf_filter_coeff5_a, rf_filter_coeff5_b};
assign ram[6] = {rf_filter_coeff6_a, rf_filter_coeff6_b};
assign ram[7] = {rf_filter_coeff7_a, rf_filter_coeff7_b};
assign ram[8] = {rf_filter_coeff8_a, rf_filter_coeff8_b};
assign ram[9] = {rf_filter_coeff9_a, rf_filter_coeff9_b};
assign ram[10] = {rf_filter_coeff10_a, rf_filter_coeff10_b};
assign ram[11] = {rf_filter_coeff11_a, rf_filter_coeff11_b};
assign ram[12] = {rf_filter_coeff12_a, rf_filter_coeff12_b};
assign ram[13] = {rf_filter_coeff13_a, rf_filter_coeff13_b};
assign ram[14] = {rf_filter_coeff14_a, rf_filter_coeff14_b};
assign ram[15] = {rf_filter_coeff15_a, rf_filter_coeff15_b};
assign ram[16] = {rf_filter_coeff16_a, rf_filter_coeff16_b};
assign ram[17] = {rf_filter_coeff17_a, rf_filter_coeff17_b};
assign ram[18] = {rf_filter_coeff18_a, rf_filter_coeff18_b};
assign ram[19] = {rf_filter_coeff19_a, rf_filter_coeff19_b};
assign ram[20] = {rf_filter_coeff20_a, rf_filter_coeff20_b};
assign ram[21] = {rf_filter_coeff21_a, rf_filter_coeff21_b};
assign ram[22] = {rf_filter_coeff22_a, rf_filter_coeff22_b};
assign ram[23] = {rf_filter_coeff23_a, rf_filter_coeff23_b};
assign ram[24] = {rf_filter_coeff24_a, rf_filter_coeff24_b};
assign ram[25] = {rf_filter_coeff25_a, rf_filter_coeff25_b};
assign ram[26] = {rf_filter_coeff26_a, rf_filter_coeff26_b};
assign ram[27] = {rf_filter_coeff27_a, rf_filter_coeff27_b};
assign ram[28] = {rf_filter_coeff28_a, rf_filter_coeff28_b};
assign ram[29] = {rf_filter_coeff29_a, rf_filter_coeff29_b};
assign ram[30] = {rf_filter_coeff30_a, rf_filter_coeff30_b};
assign ram[31] = {rf_filter_coeff31_a, rf_filter_coeff31_b};
assign ram[32] = {rf_filter_coeff32_a, rf_filter_coeff32_b};
assign ram[33] = {rf_filter_coeff33_a, rf_filter_coeff33_b};
```

```
assign ram[34] = {rf_filter_coeff34_a, rf_filter_coeff34_b};
assign ram[35] = {rf_filter_coeff35_a, rf_filter_coeff35_b};
assign ram[36] = {rf_filter_coeff36_a, rf_filter_coeff36_b};
assign ram[37] = {rf_filter_coeff37_a, rf_filter_coeff37_b};
assign ram[38] = {rf_filter_coeff38_a, rf_filter_coeff38_b};
assign ram[39] = {rf_filter_coeff39_a, rf_filter_coeff39_b};
assign ram[40] = {rf_filter_coeff40_a, rf_filter_coeff40_b};
assign ram[41] = {rf_filter_coeff41_a, rf_filter_coeff41_b};
assign ram[42] = {rf_filter_coeff42_a, rf_filter_coeff42_b};
assign ram[43] = {rf_filter_coeff43_a, rf_filter_coeff43_b};
assign ram[44] = {rf_filter_coeff44_a, rf_filter_coeff44_b};
assign ram[45] = {rf_filter_coeff45_a, rf_filter_coeff45_b};
assign ram[46] = {rf_filter_coeff46_a, rf_filter_coeff46_b};
assign ram[47] = {rf_filter_coeff47_a, rf_filter_coeff47_b};
assign ram[48] = {rf_filter_coeff48_a, rf_filter_coeff48_b};
assign ram[49] = {rf_filter_coeff49_a, rf_filter_coeff49_b};
assign ram[50] = {rf_filter_coeff50_a, rf_filter_coeff50_b};
assign ram[51] = {rf_filter_coeff51_a, rf_filter_coeff51_b};
assign ram[52] = {rf_filter_coeff52_a, rf_filter_coeff52_b};
assign ram[53] = {rf_filter_coeff53_a, rf_filter_coeff53_b};
assign ram[54] = {rf_filter_coeff54_a, rf_filter_coeff54_b};
assign ram[55] = {rf_filter_coeff55_a, rf_filter_coeff55_b};
assign ram[56] = {rf_filter_coeff56_a, rf_filter_coeff56_b};
assign ram[57] = {rf_filter_coeff57_a, rf_filter_coeff57_b};
assign ram[58] = {rf_filter_coeff58_a, rf_filter_coeff58_b};
assign ram[59] = {rf_filter_coeff59_a, rf_filter_coeff59_b};
assign ram[60] = {rf_filter_coeff60_a, rf_filter_coeff60_b};
assign ram[61] = {rf_filter_coeff61_a, rf_filter_coeff61_b};
assign ram[62] = {rf_filter_coeff62_a, rf_filter_coeff62_b};
assign ram[63] = {rf_filter_coeff63_a, rf_filter_coeff63_b};
assign ram[64] = {rf_filter_coeff64_a, rf_filter_coeff64_b};
assign ram[65] = {rf_filter_coeff65_a, rf_filter_coeff65_b};
assign ram[66] = {rf_filter_coeff66_a, rf_filter_coeff66_b};
assign ram[67] = {rf_filter_coeff67_a, rf_filter_coeff67_b};
assign ram[68] = {rf_filter_coeff68_a, rf_filter_coeff68_b};
assign ram[69] = {rf_filter_coeff69_a, rf_filter_coeff69_b};
assign ram[70] = {rf_filter_coeff70_a, rf_filter_coeff70_b};
assign ram[71] = {rf_filter_coeff71_a, rf_filter_coeff71_b};
assign ram[72] = {rf_filter_coeff72_a, rf_filter_coeff72_b};
assign ram[73] = {rf_filter_coeff73_a, rf_filter_coeff73_b};
assign ram[74] = {rf_filter_coeff74_a, rf_filter_coeff74_b};
assign ram[75] = {rf_filter_coeff75_a, rf_filter_coeff75_b};
assign ram[76] = {rf_filter_coeff76_a, rf_filter_coeff76_b};
assign ram[77] = {rf_filter_coeff77_a, rf_filter_coeff77_b};
assign ram[78] = {rf_filter_coeff78_a, rf_filter_coeff78_b};
assign ram[79] = {rf_filter_coeff79_a, rf_filter_coeff79_b};
assign ram[80] = {rf_filter_coeff80_a, rf_filter_coeff80_b};
assign ram[81] = {rf_filter_coeff81_a, rf_filter_coeff81_b};
assign ram[82] = {rf_filter_coeff82_a, rf_filter_coeff82_b};
assign ram[83] = {rf_filter_coeff83_a, rf_filter_coeff83_b};
assign ram[84] = {rf_filter_coeff84_a, rf_filter_coeff84_b};
assign ram[85] = {rf_filter_coeff85_a, rf_filter_coeff85_b};
assign ram[86] = {rf_filter_coeff86_a, rf_filter_coeff86_b};
assign ram[87] = {rf_filter_coeff87_a, rf_filter_coeff87_b};
assign ram[88] = {rf_filter_coeff88_a, rf_filter_coeff88_b};
assign ram[89] = {rf_filter_coeff89_a, rf_filter_coeff89_b};
assign ram[90] = {rf_filter_coeff90_a, rf_filter_coeff90_b};
assign ram[91] = {rf_filter_coeff91_a, rf_filter_coeff91_b};
assign ram[92] = {rf_filter_coeff92_a, rf_filter_coeff92_b};
assign ram[93] = {rf_filter_coeff93_a, rf_filter_coeff93_b};
assign ram[94] = {rf_filter_coeff94_a, rf_filter_coeff94_b};
assign ram[95] = {rf_filter_coeff95_a, rf_filter_coeff95_b};
assign ram[96] = {rf_filter_coeff96_a, rf_filter_coeff96_b};
assign ram[97] = {rf_filter_coeff97_a, rf_filter_coeff97_b};
assign ram[98] = {rf_filter_coeff98_a, rf_filter_coeff98_b};
assign ram[99] = {rf_filter_coeff99_a, rf_filter_coeff99_b};
assign ram[100] = {rf_filter_coeff100_a, rf_filter_coeff100_b};
assign ram[101] = {rf_filter_coeff101_a, rf_filter_coeff101_b};
assign ram[102] = {rf_filter_coeff102_a, rf_filter_coeff102_b};
assign ram[103] = {rf_filter_coeff103_a, rf_filter_coeff103_b};
assign ram[104] = {rf_filter_coeff104_a, rf_filter_coeff104_b};
```

```
assign ram[105] = {rf_filter_coeff105_a, rf_filter_coeff105_b};
assign ram[106] = {rf_filter_coeff106_a, rf_filter_coeff106_b};
assign ram[107] = {rf_filter_coeff107_a, rf_filter_coeff107_b};
assign ram[108] = {rf_filter_coeff108_a, rf_filter_coeff108_b};
assign ram[109] = {rf_filter_coeff109_a, rf_filter_coeff109_b};
assign ram[110] = {rf_filter_coeff110_a, rf_filter_coeff110_b};
assign ram[111] = {rf_filter_coeff111_a, rf_filter_coeff111_b};
assign ram[112] = {rf_filter_coeff112_a, rf_filter_coeff112_b};
assign ram[113] = {rf_filter_coeff113_a, rf_filter_coeff113_b};
assign ram[114] = {rf_filter_coeff114_a, rf_filter_coeff114_b};
assign ram[115] = {rf_filter_coeff115_a, rf_filter_coeff115_b};
assign ram[116] = {rf_filter_coeff116_a, rf_filter_coeff116_b};
assign ram[117] = {rf_filter_coeff117_a, rf_filter_coeff117_b};
assign ram[118] = {rf_filter_coeff118_a, rf_filter_coeff118_b};
assign ram[119] = {rf_filter_coeff119_a, rf_filter_coeff119_b};
assign ram[120] = {rf_filter_coeff120_a, rf_filter_coeff120_b};
assign ram[121] = {rf_filter_coeff121_a, rf_filter_coeff121_b};
assign ram[122] = {rf_filter_coeff122_a, rf_filter_coeff122_b};
assign ram[123] = {rf_filter_coeff123_a, rf_filter_coeff123_b};
assign ram[124] = {rf_filter_coeff124_a, rf_filter_coeff124_b};
assign ram[125] = {rf_filter_coeff125_a, rf_filter_coeff125_b};
assign ram[126] = {rf_filter_coeff126_a, rf_filter_coeff126_b};
assign ram[127] = {rf_filter_coeff127_a, rf_filter_coeff127_b};
assign ram[128] = {rf_filter_coeff128_a, rf_filter_coeff128_b};
assign ram[129] = {rf_filter_coeff129_a, rf_filter_coeff129_b};
assign ram[130] = {rf_filter_coeff130_a, rf_filter_coeff130_b};
assign ram[131] = {rf_filter_coeff131_a, rf_filter_coeff131_b};
assign ram[132] = {rf_filter_coeff132_a, rf_filter_coeff132_b};
assign ram[133] = {rf_filter_coeff133_a, rf_filter_coeff133_b};
assign ram[134] = {rf_filter_coeff134_a, rf_filter_coeff134_b};
assign ram[135] = {rf_filter_coeff135_a, rf_filter_coeff135_b};
assign ram[136] = {rf_filter_coeff136_a, rf_filter_coeff136_b};
assign ram[137] = {rf_filter_coeff137_a, rf_filter_coeff137_b};
assign ram[138] = {rf_filter_coeff138_a, rf_filter_coeff138_b};
assign ram[139] = {rf_filter_coeff139_a, rf_filter_coeff139_b};
assign ram[140] = {rf_filter_coeff140_a, rf_filter_coeff140_b};
assign ram[141] = {rf_filter_coeff141_a, rf_filter_coeff141_b};
assign ram[142] = {rf_filter_coeff142_a, rf_filter_coeff142_b};
assign ram[143] = {rf_filter_coeff143_a, rf_filter_coeff143_b};
assign ram[144] = {rf_filter_coeff144_a, rf_filter_coeff144_b};
assign ram[145] = {rf_filter_coeff145_a, rf_filter_coeff145_b};
assign ram[146] = {rf_filter_coeff146_a, rf_filter_coeff146_b};
assign ram[147] = {rf_filter_coeff147_a, rf_filter_coeff147_b};
assign ram[148] = {rf_filter_coeff148_a, rf_filter_coeff148_b};
assign ram[149] = {rf_filter_coeff149_a, rf_filter_coeff149_b};
assign ram[150] = {rf_filter_coeff150_a, rf_filter_coeff150_b};
assign ram[151] = {rf_filter_coeff151_a, rf_filter_coeff151_b};
assign ram[152] = {rf_filter_coeff152_a, rf_filter_coeff152_b};
assign ram[153] = {rf_filter_coeff153_a, rf_filter_coeff153_b};
assign ram[154] = {rf_filter_coeff154_a, rf_filter_coeff154_b};
assign ram[155] = {rf_filter_coeff155_a, rf_filter_coeff155_b};
assign ram[156] = {rf_filter_coeff156_a, rf_filter_coeff156_b};
assign ram[157] = {rf_filter_coeff157_a, rf_filter_coeff157_b};
assign ram[158] = {rf_filter_coeff158_a, rf_filter_coeff158_b};
assign ram[159] = {rf_filter_coeff159_a, rf_filter_coeff159_b};
assign ram[160] = {rf_filter_coeff160_a, rf_filter_coeff160_b};
assign ram[161] = {rf_filter_coeff161_a, rf_filter_coeff161_b};
assign ram[162] = {rf_filter_coeff162_a, rf_filter_coeff162_b};
assign ram[163] = {rf_filter_coeff163_a, rf_filter_coeff163_b};
assign ram[164] = {rf_filter_coeff164_a, rf_filter_coeff164_b};
assign ram[165] = {rf_filter_coeff165_a, rf_filter_coeff165_b};
assign ram[166] = {rf_filter_coeff166_a, rf_filter_coeff166_b};
assign ram[167] = {rf_filter_coeff167_a, rf_filter_coeff167_b};
assign ram[168] = {rf_filter_coeff168_a, rf_filter_coeff168_b};
assign ram[169] = {rf_filter_coeff169_a, rf_filter_coeff169_b};
assign ram[170] = {rf_filter_coeff170_a, rf_filter_coeff170_b};
assign ram[171] = {rf_filter_coeff171_a, rf_filter_coeff171_b};
assign ram[172] = {rf_filter_coeff172_a, rf_filter_coeff172_b};
assign ram[173] = {rf_filter_coeff173_a, rf_filter_coeff173_b};
assign ram[174] = {rf_filter_coeff174_a, rf_filter_coeff174_b};
assign ram[175] = {rf_filter_coeff175_a, rf_filter_coeff175_b};
```

```
assign ram[176] = {rf_filter_coeff176_a, rf_filter_coeff176_b};
assign ram[177] = {rf_filter_coeff177_a, rf_filter_coeff177_b};
assign ram[178] = {rf_filter_coeff178_a, rf_filter_coeff178_b};
assign ram[179] = {rf_filter_coeff179_a, rf_filter_coeff179_b};
assign ram[180] = {rf_filter_coeff180_a, rf_filter_coeff180_b};
assign ram[181] = {rf_filter_coeff181_a, rf_filter_coeff181_b};
assign ram[182] = {rf_filter_coeff182_a, rf_filter_coeff182_b};
assign ram[183] = {rf_filter_coeff183_a, rf_filter_coeff183_b};
assign ram[184] = {rf_filter_coeff184_a, rf_filter_coeff184_b};
assign ram[185] = {rf_filter_coeff185_a, rf_filter_coeff185_b};
assign ram[186] = {rf_filter_coeff186_a, rf_filter_coeff186_b};
assign ram[187] = {rf_filter_coeff187_a, rf_filter_coeff187_b};
assign ram[188] = {rf_filter_coeff188_a, rf_filter_coeff188_b};
assign ram[189] = {rf_filter_coeff189_a, rf_filter_coeff189_b};
assign ram[190] = {rf_filter_coeff190_a, rf_filter_coeff190_b};
assign ram[191] = {rf_filter_coeff191_a, rf_filter_coeff191_b};
assign ram[192] = {rf_filter_coeff192_a, rf_filter_coeff192_b};
assign ram[193] = {rf_filter_coeff193_a, rf_filter_coeff193_b};
assign ram[194] = {rf_filter_coeff194_a, rf_filter_coeff194_b};
assign ram[195] = {rf_filter_coeff195_a, rf_filter_coeff195_b};
assign ram[196] = {rf_filter_coeff196_a, rf_filter_coeff196_b};
assign ram[197] = {rf_filter_coeff197_a, rf_filter_coeff197_b};
assign ram[198] = {rf_filter_coeff198_a, rf_filter_coeff198_b};
assign ram[199] = {rf_filter_coeff199_a, rf_filter_coeff199_b};
assign ram[200] = {rf_filter_coeff200_a, rf_filter_coeff200_b};
assign ram[201] = {rf_filter_coeff201_a, rf_filter_coeff201_b};
assign ram[202] = {rf_filter_coeff202_a, rf_filter_coeff202_b};
assign ram[203] = {rf_filter_coeff203_a, rf_filter_coeff203_b};
assign ram[204] = {rf_filter_coeff204_a, rf_filter_coeff204_b};
assign ram[205] = {rf_filter_coeff205_a, rf_filter_coeff205_b};
assign ram[206] = {rf_filter_coeff206_a, rf_filter_coeff206_b};
assign ram[207] = {rf_filter_coeff207_a, rf_filter_coeff207_b};
assign ram[208] = {rf_filter_coeff208_a, rf_filter_coeff208_b};
assign ram[209] = {rf_filter_coeff209_a, rf_filter_coeff209_b};
assign ram[210] = {rf_filter_coeff210_a, rf_filter_coeff210_b};
assign ram[211] = {rf_filter_coeff211_a, rf_filter_coeff211_b};
assign ram[212] = {rf_filter_coeff212_a, rf_filter_coeff212_b};
assign ram[213] = {rf_filter_coeff213_a, rf_filter_coeff213_b};
assign ram[214] = {rf_filter_coeff214_a, rf_filter_coeff214_b};
assign ram[215] = {rf_filter_coeff215_a, rf_filter_coeff215_b};
assign ram[216] = {rf_filter_coeff216_a, rf_filter_coeff216_b};
assign ram[217] = {rf_filter_coeff217_a, rf_filter_coeff217_b};
assign ram[218] = {rf_filter_coeff218_a, rf_filter_coeff218_b};
assign ram[219] = {rf_filter_coeff219_a, rf_filter_coeff219_b};
assign ram[220] = {rf_filter_coeff220_a, rf_filter_coeff220_b};
assign ram[221] = {rf_filter_coeff221_a, rf_filter_coeff221_b};
assign ram[222] = {rf_filter_coeff222_a, rf_filter_coeff222_b};
assign ram[223] = {rf_filter_coeff223_a, rf_filter_coeff223_b};
assign ram[224] = {rf_filter_coeff224_a, rf_filter_coeff224_b};
assign ram[225] = {rf_filter_coeff225_a, rf_filter_coeff225_b};
assign ram[226] = {rf_filter_coeff226_a, rf_filter_coeff226_b};
assign ram[227] = {rf_filter_coeff227_a, rf_filter_coeff227_b};
assign ram[228] = {rf_filter_coeff228_a, rf_filter_coeff228_b};
assign ram[229] = {rf_filter_coeff229_a, rf_filter_coeff229_b};
assign ram[230] = {rf_filter_coeff230_a, rf_filter_coeff230_b};
assign ram[231] = {rf_filter_coeff231_a, rf_filter_coeff231_b};
assign ram[232] = {rf_filter_coeff232_a, rf_filter_coeff232_b};
assign ram[233] = {rf_filter_coeff233_a, rf_filter_coeff233_b};
assign ram[234] = {rf_filter_coeff234_a, rf_filter_coeff234_b};
assign ram[235] = {rf_filter_coeff235_a, rf_filter_coeff235_b};
assign ram[236] = {rf_filter_coeff236_a, rf_filter_coeff236_b};
assign ram[237] = {rf_filter_coeff237_a, rf_filter_coeff237_b};
assign ram[238] = {rf_filter_coeff238_a, rf_filter_coeff238_b};
assign ram[239] = {rf_filter_coeff239_a, rf_filter_coeff239_b};
assign ram[240] = {rf_filter_coeff240_a, rf_filter_coeff240_b};
assign ram[241] = {rf_filter_coeff241_a, rf_filter_coeff241_b};
assign ram[242] = {rf_filter_coeff242_a, rf_filter_coeff242_b};
assign ram[243] = {rf_filter_coeff243_a, rf_filter_coeff243_b};
assign ram[244] = {rf_filter_coeff244_a, rf_filter_coeff244_b};
assign ram[245] = {rf_filter_coeff245_a, rf_filter_coeff245_b};
assign ram[246] = {rf_filter_coeff246_a, rf_filter_coeff246_b};
```

```
assign ram[247] = {rf_filter_coeff247_a, rf_filter_coeff247_b};
assign ram[248] = {rf_filter_coeff248_a, rf_filter_coeff248_b};
assign ram[249] = {rf_filter_coeff249_a, rf_filter_coeff249_b};
assign ram[250] = {rf_filter_coeff250_a, rf_filter_coeff250_b};
assign ram[251] = {rf_filter_coeff251_a, rf_filter_coeff251_b};
assign ram[252] = {rf_filter_coeff252_a, rf_filter_coeff252_b};
assign ram[253] = {rf_filter_coeff253_a, rf_filter_coeff253_b};
assign ram[254] = {rf_filter_coeff254_a, rf_filter_coeff254_b};
assign ram[255] = {rf_filter_coeff255_a, rf_filter_coeff255_b};
assign ram[256] = {rf_filter_coeff256_a, rf_filter_coeff256_b};
assign ram[257] = {rf_filter_coeff257_a, rf_filter_coeff257_b};
assign ram[258] = {rf_filter_coeff258_a, rf_filter_coeff258_b};
assign ram[259] = {rf_filter_coeff259_a, rf_filter_coeff259_b};
assign ram[260] = {rf_filter_coeff260_a, rf_filter_coeff260_b};
assign ram[261] = {rf_filter_coeff261_a, rf_filter_coeff261_b};
assign ram[262] = {rf_filter_coeff262_a, rf_filter_coeff262_b};
assign ram[263] = {rf_filter_coeff263_a, rf_filter_coeff263_b};
assign ram[264] = {rf_filter_coeff264_a, rf_filter_coeff264_b};
assign ram[265] = {rf_filter_coeff265_a, rf_filter_coeff265_b};
assign ram[266] = {rf_filter_coeff266_a, rf_filter_coeff266_b};
assign ram[267] = {rf_filter_coeff267_a, rf_filter_coeff267_b};
assign ram[268] = {rf_filter_coeff268_a, rf_filter_coeff268_b};
assign ram[269] = {rf_filter_coeff269_a, rf_filter_coeff269_b};
assign ram[270] = {rf_filter_coeff270_a, rf_filter_coeff270_b};
assign ram[271] = {rf_filter_coeff271_a, rf_filter_coeff271_b};
assign ram[272] = {rf_filter_coeff272_a, rf_filter_coeff272_b};
assign ram[273] = {rf_filter_coeff273_a, rf_filter_coeff273_b};
assign ram[274] = {rf_filter_coeff274_a, rf_filter_coeff274_b};
assign ram[275] = {rf_filter_coeff275_a, rf_filter_coeff275_b};
assign ram[276] = {rf_filter_coeff276_a, rf_filter_coeff276_b};
assign ram[277] = {rf_filter_coeff277_a, rf_filter_coeff277_b};
assign ram[278] = {rf_filter_coeff278_a, rf_filter_coeff278_b};
assign ram[279] = {rf_filter_coeff279_a, rf_filter_coeff279_b};
assign ram[280] = {rf_filter_coeff280_a, rf_filter_coeff280_b};
assign ram[281] = {rf_filter_coeff281_a, rf_filter_coeff281_b};
assign ram[282] = {rf_filter_coeff282_a, rf_filter_coeff282_b};
assign ram[283] = {rf_filter_coeff283_a, rf_filter_coeff283_b};
assign ram[284] = {rf_filter_coeff284_a, rf_filter_coeff284_b};
assign ram[285] = {rf_filter_coeff285_a, rf_filter_coeff285_b};
assign ram[286] = {rf_filter_coeff286_a, rf_filter_coeff286_b};
assign ram[287] = {rf_filter_coeff287_a, rf_filter_coeff287_b};
assign ram[288] = {rf_filter_coeff288_a, rf_filter_coeff288_b};
assign ram[289] = {rf_filter_coeff289_a, rf_filter_coeff289_b};
assign ram[290] = {rf_filter_coeff290_a, rf_filter_coeff290_b};
assign ram[291] = {rf_filter_coeff291_a, rf_filter_coeff291_b};
assign ram[292] = {rf_filter_coeff292_a, rf_filter_coeff292_b};
assign ram[293] = {rf_filter_coeff293_a, rf_filter_coeff293_b};
assign ram[294] = {rf_filter_coeff294_a, rf_filter_coeff294_b};
assign ram[295] = {rf_filter_coeff295_a, rf_filter_coeff295_b};
assign ram[296] = {rf_filter_coeff296_a, rf_filter_coeff296_b};
assign ram[297] = {rf_filter_coeff297_a, rf_filter_coeff297_b};
assign ram[298] = {rf_filter_coeff298_a, rf_filter_coeff298_b};
assign ram[299] = {rf_filter_coeff299_a, rf_filter_coeff299_b};
assign ram[300] = {rf_filter_coeff300_a, rf_filter_coeff300_b};
assign ram[301] = {rf_filter_coeff301_a, rf_filter_coeff301_b};
assign ram[302] = {rf_filter_coeff302_a, rf_filter_coeff302_b};
assign ram[303] = {rf_filter_coeff303_a, rf_filter_coeff303_b};
assign ram[304] = {rf_filter_coeff304_a, rf_filter_coeff304_b};
assign ram[305] = {rf_filter_coeff305_a, rf_filter_coeff305_b};
assign ram[306] = {rf_filter_coeff306_a, rf_filter_coeff306_b};
assign ram[307] = {rf_filter_coeff307_a, rf_filter_coeff307_b};
assign ram[308] = {rf_filter_coeff308_a, rf_filter_coeff308_b};
assign ram[309] = {rf_filter_coeff309_a, rf_filter_coeff309_b};
assign ram[310] = {rf_filter_coeff310_a, rf_filter_coeff310_b};
assign ram[311] = {rf_filter_coeff311_a, rf_filter_coeff311_b};
assign ram[312] = {rf_filter_coeff312_a, rf_filter_coeff312_b};
assign ram[313] = {rf_filter_coeff313_a, rf_filter_coeff313_b};
assign ram[314] = {rf_filter_coeff314_a, rf_filter_coeff314_b};
assign ram[315] = {rf_filter_coeff315_a, rf_filter_coeff315_b};
assign ram[316] = {rf_filter_coeff316_a, rf_filter_coeff316_b};
assign ram[317] = {rf_filter_coeff317_a, rf_filter_coeff317_b};
```

```
assign ram[318] = {rf_filter_coeff318_a, rf_filter_coeff318_b};
assign ram[319] = {rf_filter_coeff319_a, rf_filter_coeff319_b};
assign ram[320] = {rf_filter_coeff320_a, rf_filter_coeff320_b};
assign ram[321] = {rf_filter_coeff321_a, rf_filter_coeff321_b};
assign ram[322] = {rf_filter_coeff322_a, rf_filter_coeff322_b};
assign ram[323] = {rf_filter_coeff323_a, rf_filter_coeff323_b};
assign ram[324] = {rf_filter_coeff324_a, rf_filter_coeff324_b};
assign ram[325] = {rf_filter_coeff325_a, rf_filter_coeff325_b};
assign ram[326] = {rf_filter_coeff326_a, rf_filter_coeff326_b};
assign ram[327] = {rf_filter_coeff327_a, rf_filter_coeff327_b};
assign ram[328] = {rf_filter_coeff328_a, rf_filter_coeff328_b};
assign ram[329] = {rf_filter_coeff329_a, rf_filter_coeff329_b};
assign ram[330] = {rf_filter_coeff330_a, rf_filter_coeff330_b};
assign ram[331] = {rf_filter_coeff331_a, rf_filter_coeff331_b};
assign ram[332] = {rf_filter_coeff332_a, rf_filter_coeff332_b};
assign ram[333] = {rf_filter_coeff333_a, rf_filter_coeff333_b};
assign ram[334] = {rf_filter_coeff334_a, rf_filter_coeff334_b};
assign ram[335] = {rf_filter_coeff335_a, rf_filter_coeff335_b};
assign ram[336] = {rf_filter_coeff336_a, rf_filter_coeff336_b};
assign ram[337] = {rf_filter_coeff337_a, rf_filter_coeff337_b};
assign ram[338] = {rf_filter_coeff338_a, rf_filter_coeff338_b};
assign ram[339] = {rf_filter_coeff339_a, rf_filter_coeff339_b};
assign ram[340] = {rf_filter_coeff340_a, rf_filter_coeff340_b};
assign ram[341] = {rf_filter_coeff341_a, rf_filter_coeff341_b};
assign ram[342] = {rf_filter_coeff342_a, rf_filter_coeff342_b};
assign ram[343] = {rf_filter_coeff343_a, rf_filter_coeff343_b};
assign ram[344] = {rf_filter_coeff344_a, rf_filter_coeff344_b};
assign ram[345] = {rf_filter_coeff345_a, rf_filter_coeff345_b};
assign ram[346] = {rf_filter_coeff346_a, rf_filter_coeff346_b};
assign ram[347] = {rf_filter_coeff347_a, rf_filter_coeff347_b};
assign ram[348] = {rf_filter_coeff348_a, rf_filter_coeff348_b};
assign ram[349] = {rf_filter_coeff349_a, rf_filter_coeff349_b};
assign ram[350] = {rf_filter_coeff350_a, rf_filter_coeff350_b};
assign ram[351] = {rf_filter_coeff351_a, rf_filter_coeff351_b};
assign ram[352] = {rf_filter_coeff352_a, rf_filter_coeff352_b};
assign ram[353] = {rf_filter_coeff353_a, rf_filter_coeff353_b};
assign ram[354] = {rf_filter_coeff354_a, rf_filter_coeff354_b};
assign ram[355] = {rf_filter_coeff355_a, rf_filter_coeff355_b};
assign ram[356] = {rf_filter_coeff356_a, rf_filter_coeff356_b};
assign ram[357] = {rf_filter_coeff357_a, rf_filter_coeff357_b};
assign ram[358] = {rf_filter_coeff358_a, rf_filter_coeff358_b};
assign ram[359] = {rf_filter_coeff359_a, rf_filter_coeff359_b};
assign ram[360] = {rf_filter_coeff360_a, rf_filter_coeff360_b};
assign ram[361] = {rf_filter_coeff361_a, rf_filter_coeff361_b};
assign ram[362] = {rf_filter_coeff362_a, rf_filter_coeff362_b};
assign ram[363] = {rf_filter_coeff363_a, rf_filter_coeff363_b};
assign ram[364] = {rf_filter_coeff364_a, rf_filter_coeff364_b};
assign ram[365] = {rf_filter_coeff365_a, rf_filter_coeff365_b};
assign ram[366] = {rf_filter_coeff366_a, rf_filter_coeff366_b};
assign ram[367] = {rf_filter_coeff367_a, rf_filter_coeff367_b};
assign ram[368] = {rf_filter_coeff368_a, rf_filter_coeff368_b};
assign ram[369] = {rf_filter_coeff369_a, rf_filter_coeff369_b};
assign ram[370] = {rf_filter_coeff370_a, rf_filter_coeff370_b};
assign ram[371] = {rf_filter_coeff371_a, rf_filter_coeff371_b};
assign ram[372] = {rf_filter_coeff372_a, rf_filter_coeff372_b};
assign ram[373] = {rf_filter_coeff373_a, rf_filter_coeff373_b};
assign ram[374] = {rf_filter_coeff374_a, rf_filter_coeff374_b};
assign ram[375] = {rf_filter_coeff375_a, rf_filter_coeff375_b};
assign ram[376] = {rf_filter_coeff376_a, rf_filter_coeff376_b};
assign ram[377] = {rf_filter_coeff377_a, rf_filter_coeff377_b};
assign ram[378] = {rf_filter_coeff378_a, rf_filter_coeff378_b};
assign ram[379] = {rf_filter_coeff379_a, rf_filter_coeff379_b};
assign ram[380] = {rf_filter_coeff380_a, rf_filter_coeff380_b};
assign ram[381] = {rf_filter_coeff381_a, rf_filter_coeff381_b};
assign ram[382] = {rf_filter_coeff382_a, rf_filter_coeff382_b};
assign ram[383] = {rf_filter_coeff383_a, rf_filter_coeff383_b};
assign ram[384] = {rf_filter_coeff384_a, rf_filter_coeff384_b};
assign ram[385] = {rf_filter_coeff385_a, rf_filter_coeff385_b};
assign ram[386] = {rf_filter_coeff386_a, rf_filter_coeff386_b};
assign ram[387] = {rf_filter_coeff387_a, rf_filter_coeff387_b};
assign ram[388] = {rf_filter_coeff388_a, rf_filter_coeff388_b};
```

```
assign ram[389] = {rf_filter_coeff389_a, rf_filter_coeff389_b};
assign ram[390] = {rf_filter_coeff390_a, rf_filter_coeff390_b};
assign ram[391] = {rf_filter_coeff391_a, rf_filter_coeff391_b};
assign ram[392] = {rf_filter_coeff392_a, rf_filter_coeff392_b};
assign ram[393] = {rf_filter_coeff393_a, rf_filter_coeff393_b};
assign ram[394] = {rf_filter_coeff394_a, rf_filter_coeff394_b};
assign ram[395] = {rf_filter_coeff395_a, rf_filter_coeff395_b};
assign ram[396] = {rf_filter_coeff396_a, rf_filter_coeff396_b};
assign ram[397] = {rf_filter_coeff397_a, rf_filter_coeff397_b};
assign ram[398] = {rf_filter_coeff398_a, rf_filter_coeff398_b};
assign ram[399] = {rf_filter_coeff399_a, rf_filter_coeff399_b};
assign ram[400] = {rf_filter_coeff400_a, rf_filter_coeff400_b};
assign ram[401] = {rf_filter_coeff401_a, rf_filter_coeff401_b};
assign ram[402] = {rf_filter_coeff402_a, rf_filter_coeff402_b};
assign ram[403] = {rf_filter_coeff403_a, rf_filter_coeff403_b};
assign ram[404] = {rf_filter_coeff404_a, rf_filter_coeff404_b};
assign ram[405] = {rf_filter_coeff405_a, rf_filter_coeff405_b};
assign ram[406] = {rf_filter_coeff406_a, rf_filter_coeff406_b};
assign ram[407] = {rf_filter_coeff407_a, rf_filter_coeff407_b};
assign ram[408] = {rf_filter_coeff408_a, rf_filter_coeff408_b};
assign ram[409] = {rf_filter_coeff409_a, rf_filter_coeff409_b};
assign ram[410] = {rf_filter_coeff410_a, rf_filter_coeff410_b};
assign ram[411] = {rf_filter_coeff411_a, rf_filter_coeff411_b};
assign ram[412] = {rf_filter_coeff412_a, rf_filter_coeff412_b};
assign ram[413] = {rf_filter_coeff413_a, rf_filter_coeff413_b};
assign ram[414] = {rf_filter_coeff414_a, rf_filter_coeff414_b};
assign ram[415] = {rf_filter_coeff415_a, rf_filter_coeff415_b};
assign ram[416] = {rf_filter_coeff416_a, rf_filter_coeff416_b};
assign ram[417] = {rf_filter_coeff417_a, rf_filter_coeff417_b};
assign ram[418] = {rf_filter_coeff418_a, rf_filter_coeff418_b};
assign ram[419] = {rf_filter_coeff419_a, rf_filter_coeff419_b};
assign ram[420] = {rf_filter_coeff420_a, rf_filter_coeff420_b};
assign ram[421] = {rf_filter_coeff421_a, rf_filter_coeff421_b};
assign ram[422] = {rf_filter_coeff422_a, rf_filter_coeff422_b};
assign ram[423] = {rf_filter_coeff423_a, rf_filter_coeff423_b};
assign ram[424] = {rf_filter_coeff424_a, rf_filter_coeff424_b};
assign ram[425] = {rf_filter_coeff425_a, rf_filter_coeff425_b};
assign ram[426] = {rf_filter_coeff426_a, rf_filter_coeff426_b};
assign ram[427] = {rf_filter_coeff427_a, rf_filter_coeff427_b};
assign ram[428] = {rf_filter_coeff428_a, rf_filter_coeff428_b};
assign ram[429] = {rf_filter_coeff429_a, rf_filter_coeff429_b};
assign ram[430] = {rf_filter_coeff430_a, rf_filter_coeff430_b};
assign ram[431] = {rf_filter_coeff431_a, rf_filter_coeff431_b};
assign ram[432] = {rf_filter_coeff432_a, rf_filter_coeff432_b};
assign ram[433] = {rf_filter_coeff433_a, rf_filter_coeff433_b};
assign ram[434] = {rf_filter_coeff434_a, rf_filter_coeff434_b};
assign ram[435] = {rf_filter_coeff435_a, rf_filter_coeff435_b};
assign ram[436] = {rf_filter_coeff436_a, rf_filter_coeff436_b};
assign ram[437] = {rf_filter_coeff437_a, rf_filter_coeff437_b};
assign ram[438] = {rf_filter_coeff438_a, rf_filter_coeff438_b};
assign ram[439] = {rf_filter_coeff439_a, rf_filter_coeff439_b};
assign ram[440] = {rf_filter_coeff440_a, rf_filter_coeff440_b};
assign ram[441] = {rf_filter_coeff441_a, rf_filter_coeff441_b};
assign ram[442] = {rf_filter_coeff442_a, rf_filter_coeff442_b};
assign ram[443] = {rf_filter_coeff443_a, rf_filter_coeff443_b};
assign ram[444] = {rf_filter_coeff444_a, rf_filter_coeff444_b};
assign ram[445] = {rf_filter_coeff445_a, rf_filter_coeff445_b};
assign ram[446] = {rf_filter_coeff446_a, rf_filter_coeff446_b};
assign ram[447] = {rf_filter_coeff447_a, rf_filter_coeff447_b};
assign ram[448] = {rf_filter_coeff448_a, rf_filter_coeff448_b};
assign ram[449] = {rf_filter_coeff449_a, rf_filter_coeff449_b};
assign ram[450] = {rf_filter_coeff450_a, rf_filter_coeff450_b};
assign ram[451] = {rf_filter_coeff451_a, rf_filter_coeff451_b};
assign ram[452] = {rf_filter_coeff452_a, rf_filter_coeff452_b};
assign ram[453] = {rf_filter_coeff453_a, rf_filter_coeff453_b};
assign ram[454] = {rf_filter_coeff454_a, rf_filter_coeff454_b};
assign ram[455] = {rf_filter_coeff455_a, rf_filter_coeff455_b};
assign ram[456] = {rf_filter_coeff456_a, rf_filter_coeff456_b};
assign ram[457] = {rf_filter_coeff457_a, rf_filter_coeff457_b};
assign ram[458] = {rf_filter_coeff458_a, rf_filter_coeff458_b};
assign ram[459] = {rf_filter_coeff459_a, rf_filter_coeff459_b};
```

```
assign ram[460] = {rf_filter_coeff460_a, rf_filter_coeff460_b};
assign ram[461] = {rf_filter_coeff461_a, rf_filter_coeff461_b};
assign ram[462] = {rf_filter_coeff462_a, rf_filter_coeff462_b};
assign ram[463] = {rf_filter_coeff463_a, rf_filter_coeff463_b};
assign ram[464] = {rf_filter_coeff464_a, rf_filter_coeff464_b};
assign ram[465] = {rf_filter_coeff465_a, rf_filter_coeff465_b};
assign ram[466] = {rf_filter_coeff466_a, rf_filter_coeff466_b};
assign ram[467] = {rf_filter_coeff467_a, rf_filter_coeff467_b};
assign ram[468] = {rf_filter_coeff468_a, rf_filter_coeff468_b};
assign ram[469] = {rf_filter_coeff469_a, rf_filter_coeff469_b};
assign ram[470] = {rf_filter_coeff470_a, rf_filter_coeff470_b};
assign ram[471] = {rf_filter_coeff471_a, rf_filter_coeff471_b};
assign ram[472] = {rf_filter_coeff472_a, rf_filter_coeff472_b};
assign ram[473] = {rf_filter_coeff473_a, rf_filter_coeff473_b};
assign ram[474] = {rf_filter_coeff474_a, rf_filter_coeff474_b};
assign ram[475] = {rf_filter_coeff475_a, rf_filter_coeff475_b};
assign ram[476] = {rf_filter_coeff476_a, rf_filter_coeff476_b};
assign ram[477] = {rf_filter_coeff477_a, rf_filter_coeff477_b};
assign ram[478] = {rf_filter_coeff478_a, rf_filter_coeff478_b};
assign ram[479] = {rf_filter_coeff479_a, rf_filter_coeff479_b};
assign ram[480] = {rf_filter_coeff480_a, rf_filter_coeff480_b};
assign ram[481] = {rf_filter_coeff481_a, rf_filter_coeff481_b};
assign ram[482] = {rf_filter_coeff482_a, rf_filter_coeff482_b};
assign ram[483] = {rf_filter_coeff483_a, rf_filter_coeff483_b};
assign ram[484] = {rf_filter_coeff484_a, rf_filter_coeff484_b};
assign ram[485] = {rf_filter_coeff485_a, rf_filter_coeff485_b};
assign ram[486] = {rf_filter_coeff486_a, rf_filter_coeff486_b};
assign ram[487] = {rf_filter_coeff487_a, rf_filter_coeff487_b};
assign ram[488] = {rf_filter_coeff488_a, rf_filter_coeff488_b};
assign ram[489] = {rf_filter_coeff489_a, rf_filter_coeff489_b};
assign ram[490] = {rf_filter_coeff490_a, rf_filter_coeff490_b};
assign ram[491] = {rf_filter_coeff491_a, rf_filter_coeff491_b};
assign ram[492] = {rf_filter_coeff492_a, rf_filter_coeff492_b};
assign ram[493] = {rf_filter_coeff493_a, rf_filter_coeff493_b};
assign ram[494] = {rf_filter_coeff494_a, rf_filter_coeff494_b};
assign ram[495] = {rf_filter_coeff495_a, rf_filter_coeff495_b};
assign ram[496] = {rf_filter_coeff496_a, rf_filter_coeff496_b};
assign ram[497] = {rf_filter_coeff497_a, rf_filter_coeff497_b};
assign ram[498] = {rf_filter_coeff498_a, rf_filter_coeff498_b};
assign ram[499] = {rf_filter_coeff499_a, rf_filter_coeff499_b};
assign ram[500] = {rf_filter_coeff500_a, rf_filter_coeff500_b};
assign ram[501] = {rf_filter_coeff501_a, rf_filter_coeff501_b};
assign ram[502] = {rf_filter_coeff502_a, rf_filter_coeff502_b};
assign ram[503] = {rf_filter_coeff503_a, rf_filter_coeff503_b};
assign ram[504] = {rf_filter_coeff504_a, rf_filter_coeff504_b};
assign ram[505] = {rf_filter_coeff505_a, rf_filter_coeff505_b};
assign ram[506] = {rf_filter_coeff506_a, rf_filter_coeff506_b};
assign ram[507] = {rf_filter_coeff507_a, rf_filter_coeff507_b};
assign ram[508] = {rf_filter_coeff508_a, rf_filter_coeff508_b};
assign ram[509] = {rf_filter_coeff509_a, rf_filter_coeff509_b};
assign ram[510] = {rf_filter_coeff510_a, rf_filter_coeff510_b};
assign ram[511] = {rf_filter_coeff511_a, rf_filter_coeff511_b};


always @(posedge clk)
        begin
                        if(rden)
                                rddata <= ram[rdptr];
        end

endmodule
```

filter_stm.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_stm.v
// Create Date:             10/15/2016
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps


module filter_stm( clk, rst_n,
                           filter_aud_in_rts, filter_aud_in_rtr, filter_aud_out_rts,
filter_aud_out_rtr,
                           filter_aud_in, accumulator_load, accumulator_enable,
accumulator_in_left, accumulator_in_right,
                           rf_filter_coeff, mux_re, mux_rdptr
    );

    input
clk;                                                                        //Clock for State
Machine
    input                                  rst_n;
                                                                     //Active -low reset signal

    input           [31:0]      filter_aud_in;
    input                        [15:0]      rf_filter_coeff;
    input                                    filter_aud_in_rts;
//Ready to Send
    output                                   filter_aud_in_rtr;
//Ready to Recieve
    output                                   filter_aud_out_rts;
    input                                filter_aud_out_rtr;

    output                                   mux_re;
    output                       [8:0]       mux_rdptr;

    output                                   accumulator_load;
    output                                   accumulator_enable;
    output                       [31:0]      accumulator_in_left;
    output                       [31:0]      accumulator_in_right;

//*****************************************************************************
    localparam                           IDLE                                = 4'b0001,
                                 TRANSFER                            = 4'b0010,
                                 MULTIPLY_1ST        = 4'b0100,
                                 MULTIPLY                            = 4'b1000;
//*****************************************************************************
    localparam                  IDLE_ID                        = 0,
                                 TRANSFER_ID                    = 1,
                                 MULTIPLY_1ST_ID     = 2,
                                 MULTIPLY_ID                    = 3;
//*****************************************************************************
    localparam                           PTR                                     = 9,
                                 WIDTH                               = 16,
                                 TAPS                                = 512;
//*****************************************************************************
    reg              [3:0]       filter_state, filter_state_nxt;                    //Current
State | State for next Clock Cycle
    reg                                                  filter_running_1st,
filter_running_1st_nxt;
    reg                                                  filter_running, filter_running_nxt;
//*****************************************************************************
    reg                                                  accumulator_enable,
accumulator_enable_nxt;
    reg                                                  accumulator_load, accumulator_load_nxt;
//*****************************************************************************
    reg                                                  filter_init, filter_init_nxt;
```

```
    reg                                                 filter_need_new, filter_need_new_nxt;
    reg               [PTR-1:0]        filter_count, filter_count_nxt;
    reg                                                 filter_aud_out_rts,
filter_aud_out_rts_nxt;
    reg                                                 filter_aud_in_rtr, filter_aud_in_rtr_nxt;
//*******************************************************************************
    reg               [PTR-1:0]        wr_addr_x, wr_addr_x_nxt;
    reg               [PTR-1:0]        rd_addr_x, rd_addr_x_nxt;
    reg                                                 arr_re_x, arr_re_x_nxt;
    reg                                                 arr_we_x, arr_we_x_nxt;
//*******************************************************************************
    reg               [PTR-1:0]        mux_rdptr, mux_rdptr_nxt;
    reg                                                 mux_re, mux_re_nxt;
//*******************************************************************************
    wire                                                                  filter_xfc_in;
    wire signed     [15:0]                    h_unit;
    wire signed     [31:0]                    x_unit;
    wire signed     [15:0]                    x_unit_left; // Left is 31:16
    wire signed     [15:0]                    x_unit_right;// Right is 15:0
//*******************************************************************************
    wire signed     [31:0]                    accumulator_in_left;
    wire signed     [31:0]                    accumulator_in_right;
//*******************************************************************************
    assign h_unit                                                 = rf_filter_coeff;
    assign x_unit_left                                      = x_unit[31:16];
    assign x_unit_right                                     = x_unit[15: 0];
    assign filter_xfc_in                          = filter_aud_in_rtr && filter_aud_in_rts;
    assign accumulator_in_left                = x_unit_left * h_unit;
    assign accumulator_in_right           = x_unit_right * h_unit;

//*******************************************************************************

    always@(*)
          begin

                   filter_state_nxt                      = filter_state;
//*******************************************************************************
                   wr_addr_x_nxt                                  = wr_addr_x;
                   rd_addr_x_nxt                                  = rd_addr_x;
                   arr_re_x_nxt                                   = arr_re_x;
                   arr_we_x_nxt                                   = arr_we_x;
//*******************************************************************************
                   mux_rdptr_nxt                                  = mux_rdptr;
                   mux_re_nxt                                              = mux_re;
//*******************************************************************************

                   filter_running_1st_nxt       = filter_running_1st;
                   filter_running_nxt           = filter_running;
                   filter_need_new_nxt          = filter_need_new;
                   filter_count_nxt                      = filter_count;
//*******************************************************************************
                   filter_aud_in_rtr_nxt        = filter_aud_in_rtr;
                   filter_aud_out_rts_nxt  = filter_aud_out_rts;
                   accumulator_enable_nxt  = accumulator_enable;
                   accumulator_load_nxt         = accumulator_load;
                   case(1'b1)
//*******************************************************************************
// IDLE STATE
//*******************************************************************************
                   filter_state[IDLE_ID]:begin
                             if(filter_xfc_in)
            begin
                filter_state_nxt       = TRANSFER;
                arr_we_x_nxt           = 1'b1;
            end
                             else
            begin
                filter_aud_in_rtr_nxt = 1'b1;
            end
                   end
//*******************************************************************************
```

```
// TRANSFER STATE
//*******************************************************************************

                filter_state[TRANSFER_ID]:begin
                          if(filter_running_1st)
            begin
                filter_state_nxt                = MULTIPLY_1ST;
                filter_running_1st_nxt= 1'b0;
                filter_running_nxt              = 1'b1;
                arr_re_x_nxt                            = 1'b1;
                mux_re_nxt                              = 1'b1;
                arr_we_x_nxt                            = 1'b0;
                wr_addr_x_nxt                       = wr_addr_x + 1'b1;
            end
                          else
            begin
                filter_aud_out_rts_nxt                  = 1'b0;
                if(filter_xfc_in)
                begin
                    filter_aud_in_rtr_nxt       = 1'b0;
                    arr_we_x_nxt                        = 1'b1;
                    filter_running_1st_nxt      = 1'b1;

                end
            end
                end


//*******************************************************************************
// MULTIPLY STATE 1ST
//*******************************************************************************

                filter_state[MULTIPLY_1ST_ID]:begin

                          if(filter_running)
            begin
                filter_state_nxt                = MULTIPLY;
                filter_running_nxt              = 1'b0;
                accumulator_load_nxt   = 1'b1;
                accumulator_enable_nxt          = 1'b1;
                rd_addr_x_nxt                           = rd_addr_x - 1'b1;
                mux_rdptr_nxt                           = mux_rdptr + 1'b1;
                filter_count_nxt                = filter_count + 1'b1;
            end
                          else
            begin

            end
                end

//*******************************************************************************
// MULTIPLY STATE
//*******************************************************************************

                filter_state[MULTIPLY_ID]:begin
                          if(filter_need_new)
            begin
                accumulator_enable_nxt          = 1'b0;
                if (filter_aud_out_rtr)
                begin
                    filter_state_nxt            = TRANSFER;
                    filter_need_new_nxt                     = 1'b0;
                    filter_aud_in_rtr_nxt       = 1'b1;
                end
            end
                          else
            begin
                rd_addr_x_nxt                           = rd_addr_x - 1'b1;
                mux_rdptr_nxt                           = mux_rdptr + 1'b1;
                filter_count_nxt                = filter_count + 1'b1;
                accumulator_load_nxt   = 1'b0;
```

```
                if (filter_count == TAPS-1)
                begin
                    rd_addr_x_nxt = rd_addr_x;
                    filter_need_new_nxt     = 1'b1;
                    arr_re_x_nxt                    = 1'b0;
                    mux_re_nxt                          = 1'b0;
                    filter_aud_out_rts_nxt  = 1'b1;
                end
            end
                   end

                   default: begin end
                   endcase
          end
//*****************************************************************************
    always@(posedge clk or negedge rst_n)
          begin
        if(!rst_n)
        begin
            filter_state                                    <= IDLE;
            wr_addr_x                                               <= 1'b0;
            rd_addr_x                                               <= 1'b0;
            arr_re_x                                                      <= 1'b0;
            arr_we_x                                                      <= 1'b0;
            mux_rdptr                                               <= 1'b0;
            mux_re                                                  <= 1'b0;
            filter_running                                  <= 1'b0;
            filter_running_1st                          <= 1'b0;
            filter_need_new                             <= 1'b0;
            filter_count                                    <= 1'b0;
            filter_aud_in_rtr                               <= 1'b0;
            filter_aud_out_rts                <= 1'b0;
            accumulator_enable                          <= 1'b0;
            accumulator_load                        <= 1'b0;

        end
        else
        begin
            filter_state                                    <= filter_state_nxt;
            wr_addr_x                                               <= wr_addr_x_nxt;
            rd_addr_x                                               <= rd_addr_x_nxt;
            arr_re_x                                                      <= arr_re_x_nxt;
            arr_we_x                                                      <= arr_we_x_nxt;
            mux_rdptr                                               <= mux_rdptr_nxt;
            mux_re                                                  <= mux_re_nxt;

            filter_running                              <= filter_running_nxt;
            filter_running_1st                          <= filter_running_1st_nxt;
            filter_need_new                             <= filter_need_new_nxt;
            filter_count                                    <= filter_count_nxt;
            filter_aud_in_rtr                               <= filter_aud_in_rtr_nxt;
            filter_aud_out_rts                          <= filter_aud_out_rts_nxt;
            accumulator_enable                          <= accumulator_enable_nxt;
            accumulator_load                        <= accumulator_load_nxt;
        end
          end


          filter_storage     filter_storage_x(
                    .clk               (clk),
                                                .rst_n              (rst_n),
                                                .wren               (arr_we_x),
                                                .wrptr        (wr_addr_x),
                                                .wrdata       (filter_aud_in),
                                                .rden               (arr_re_x),
                                                .rdptr        (rd_addr_x),
                                                .rddata       (x_unit)
    );

endmodule
```

filter_storage.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_storage.v
// Create Date:             10/15/2016
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module filter_storage( clk, rst_n, wren, wrptr, wrdata, rden, rdptr, rddata
    );

    input                                                                   clk;
    input                                                                   rst_n;
    input                                                                   wren;
    input              [8:0]                                        wrptr;
    input              [31:0]                                           wrdata;
    input                                                                   rden;
    input              [8:0]                                        rdptr;
    output                 [31:0]                                       rddata;

    localparam                                 DEPTH = 511; //2^9 <= 16'b0; 512
    localparam                                 WIDTH = 31;

    reg               [WIDTH:0]               ram [DEPTH:0];
    reg               [WIDTH:0]               rddata;


    always @(posedge clk or negedge rst_n)
    begin
        if (!rst_n)
        begin
             ram[0] <= 16'b0;
             ram[1] <= 16'b0;
             ram[2] <= 16'b0;
             ram[3] <= 16'b0;
             ram[4] <= 16'b0;
             ram[5] <= 16'b0;
             ram[6] <= 16'b0;
             ram[7] <= 16'b0;
             ram[8] <= 16'b0;
             ram[9] <= 16'b0;
             ram[10] <= 16'b0;
             ram[11] <= 16'b0;
             ram[12] <= 16'b0;
             ram[13] <= 16'b0;
             ram[14] <= 16'b0;
             ram[15] <= 16'b0;
             ram[16] <= 16'b0;
             ram[17] <= 16'b0;
             ram[18] <= 16'b0;
             ram[19] <= 16'b0;
             ram[20] <= 16'b0;
             ram[21] <= 16'b0;
             ram[22] <= 16'b0;
             ram[23] <= 16'b0;
             ram[24] <= 16'b0;
             ram[25] <= 16'b0;
             ram[26] <= 16'b0;
             ram[27] <= 16'b0;
             ram[28] <= 16'b0;
             ram[29] <= 16'b0;
             ram[30] <= 16'b0;
             ram[31] <= 16'b0;
             ram[32] <= 16'b0;
             ram[33] <= 16'b0;
             ram[34] <= 16'b0;
```

```
ram[35] <= 16'b0;
ram[36] <= 16'b0;
ram[37] <= 16'b0;
ram[38] <= 16'b0;
ram[39] <= 16'b0;
ram[40] <= 16'b0;
ram[41] <= 16'b0;
ram[42] <= 16'b0;
ram[43] <= 16'b0;
ram[44] <= 16'b0;
ram[45] <= 16'b0;
ram[46] <= 16'b0;
ram[47] <= 16'b0;
ram[48] <= 16'b0;
ram[49] <= 16'b0;
ram[50] <= 16'b0;
ram[51] <= 16'b0;
ram[52] <= 16'b0;
ram[53] <= 16'b0;
ram[54] <= 16'b0;
ram[55] <= 16'b0;
ram[56] <= 16'b0;
ram[57] <= 16'b0;
ram[58] <= 16'b0;
ram[59] <= 16'b0;
ram[60] <= 16'b0;
ram[61] <= 16'b0;
ram[62] <= 16'b0;
ram[63] <= 16'b0;
ram[64] <= 16'b0;
ram[65] <= 16'b0;
ram[66] <= 16'b0;
ram[67] <= 16'b0;
ram[68] <= 16'b0;
ram[69] <= 16'b0;
ram[70] <= 16'b0;
ram[71] <= 16'b0;
ram[72] <= 16'b0;
ram[73] <= 16'b0;
ram[74] <= 16'b0;
ram[75] <= 16'b0;
ram[76] <= 16'b0;
ram[77] <= 16'b0;
ram[78] <= 16'b0;
ram[79] <= 16'b0;
ram[80] <= 16'b0;
ram[81] <= 16'b0;
ram[82] <= 16'b0;
ram[83] <= 16'b0;
ram[84] <= 16'b0;
ram[85] <= 16'b0;
ram[86] <= 16'b0;
ram[87] <= 16'b0;
ram[88] <= 16'b0;
ram[89] <= 16'b0;
ram[90] <= 16'b0;
ram[91] <= 16'b0;
ram[92] <= 16'b0;
ram[93] <= 16'b0;
ram[94] <= 16'b0;
ram[95] <= 16'b0;
ram[96] <= 16'b0;
ram[97] <= 16'b0;
ram[98] <= 16'b0;
ram[99] <= 16'b0;
ram[100] <= 16'b0;
ram[101] <= 16'b0;
ram[102] <= 16'b0;
ram[103] <= 16'b0;
ram[104] <= 16'b0;
ram[105] <= 16'b0;
```

```
ram[106] <= 16'b0;
ram[107] <= 16'b0;
ram[108] <= 16'b0;
ram[109] <= 16'b0;
ram[110] <= 16'b0;
ram[111] <= 16'b0;
ram[112] <= 16'b0;
ram[113] <= 16'b0;
ram[114] <= 16'b0;
ram[115] <= 16'b0;
ram[116] <= 16'b0;
ram[117] <= 16'b0;
ram[118] <= 16'b0;
ram[119] <= 16'b0;
ram[120] <= 16'b0;
ram[121] <= 16'b0;
ram[122] <= 16'b0;
ram[123] <= 16'b0;
ram[124] <= 16'b0;
ram[125] <= 16'b0;
ram[126] <= 16'b0;
ram[127] <= 16'b0;
ram[128] <= 16'b0;
ram[129] <= 16'b0;
ram[130] <= 16'b0;
ram[131] <= 16'b0;
ram[132] <= 16'b0;
ram[133] <= 16'b0;
ram[134] <= 16'b0;
ram[135] <= 16'b0;
ram[136] <= 16'b0;
ram[137] <= 16'b0;
ram[138] <= 16'b0;
ram[139] <= 16'b0;
ram[140] <= 16'b0;
ram[141] <= 16'b0;
ram[142] <= 16'b0;
ram[143] <= 16'b0;
ram[144] <= 16'b0;
ram[145] <= 16'b0;
ram[146] <= 16'b0;
ram[147] <= 16'b0;
ram[148] <= 16'b0;
ram[149] <= 16'b0;
ram[150] <= 16'b0;
ram[151] <= 16'b0;
ram[152] <= 16'b0;
ram[153] <= 16'b0;
ram[154] <= 16'b0;
ram[155] <= 16'b0;
ram[156] <= 16'b0;
ram[157] <= 16'b0;
ram[158] <= 16'b0;
ram[159] <= 16'b0;
ram[160] <= 16'b0;
ram[161] <= 16'b0;
ram[162] <= 16'b0;
ram[163] <= 16'b0;
ram[164] <= 16'b0;
ram[165] <= 16'b0;
ram[166] <= 16'b0;
ram[167] <= 16'b0;
ram[168] <= 16'b0;
ram[169] <= 16'b0;
ram[170] <= 16'b0;
ram[171] <= 16'b0;
ram[172] <= 16'b0;
ram[173] <= 16'b0;
ram[174] <= 16'b0;
ram[175] <= 16'b0;
ram[176] <= 16'b0;
```

```
ram[177] <= 16'b0;
ram[178] <= 16'b0;
ram[179] <= 16'b0;
ram[180] <= 16'b0;
ram[181] <= 16'b0;
ram[182] <= 16'b0;
ram[183] <= 16'b0;
ram[184] <= 16'b0;
ram[185] <= 16'b0;
ram[186] <= 16'b0;
ram[187] <= 16'b0;
ram[188] <= 16'b0;
ram[189] <= 16'b0;
ram[190] <= 16'b0;
ram[191] <= 16'b0;
ram[192] <= 16'b0;
ram[193] <= 16'b0;
ram[194] <= 16'b0;
ram[195] <= 16'b0;
ram[196] <= 16'b0;
ram[197] <= 16'b0;
ram[198] <= 16'b0;
ram[199] <= 16'b0;
ram[200] <= 16'b0;
ram[201] <= 16'b0;
ram[202] <= 16'b0;
ram[203] <= 16'b0;
ram[204] <= 16'b0;
ram[205] <= 16'b0;
ram[206] <= 16'b0;
ram[207] <= 16'b0;
ram[208] <= 16'b0;
ram[209] <= 16'b0;
ram[210] <= 16'b0;
ram[211] <= 16'b0;
ram[212] <= 16'b0;
ram[213] <= 16'b0;
ram[214] <= 16'b0;
ram[215] <= 16'b0;
ram[216] <= 16'b0;
ram[217] <= 16'b0;
ram[218] <= 16'b0;
ram[219] <= 16'b0;
ram[220] <= 16'b0;
ram[221] <= 16'b0;
ram[222] <= 16'b0;
ram[223] <= 16'b0;
ram[224] <= 16'b0;
ram[225] <= 16'b0;
ram[226] <= 16'b0;
ram[227] <= 16'b0;
ram[228] <= 16'b0;
ram[229] <= 16'b0;
ram[230] <= 16'b0;
ram[231] <= 16'b0;
ram[232] <= 16'b0;
ram[233] <= 16'b0;
ram[234] <= 16'b0;
ram[235] <= 16'b0;
ram[236] <= 16'b0;
ram[237] <= 16'b0;
ram[238] <= 16'b0;
ram[239] <= 16'b0;
ram[240] <= 16'b0;
ram[241] <= 16'b0;
ram[242] <= 16'b0;
ram[243] <= 16'b0;
ram[244] <= 16'b0;
ram[245] <= 16'b0;
ram[246] <= 16'b0;
ram[247] <= 16'b0;
```

```
ram[248] <= 16'b0;
ram[249] <= 16'b0;
ram[250] <= 16'b0;
ram[251] <= 16'b0;
ram[252] <= 16'b0;
ram[253] <= 16'b0;
ram[254] <= 16'b0;
ram[255] <= 16'b0;
ram[256] <= 16'b0;
ram[257] <= 16'b0;
ram[258] <= 16'b0;
ram[259] <= 16'b0;
ram[260] <= 16'b0;
ram[261] <= 16'b0;
ram[262] <= 16'b0;
ram[263] <= 16'b0;
ram[264] <= 16'b0;
ram[265] <= 16'b0;
ram[266] <= 16'b0;
ram[267] <= 16'b0;
ram[268] <= 16'b0;
ram[269] <= 16'b0;
ram[270] <= 16'b0;
ram[271] <= 16'b0;
ram[272] <= 16'b0;
ram[273] <= 16'b0;
ram[274] <= 16'b0;
ram[275] <= 16'b0;
ram[276] <= 16'b0;
ram[277] <= 16'b0;
ram[278] <= 16'b0;
ram[279] <= 16'b0;
ram[280] <= 16'b0;
ram[281] <= 16'b0;
ram[282] <= 16'b0;
ram[283] <= 16'b0;
ram[284] <= 16'b0;
ram[285] <= 16'b0;
ram[286] <= 16'b0;
ram[287] <= 16'b0;
ram[288] <= 16'b0;
ram[289] <= 16'b0;
ram[290] <= 16'b0;
ram[291] <= 16'b0;
ram[292] <= 16'b0;
ram[293] <= 16'b0;
ram[294] <= 16'b0;
ram[295] <= 16'b0;
ram[296] <= 16'b0;
ram[297] <= 16'b0;
ram[298] <= 16'b0;
ram[299] <= 16'b0;
ram[300] <= 16'b0;
ram[301] <= 16'b0;
ram[302] <= 16'b0;
ram[303] <= 16'b0;
ram[304] <= 16'b0;
ram[305] <= 16'b0;
ram[306] <= 16'b0;
ram[307] <= 16'b0;
ram[308] <= 16'b0;
ram[309] <= 16'b0;
ram[310] <= 16'b0;
ram[311] <= 16'b0;
ram[312] <= 16'b0;
ram[313] <= 16'b0;
ram[314] <= 16'b0;
ram[315] <= 16'b0;
ram[316] <= 16'b0;
ram[317] <= 16'b0;
ram[318] <= 16'b0;
```

```
ram[319] <= 16'b0;
ram[320] <= 16'b0;
ram[321] <= 16'b0;
ram[322] <= 16'b0;
ram[323] <= 16'b0;
ram[324] <= 16'b0;
ram[325] <= 16'b0;
ram[326] <= 16'b0;
ram[327] <= 16'b0;
ram[328] <= 16'b0;
ram[329] <= 16'b0;
ram[330] <= 16'b0;
ram[331] <= 16'b0;
ram[332] <= 16'b0;
ram[333] <= 16'b0;
ram[334] <= 16'b0;
ram[335] <= 16'b0;
ram[336] <= 16'b0;
ram[337] <= 16'b0;
ram[338] <= 16'b0;
ram[339] <= 16'b0;
ram[340] <= 16'b0;
ram[341] <= 16'b0;
ram[342] <= 16'b0;
ram[343] <= 16'b0;
ram[344] <= 16'b0;
ram[345] <= 16'b0;
ram[346] <= 16'b0;
ram[347] <= 16'b0;
ram[348] <= 16'b0;
ram[349] <= 16'b0;
ram[350] <= 16'b0;
ram[351] <= 16'b0;
ram[352] <= 16'b0;
ram[353] <= 16'b0;
ram[354] <= 16'b0;
ram[355] <= 16'b0;
ram[356] <= 16'b0;
ram[357] <= 16'b0;
ram[358] <= 16'b0;
ram[359] <= 16'b0;
ram[360] <= 16'b0;
ram[361] <= 16'b0;
ram[362] <= 16'b0;
ram[363] <= 16'b0;
ram[364] <= 16'b0;
ram[365] <= 16'b0;
ram[366] <= 16'b0;
ram[367] <= 16'b0;
ram[368] <= 16'b0;
ram[369] <= 16'b0;
ram[370] <= 16'b0;
ram[371] <= 16'b0;
ram[372] <= 16'b0;
ram[373] <= 16'b0;
ram[374] <= 16'b0;
ram[375] <= 16'b0;
ram[376] <= 16'b0;
ram[377] <= 16'b0;
ram[378] <= 16'b0;
ram[379] <= 16'b0;
ram[380] <= 16'b0;
ram[381] <= 16'b0;
ram[382] <= 16'b0;
ram[383] <= 16'b0;
ram[384] <= 16'b0;
ram[385] <= 16'b0;
ram[386] <= 16'b0;
ram[387] <= 16'b0;
ram[388] <= 16'b0;
ram[389] <= 16'b0;
```

```
ram[390] <= 16'b0;
ram[391] <= 16'b0;
ram[392] <= 16'b0;
ram[393] <= 16'b0;
ram[394] <= 16'b0;
ram[395] <= 16'b0;
ram[396] <= 16'b0;
ram[397] <= 16'b0;
ram[398] <= 16'b0;
ram[399] <= 16'b0;
ram[400] <= 16'b0;
ram[401] <= 16'b0;
ram[402] <= 16'b0;
ram[403] <= 16'b0;
ram[404] <= 16'b0;
ram[405] <= 16'b0;
ram[406] <= 16'b0;
ram[407] <= 16'b0;
ram[408] <= 16'b0;
ram[409] <= 16'b0;
ram[410] <= 16'b0;
ram[411] <= 16'b0;
ram[412] <= 16'b0;
ram[413] <= 16'b0;
ram[414] <= 16'b0;
ram[415] <= 16'b0;
ram[416] <= 16'b0;
ram[417] <= 16'b0;
ram[418] <= 16'b0;
ram[419] <= 16'b0;
ram[420] <= 16'b0;
ram[421] <= 16'b0;
ram[422] <= 16'b0;
ram[423] <= 16'b0;
ram[424] <= 16'b0;
ram[425] <= 16'b0;
ram[426] <= 16'b0;
ram[427] <= 16'b0;
ram[428] <= 16'b0;
ram[429] <= 16'b0;
ram[430] <= 16'b0;
ram[431] <= 16'b0;
ram[432] <= 16'b0;
ram[433] <= 16'b0;
ram[434] <= 16'b0;
ram[435] <= 16'b0;
ram[436] <= 16'b0;
ram[437] <= 16'b0;
ram[438] <= 16'b0;
ram[439] <= 16'b0;
ram[440] <= 16'b0;
ram[441] <= 16'b0;
ram[442] <= 16'b0;
ram[443] <= 16'b0;
ram[444] <= 16'b0;
ram[445] <= 16'b0;
ram[446] <= 16'b0;
ram[447] <= 16'b0;
ram[448] <= 16'b0;
ram[449] <= 16'b0;
ram[450] <= 16'b0;
ram[451] <= 16'b0;
ram[452] <= 16'b0;
ram[453] <= 16'b0;
ram[454] <= 16'b0;
ram[455] <= 16'b0;
ram[456] <= 16'b0;
ram[457] <= 16'b0;
ram[458] <= 16'b0;
ram[459] <= 16'b0;
ram[460] <= 16'b0;
```

```
            ram[461] <= 16'b0;
            ram[462] <= 16'b0;
            ram[463] <= 16'b0;
            ram[464] <= 16'b0;
            ram[465] <= 16'b0;
            ram[466] <= 16'b0;
            ram[467] <= 16'b0;
            ram[468] <= 16'b0;
            ram[469] <= 16'b0;
            ram[470] <= 16'b0;
            ram[471] <= 16'b0;
            ram[472] <= 16'b0;
            ram[473] <= 16'b0;
            ram[474] <= 16'b0;
            ram[475] <= 16'b0;
            ram[476] <= 16'b0;
            ram[477] <= 16'b0;
            ram[478] <= 16'b0;
            ram[479] <= 16'b0;
            ram[480] <= 16'b0;
            ram[481] <= 16'b0;
            ram[482] <= 16'b0;
            ram[483] <= 16'b0;
            ram[484] <= 16'b0;
            ram[485] <= 16'b0;
            ram[486] <= 16'b0;
            ram[487] <= 16'b0;
            ram[488] <= 16'b0;
            ram[489] <= 16'b0;
            ram[490] <= 16'b0;
            ram[491] <= 16'b0;
            ram[492] <= 16'b0;
            ram[493] <= 16'b0;
            ram[494] <= 16'b0;
            ram[495] <= 16'b0;
            ram[496] <= 16'b0;
            ram[497] <= 16'b0;
            ram[498] <= 16'b0;
            ram[499] <= 16'b0;
            ram[500] <= 16'b0;
            ram[501] <= 16'b0;
            ram[502] <= 16'b0;
            ram[503] <= 16'b0;
            ram[504] <= 16'b0;
            ram[505] <= 16'b0;
            ram[506] <= 16'b0;
            ram[507] <= 16'b0;
            ram[508] <= 16'b0;
            ram[509] <= 16'b0;
            ram[510] <= 16'b0;
            ram[511] <= 16'b0;
        end

        else if(wren)
            ram[wrptr] <= wrdata;

    end

    always @(posedge clk)
    begin
        if(rden)
            rddata <= ram[rdptr];
    end

endmodule
```

filter_round_truncate.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_round_truncate.v
// Create Date:             10/15/2016
// Last Modification:       4/1/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

// ext_acc_in, acc_r should be 43 bits, not 47 bits
// Reasoning:
//      12-19 bits lopped off during shift
//      16 bits to keep
//      8 bits of integer overflow, used for clipping
//  MAX needed is 8 + 16 + 19 bits
//
//  After rounding, keep 24 bits
//  acc_t should be 24 bits, not 28 bits

module filter_round_truncate(clk, rst_n, acc_in, rf_sat, rf_shift, trig_filter_ovf_flag_clear,
filter_out, ro_filter_ovf_flag
    );

    // MODULE CONNECTIONS
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------
    input                                   clk;
    input                                   rst_n;
    input               [39:0]              acc_in;
    input                                   rf_sat;
    input               [2:0]               rf_shift;
    input                                   trig_filter_ovf_flag_clear;
    output reg          [15:0]              filter_out;
    output reg                              ro_filter_ovf_flag;
    reg signed          [42:0]              acc_r;
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------

    reg signed          [23:0]              acc_t;
    wire                [4:0]               num_shift;
    wire                                    sign_bit;
    wire                [42:0]              ext_acc_in;


    assign num_shift = rf_shift + 12;
    assign sign_bit = acc_in[39];
    assign ext_acc_in = {{3{sign_bit}}, acc_in};

    always@(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            filter_out <= 1'b0;
            ro_filter_ovf_flag <= 1'b0;
        end

        else
        begin
            acc_r <= ext_acc_in + (1<<(num_shift-1));
            acc_t <= acc_r[num_shift+:24];

            // when num_shift is 12 == acc_t[35:12]
            // when num_shift is 19 == acc_t[42:19]

            if (acc_t > (1<<15)-1)
            begin
                ro_filter_ovf_flag <= 1;
```

```
                if (rf_sat)
                                                        filter_out <= (1<<15)-1;
                else
                    filter_out <= acc_t[15:0];
            end

            else if (acc_t < -(1<<15))
            begin
                ro_filter_ovf_flag <= 1;
                if (rf_sat)
                                                    filter_out <= -(1<<15);
                else
                    filter_out <= acc_t[15:0];
            end

            else
                filter_out <= acc_t[15:0];

            if (trig_filter_ovf_flag_clear)
                ro_filter_ovf_flag <= 1'b0;
        end
    end

endmodule
```

i2c.v:

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2c.v
// Create Date:             10/15/2016
// Last Modification:       1/13/2015
// Author:                  Whitley Forman
// Description:
////////////////////////////////////////////////////////////////////////////////

module i2c(
    input [2:0] i2c_addr_bits,                  //descriptions needed
    input i2c_sda_in,                           //
    input i2c_scl,                              //
    output i2c_op,
    output i2c_xfc_write,
    output [10:0] i2c_wraddr,
    output [7:0] i2c_wdata,
    input [7:0] i2c_rdata,
    input i2c_xfc_read,
    output i2c_sda_out,
    input clk,
    input reset
    );

    wire [10:0] addr_wire;
    wire [7:0] data_wire;

    Deserializer i2c_top_deserializer (
        .Clock                                  (clk),              //external input   *
        .Reset                                  (reset),            //external input   *
        .i2c_sda_raw                (i2c_sda_in),                   //externa input    *
        .i2c_scl_raw                (i2c_scl),                      //external input   *
        .i2c_addr_bits          (i2c_addr_bits),                    //external input   *
        .i2c_RW                                 (i2c_RW),           //internal output  [1]*
        .i2c_addr                   (addr_wire),                    //internal output  [11]*
        .addr_xfc                   (addr_xfc),                     //internal output  [1]*
        .i2c_ack                    (i2c_ack),                      //internal output  [1]*
        .data_xfc                   (data_xfc),                     //internal output  [1]*
        .serial_data                (data_wire),                    //internal output  [8]*
        .stop_out                   (stop_out)                      //internal output  [1]*
        );

    Sequencer i2c_top_sequencer (
        .Clock                                  (clk),              //external input   *
        .i2c_RW                                 (i2c_RW),           //internal input   *
        .i2c_op                                 (i2c_op),           //external output  *
        .i2c_addr_in                (addr_wire),                    //internal input   *
        .i2c_addr_out               (i2c_wraddr),                   //external output  *
        .i2c_data_in                (data_wire),                    //internal input   *
        .i2c_data_out               (i2c_wdata),                    //external output  *
        .addr_xfc                   (addr_xfc),                     //internal input   *
        .data_xfc                   (data_xfc),                     //internal input   *
        .i2c_xfc                    (i2c_xfc_write),                //external output  *
        .reset                                  (reset),            //external input   *
        .stop_out                   (stop_out)                      //internal input   *
        );

    Serializer i2c_top_serializer (
        .i2c_scl                    (i2c_scl),                      //external input   *
        .i2c_sda_out                (i2c_sda_out),                  //external input   *
        .i2c_ack                    (i2c_ack),                      //internal input   *
        .Clock                                  (clk),              //external input   *
        .reset                                  (reset),            //external input   *
        .i2c_rdata                  (i2c_rdata),                    //external input   *
        .i2c_xfc_read               (i2c_xfc_read)                  //external input   *
        );

endmodule
```

i2c_deserializer.v:

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2c_deserializer.v
// Create Date:             10/15/2016
// Last Modification:       1/13/2015
// Author:                  Whitley Forman
// Description:
////////////////////////////////////////////////////////////////////////////////

module Deserializer(
    input Clock,
    input Reset,
    input i2c_sda_raw,
          input i2c_scl_raw,
    input [2:0] i2c_addr_bits,

    output reg i2c_RW,
    output reg [10:0] i2c_addr ,
          output reg addr_xfc,
          output reg i2c_ack,
          output reg data_xfc,
          output reg [7:0] serial_data,
          output reg stop_out
          );

    //Double and triple rank sda and scl
    reg scl_p1;
    reg sda_p1;
    reg sda_p2;
    reg i2c_scl;
    reg i2c_sda;

    always@(posedge Clock)
    begin
        scl_p1 <= i2c_scl_raw;
        i2c_scl <= scl_p1;

        sda_p1 <= i2c_sda_raw;
        sda_p2 <= sda_p1;
        i2c_sda <= sda_p2;
    end


    //Create SDA and SCL Pulse Signals, and states
    wire i2c_sda_pos_pulse;
    wire i2c_sda_neg_pulse;
    wire i2c_scl_pos_pulse;
    wire i2c_scl_neg_pulse;
    reg Q_sda;
    reg Q_scl;
    reg sda_state;
    reg scl_state;

    //Find incoming pulses and createlocal clocked pulses at edges
    always@(posedge Clock)
    begin
        Q_sda = !i2c_sda;
        Q_scl = !i2c_scl;
    end
    assign i2c_sda_neg_pulse = !Q_sda && !i2c_sda;
    assign i2c_sda_pos_pulse = Q_sda && i2c_sda;
    assign i2c_scl_neg_pulse = !Q_scl && !i2c_scl;
    assign i2c_scl_pos_pulse = Q_scl && i2c_scl;

    //Transform local clocked pulses to full waveform states
    always@(posedge Clock)
    begin
```

```
        if (i2c_sda_pos_pulse)
        begin
        sda_state <= 1;
        end
        else if (i2c_sda_neg_pulse)
        begin
        sda_state <= 0;
        end
        else if(stop)
        begin
        sda_state <= 1;
        end
    end

    always@(posedge Clock)
    begin
        if (i2c_scl_pos_pulse)
        begin
        scl_state <= 1;
        end
        else if (i2c_scl_neg_pulse)
        begin
        scl_state <= 0;
        end
        else if (stop)
        begin
        scl_state <= 1;
        end
    end

    //Deserializer State Update
    //added 9/29/2015 for control of deserializer state, slaveaddress-RW 00, burst_addr 01, data
10
    reg [1:0] deserial_state;

    always@(posedge Clock)
    begin
        //change to register address mode part 1
        if(slave_ack)
            begin
            deserial_state <= 2'b01;
            end

            //change to register address mode part 2
        if(addr_ack && (deserial_state == 2'b01))
            begin
            deserial_state <= 2'b10;
            end

        //change to data fetch mode for write
        if(addr_ack2 && (deserial_state == 2'b10) && i2c_RW)
            begin
            deserial_state <= 2'b11;
            end

        //change to salve address mode if read, wait for start conditions
        if(addr_ack2 && (deserial_state == 2'b10) && !i2c_RW)
            begin
            deserial_state <= 2'b00;
            end

        if(stop)
            begin
            deserial_state <= 2'b00;
            end
    end


    //Set address with offboard bits 1010XXX
    wire [6:0] slave_addr ;
```

```
        assign slave_addr = {4'b1010 , i2c_addr_bits};  //set slave_addr = 1010XXX with external
pins

    //Start and Stop conditions
    reg stop;

    always@(posedge Clock or negedge Reset) //Stop conditions erratic
    begin
        if (!Reset || slave_addr_stop || (i2c_sda_pos_pulse && scl_state))
        begin
        stop <= 1;
        stop_out <= 1;
        end

        else if (i2c_sda_neg_pulse && scl_state)
        begin
        stop <= 0;
        stop_out <= 0;
        end

        else if (~i2c_RW & ~addr_ack2 & i2c_ack & (deserial_state == 2'b00))  //stop condition
after read request
        begin
        stop <= 1;
        end

    end

    //Deserialize slave address
    reg [3:0] bit_counter_slave_addr  = 4'b0;
    reg [7:0] incoming_slave_addr  = 8'b0;
    reg got_slave_addr;


    always@(posedge Clock)
            begin
            if(stop == 1)
            begin
            got_slave_addr <= 0;
            incoming_slave_addr  <= 8'b0;
            bit_counter_slave_addr  <= 4'b0;
            end

            if ((deserial_state == 2'b00) && i2c_scl_pos_pulse && !stop &&
!addr_xfc)                              //when looking for slave address - do
            begin
            case(bit_counter_slave_addr)                          //when bit counter = do these,
set incoming address bit
            4'b0000 : begin
                        incoming_slave_addr [7] <= sda_state;
                        bit_counter_slave_addr <= 4'b0001;
                      end
            4'b0001 : begin
                        incoming_slave_addr [6] <= sda_state;
                        bit_counter_slave_addr <= 4'b0010;
                      end
            4'b0010 : begin
                        incoming_slave_addr [5] <= sda_state;
                        bit_counter_slave_addr <= 4'b0011;
                      end
            4'b0011 : begin
                        incoming_slave_addr [4] <= sda_state;
                        bit_counter_slave_addr <= 4'b0100;
                      end
            4'b0100 : begin
                        incoming_slave_addr [3] <= sda_state;
                        bit_counter_slave_addr <= 4'b0101;
                      end
            4'b0101 : begin
                        incoming_slave_addr [2] <= sda_state;
                        bit_counter_slave_addr <= 4'b0110;
```

```
                    end
            4'b0110 : begin
                        incoming_slave_addr [1] <= sda_state;
                        bit_counter_slave_addr <= 4'b0111;
                    end
            4'b0111 : begin
                        incoming_slave_addr [0] <= sda_state;
                        bit_counter_slave_addr <= 4'b1000;
                        got_slave_addr <= 1;
                    end
            4'b1000 : begin
                        bit_counter_slave_addr <= 4'b0000;
                        got_slave_addr <= 0;
                    end
            endcase
            end
        end

    //Check Address
    reg slave_addr_stop;
    reg slave_ack;

    always@(posedge Clock)
    begin

        if(stop)
        begin
        slave_ack <= 0;
        i2c_RW <= 0;
        slave_addr_stop <= 0;
        end

        else if(slave_ack & i2c_scl_neg_pulse)
        begin
        slave_ack <= 0;
        end

        //if slave address good, acknowledge
        else if ((deserial_state == 2'b00) & i2c_scl_neg_pulse & got_slave_addr &
(incoming_slave_addr [7:1] == slave_addr [6:0]) & !addr_xfc)
        begin
        slave_ack <= 1;
        i2c_RW <= !incoming_slave_addr [0]; //code written in reverse by accident, so invert RW
bit
        end

        //if incoming slave adrres bad, stop sequence
        else if (got_slave_addr & (incoming_slave_addr [7:1] !== slave_addr [6:0]))
        begin
        slave_addr_stop <= 1;
        end
    end

    //i2c_ack to serializer
    always@(posedge Clock)
    begin
        if(stop)
        begin
        i2c_ack <= 0;
        end

        else if(slave_ack | addr_ack | data_xfc | addr_ack2)
        begin
        i2c_ack <= 1;
        end

        else if (!slave_ack & !addr_ack & !data_xfc & !addr_ack2)
        begin
        i2c_ack <= 0;
        end
    end
```

```
//Deserialize burst address
reg [3:0] bit_counter_burst_addr;
reg got_addr;
reg got_addr2;

always@(posedge Clock)
begin

    if (stop == 1)
    begin
    bit_counter_burst_addr  <= 0;
    i2c_addr <= 0;
    got_addr <= 1'b0;
    got_addr2 <= 1'b0;
    end

    else if(addr_ack)
    begin
    got_addr <= 0;
    bit_counter_burst_addr <= 4'b0000;
    end

        else if(addr_ack2)
    begin
    got_addr2 <= 0;
    bit_counter_burst_addr <= 4'b0000;
    end

    //Get burst address part 1
    else if (!slave_ack && (deserial_state  == 2'b01) && i2c_scl_pos_pulse)
    begin
    case(bit_counter_burst_addr)
        4'b0000 :        begin
                        i2c_addr [10] <= sda_state;
                        bit_counter_burst_addr <= 4'b0001;
                        end
        4'b0001 :        begin
                        i2c_addr [9] <= sda_state;
                        bit_counter_burst_addr <= 4'b0010;
                        end
        4'b0010 :        begin
                        i2c_addr [8] <= sda_state;
                        bit_counter_burst_addr <= 4'b0011;
                        end
        4'b0011 :        begin
                        i2c_addr [7] <= sda_state;
                        bit_counter_burst_addr <= 4'b0100;
                        end
        4'b0100 :        begin
                        i2c_addr [6] <= sda_state;
                        bit_counter_burst_addr <= 4'b0101;
                        end
        4'b0101 :        begin
                        i2c_addr [5] <= sda_state;
                        bit_counter_burst_addr <= 4'b0110;
                        end
        4'b0110 :        begin
                        i2c_addr [4] <= sda_state;
                        bit_counter_burst_addr <= 4'b0111;
                        end
        4'b0111 :        begin
                        i2c_addr [3] <= sda_state;
                        bit_counter_burst_addr <= 4'b1000;
                        got_addr <= 1'b1;
                        end
    endcase
    end

    //Get burst address part 2
    else if (!slave_ack && (deserial_state  == 2'b10) && i2c_scl_pos_pulse)
```

```
        begin
        case(bit_counter_burst_addr)
            4'b0000 :        begin
                             i2c_addr [2] <= sda_state;
                             bit_counter_burst_addr <= 4'b0001;
                             end
            4'b0001 :        begin
                             i2c_addr [1] <= sda_state;
                             bit_counter_burst_addr <= 4'b0010;
                             end
            4'b0010 :        begin
                             i2c_addr [0] <= sda_state;
                             bit_counter_burst_addr <= 4'b0011;
                             end
            4'b0011 :        begin
                             //burst_start_addr [7] <= sda_state;
                             bit_counter_burst_addr <= 4'b0100;
                             end
            4'b0100 :        begin
                             //burst_start_addr [6] <= sda_state;
                             bit_counter_burst_addr <= 4'b0101;
                             end
            4'b0101 :        begin
                             //burst_start_addr [5] <= sda_state;
                             bit_counter_burst_addr <= 4'b0110;
                             end
            4'b0110 :        begin
                             //burst_start_addr [4] <= sda_state;
                             bit_counter_burst_addr <= 4'b0111;
                             end
            4'b0111 :        begin
                             //burst_start_addr [3] <= sda_state;
                             bit_counter_burst_addr <= 4'b1000;
                             got_addr2 <= 1'b1;
                             end
            //4'b1000 :      begin
            //                                    i2c_addr [2:0]  <= burst_start_addr
[2:0];                                          //set busrt start address part 2
            //                                    bit_counter_burst_addr <= 4'b0000;
            //                                    got_addr2 <= 1'b0;
            //                                    end
        endcase
        end
    end

    //Address ack
    reg addr_ack;
    reg addr_ack2;

    always@(posedge Clock)
    begin

        if(stop)
        begin
        addr_ack <= 0;
        addr_xfc <=0;
        addr_ack2 <=0;
        end

        else if(addr_ack & i2c_scl_neg_pulse)
        begin
        addr_ack <= 0;
        end

        else if (i2c_scl_neg_pulse & (got_addr == 1'b1))
        begin
        addr_ack <= 1;
        end

        else if(addr_ack2 & i2c_scl_neg_pulse)
        begin
```

```
        addr_ack2 <= 0;
        end

        else if (i2c_scl_neg_pulse & (got_addr2 == 1'b1))
        begin
        addr_ack2 <= 1;
        end

        else if((deserial_state == 2'b10) & addr_ack2)
        begin
        addr_xfc <= 1;
        end

        else if(~addr_ack2)
        begin
        addr_xfc <= 0;
        end


    end


    //Deserialize data

    //reg data_xfc = 1'b0;
    //             initial data_xfc = 0;
    reg [3:0] bit_counter_data;
            //initial bit_counter_data = 0;//initialize bit counter to 0
    reg got_data;

    always@(posedge Clock)
    begin
        if(stop)
        begin
        serial_data <= 0;
        bit_counter_data <= 0;
        //data_xfc <= 0;
        got_data <= 0;
        end

        else if(data_xfc)
        begin
        got_data <= 0;
        bit_counter_data <= 4'b0000;
        serial_data <= 8'b00000000;
        end

        else if (!addr_ack2 & !got_data & i2c_scl_pos_pulse && (deserial_state == 2'b11) &
(i2c_RW == 1))
        begin
        case (bit_counter_data)
            4'b0000 :        begin
                             serial_data [7] <= sda_state;
                             bit_counter_data <= 4'b0001;
                        end
            4'b0001 :        begin
                             serial_data [6] <= sda_state;
                             bit_counter_data <= 4'b0010;
                        end
            4'b0010 :        begin
                             serial_data [5] <= sda_state;
                             bit_counter_data <= 4'b0011;
                        end
            4'b0011 :        begin
                             serial_data [4] <= sda_state;
                             bit_counter_data <= 4'b0100;
                        end
            4'b0100 :        begin
                             serial_data [3] <= sda_state;
                             bit_counter_data <= 4'b0101;
                        end
```

```
            4'b0101 :        begin
                                serial_data [2] <= sda_state;
                                bit_counter_data <= 4'b0110;
                            end
            4'b0110 :        begin
                                serial_data [1] <= sda_state;
                                bit_counter_data <= 4'b0111;
                            end
            4'b0111 :        begin
                                serial_data [0] <= sda_state;
                                got_data <= 1'b1;
                            end
        endcase
        end
    end

    always@(posedge Clock)
    begin

        if(stop)
        begin
        data_xfc <= 0;
        end

        else if(data_xfc & i2c_scl_neg_pulse)
        begin
        data_xfc <= 0;
        end

        else if ((deserial_state == 2'b11) & i2c_scl_neg_pulse & got_data)
        begin
        data_xfc <= 1;
        end
    end

endmodule
```

i2c_sequencer.v:

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2c_sequencer.v
// Create Date:             10/15/2016
// Last Modification:       1/13/2015
// Author:                  Whitley Forman
// Description:
////////////////////////////////////////////////////////////////////////////////

module Sequencer(
          input Clock,
    input i2c_RW,
    output reg i2c_op,
    input [10:0] i2c_addr_in,
    output reg [10:0] i2c_addr_out,
    input [7:0] i2c_data_in,
    output reg [7:0] i2c_data_out,
    input addr_xfc,
    input data_xfc,
          output reg i2c_xfc,
          input reset,
          input stop_out
    );

    //Create address incremental value register
    reg [10:0] addr_increment;
        initial addr_increment = 0;
    reg xfc_ready;
        initial xfc_ready=0;

    //single cycle address acknowledge
    wire addr_ack_temp;
    wire data_ack_temp;
    reg Q_addr;
    reg Q_data;
    always@(posedge Clock)
    begin
        Q_addr <= !addr_xfc;
        Q_data <= !data_xfc;
    end
    assign addr_ack_temp = Q_addr && addr_xfc;
    assign data_ack_temp = Q_data && data_xfc;

    //always block to perform address and data strobe
    //wire ack_not_RW = addr_ack_temp & !i2c_RW;
    reg stop_read;
    reg [10:0] i2c_addr_write;
    always@(posedge Clock or negedge reset)
    begin
        if(stop_out | !reset | stop_read)
        begin
            i2c_op <= 0;
            i2c_addr_out <= 0;
            i2c_data_out <= 0;
            i2c_xfc <= 0;
            addr_increment <= 0;
            stop_read <= 0;
            i2c_addr_write <= 0;
        end

    //Read Request Sequence
        else if(addr_ack_temp & !i2c_RW) //read request
        begin
        i2c_addr_out <= i2c_addr_in;
        i2c_op <= 0;
        xfc_ready <= 1;
        end
```

```
        else if(xfc_ready & !i2c_RW) //xfc high on address ready, READ
        begin
        i2c_xfc <= 1;
        xfc_ready <= 0;
        end

        else if(i2c_xfc & !i2c_RW) //zero xfc after 1 clock cycle for read, READ
        begin
        i2c_xfc <= 0;
        stop_read <= 1;
        end


    //Write Request Sequence
        else if(addr_ack_temp & i2c_RW) //write request store address
        begin
        i2c_op <= 1;
        i2c_addr_write <= i2c_addr_in;
        //xfc_ready <= 1;

        end

        else if(data_ack_temp & i2c_RW) //write send
        begin
        i2c_data_out <= i2c_data_in;
        i2c_addr_out <= i2c_addr_write + addr_increment;
        xfc_ready <= 1;
        end

        else if(xfc_ready & i2c_RW) //xfc high on data ready
        begin
        i2c_xfc <= 1;
        xfc_ready <= 0;
        end

        else if(i2c_xfc & i2c_RW) //zero xfc after 1 clock cycle for write
        begin
        i2c_xfc <= 0;
        addr_increment <= addr_increment + 1;
        i2c_data_out <= 0;
        i2c_addr_out <= 0;
        end
    end

endmodule
```

i2c_serializer.v:

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2c_serializer.v
// Create Date:             10/15/2016
// Last Modification:       1/13/2015
// Author:                  Whitley Forman
// Description:
////////////////////////////////////////////////////////////////////////////////

module Serializer(
          input i2c_scl,
    output reg i2c_sda_out,
          input i2c_ack,
    input Clock,
    input reset,
    input [7:0] i2c_rdata,
    input i2c_xfc_read
    );

    //Create SCL Pulse Signals, and states
    wire i2c_sda_neg_pulse;
    wire i2c_scl_neg_pulse;
    reg Q_scl;

    always@(posedge Clock)
    begin
        Q_scl = !i2c_scl;
    end

    assign i2c_scl_neg_pulse = !Q_scl && !i2c_scl;

    //Stop Conditions
    reg stop;
        //initial stop = 0;
    always@(posedge Clock or negedge reset)
    begin
        if (!reset /*| (!scl_state & i2c_sda_pos_pulse) */| serialize_done /*| stop_out*/)
        begin
        stop <= 1;
        end

        else if (i2c_xfc_read | i2c_ack)
        begin
        stop <= 0;
        end

        else if (stop)
        begin
        stop <=0;
        end
    end

    //Take in transfer Data
    reg [7:0] data_read;
    reg serialize_data_ready;

    always@(posedge Clock)
    begin
        if(stop)
        begin
        data_read = 8'b00000000;
        serialize_data_ready = 0;
        end

        else if(i2c_xfc_read)
        begin
        data_read = i2c_rdata;
```

```
    serialize_data_ready = 1;
    end
end


//Serialize ack for slave and data (register address and data)
reg [3:0] serialize_bit_counter;
reg serialize_done;
reg serialize_ack_done;
reg first_data_bit_ready;

wire i2c_ack_state;
assign i2c_ack_state = i2c_ack;

always@(posedge Clock)
begin
    if(stop)
    begin
    i2c_sda_out <=1;
    serialize_done <= 0;
    serialize_bit_counter <= 8'b00000000;
    first_data_bit_ready <= 0;
    serialize_ack_done <= 0;
    end

    //Serialize General i2c ACK for address match, data
    else if(i2c_ack_state)
    begin
    i2c_sda_out <= !i2c_ack_state;
    serialize_ack_done <= 1;
    end

    else if(!i2c_ack & serialize_ack_done)
    begin
    //serialize_done <= 1;
    serialize_ack_done <= 0;
    i2c_sda_out <= !i2c_ack;
    first_data_bit_ready <= 1;
    end

    else if(first_data_bit_ready & !serialize_data_ready)
    begin
    first_data_bit_ready <= 0;
    end

    else if(serialize_data_ready & (i2c_scl_neg_pulse | first_data_bit_ready))
    begin
    case(serialize_bit_counter)
    4'b0000 :  begin
                    i2c_sda_out <= data_read[7];
                    serialize_bit_counter <= 4'b0001;
                    first_data_bit_ready <= 0;
                    end
    4'b0001 :  begin
                    i2c_sda_out <= data_read[6];
                    serialize_bit_counter <= 4'b0010;
                    end
    4'b0010 :  begin
                    i2c_sda_out <= data_read[5];
                    serialize_bit_counter <= 4'b0011;
                    end
    4'b0011 :  begin
                    i2c_sda_out <= data_read[4];
                    serialize_bit_counter <= 4'b0100;
                    end
    4'b0100 :  begin
                    i2c_sda_out <= data_read[3];
                    serialize_bit_counter <= 4'b0101;
                    end
    4'b0101 :  begin
                    i2c_sda_out <= data_read[2];
```

```
                    serialize_bit_counter <= 4'b0110;
                    end
        4'b0110 :  begin
                    i2c_sda_out <= data_read[1];
                    serialize_bit_counter <= 4'b0111;
                    end
        4'b0111 :  begin
                    i2c_sda_out <= data_read[0];
                    serialize_bit_counter <= 4'b1000;
                    //serialize_done <= 1;
                    end
        4'b1000 :  begin
                    //i2c_sda_out <= data_read[0];
                    serialize_bit_counter <= 4'b0000;
                    serialize_done <= 1;
                    end
        endcase
        end

    end

endmodule
```

fifo.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             fifo.v
// Create Date:             7/13/2015
// Last Modification:       1/23/2016
// Author:                  Zachary Nelson
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define BUF_SIZE (1 << BUF_WIDTH)
         // Number of elements allowed in buffer = 2^Buffer Width

module
fifo(clk,rst_n,fifo_inp_data,fifo_out_data,fifo_inp_rts,fifo_out_rtr,fifo_out_rts,fifo_inp_rtr);

    parameter                               DATA_SIZE = 32;
// Number of bits for fifo data
    parameter                               BUF_WIDTH = 3;
// Number of bits to be used in pointer

    input                                   clk;
// Master clock
    input                                   rst_n;
// Reset

    input                                   fifo_inp_rts;
// Write client asserts ready to send
    output reg                              fifo_inp_rtr;
// Write client asserts ready to read
    input           [DATA_SIZE - 1:0]       fifo_inp_data;
// Data input to be pushed to buffer

    output reg                              fifo_out_rts;
// Output FIFO asserts read to send
    input                                   fifo_out_rtr;
    output reg      [DATA_SIZE - 1:0]       fifo_out_data;
// Port to output the data using pop.



    reg             [BUF_WIDTH    :0]       fifo_counter;
    reg             [BUF_WIDTH - 1:0]       rd_ptr;
// Pointer to read
    reg             [BUF_WIDTH - 1:0]       wr_ptr;
// Write addresses
    reg             [DATA_SIZE - 1:0]       buf_mem[`BUF_SIZE - 1:0];
// Buffer memory

    always @(fifo_counter)
    begin
       //Output FIFO is ready to send if the buffer is not empty
       fifo_out_rts = (fifo_counter != 0);

       //Output FIFO is ready to recieve if the buffer is not full
       fifo_inp_rtr = (fifo_counter != `BUF_SIZE);
    end

    //Update counter based on state of fifo
    always @(posedge clk or negedge rst_n)
    begin
       if(!rst_n)
           fifo_counter <= 0;
       //if both read and write occur
       else if((fifo_inp_rtr && fifo_inp_rts) && (fifo_out_rts && fifo_out_rtr))
           fifo_counter <= fifo_counter;
       //write enabled and buffer is not full
       else if(fifo_inp_rtr && fifo_inp_rts)
           //counter increased by 1
           fifo_counter <= (fifo_counter + 1'b1);
```

```
      //read enabled and buffer is not empty
      else if(fifo_out_rts && fifo_out_rtr)
          //counter decreased by 1
          fifo_counter <= (fifo_counter - 1'b1);
      else
         fifo_counter <= fifo_counter;
   end

   //Update output of the fifo
   always @(posedge clk or negedge rst_n)
   begin
      if(!rst_n)
         fifo_out_data <= 0;
      else
      begin
         //If read enabled and buffer is empty
         if(fifo_out_rtr && fifo_out_rts)
            //Output is equal to the data in memory at the read pointer
            fifo_out_data <= buf_mem[rd_ptr];
         else
            fifo_out_data <= fifo_out_data;
      end
   end

   //Update fifo memory
   always @(posedge clk)
   begin
      //If write is enabled and the buffer is not full
      if(fifo_inp_rts && fifo_inp_rtr)
         //Memory at location write pointer is now equal to the input
         buf_mem[wr_ptr] <= fifo_inp_data;
      else
         buf_mem[wr_ptr] <= buf_mem[wr_ptr];
   end

   //Update pointers
   always @(posedge clk or negedge rst_n)
   begin
      if(!rst_n)
      begin
         wr_ptr <= 0;
         rd_ptr <= 0;
      end
      else
      begin
         //If buffer is not full and write is enabled then the write pointer is increased by one
         if(fifo_inp_rtr && fifo_inp_rts)
             wr_ptr <= wr_ptr + 1'b1;
         else
             wr_ptr <= wr_ptr;
         //If buffer is not empty and read is enabled then the read pointer is increased by one
         if(fifo_out_rts && fifo_out_rtr)
             rd_ptr <= rd_ptr + 1'b1;
         else
             rd_ptr <= rd_ptr;
      end
   end

endmodule
```

i2s_in.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2s_in.v
// Create Date:             10/13/2015
// Last Modification:       3/25/2016
// Author:                  Kevin Cao
// Description:             Top Module of I2S In Interface
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module i2s_in(              clk, rst_n,
                            inp_sck, inp_ws, inp_sd,
                            rf_i2si_en, rf_bist_start_val, rf_bist_inc, rf_bist_up_limit,
rf_mux_en,
                            i2si_rtr, i2si_data, i2si_rts,
                            sync_sck, sync_sck_transition,
                            trig_fifo_overrun_clr,
                            ro_fifo_overrun
    );

    //Ports
    input                           clk;                                //Master clock
    input                           rst_n;                              //Reset

    input                           inp_sck;                            //Digital audio bit clock
    input                           inp_ws;                             //Word select - selects
what audio channel is being read. 0 = left channel, 1 = right channel
    input                           inp_sd;                             //Digital audio serial
data

    input                           rf_i2si_en;                         //Enable bit for
Deserializer. 0 = inactive, 1 = active
    input       [11:0]              rf_bist_start_val;                  //Bist start value
    input       [11:0]              rf_bist_up_limit;                   //Bist upper limit
    input       [ 7:0]              rf_bist_inc;                        //Bist increment signal
by this much
    input                           rf_mux_en;                          //Mux select bit for BIST
or Deserializer data/xfc signals

    input                           i2si_rtr;                           //Ready to read handshake
signal between I2S_In and Filter Block
    output                          i2si_rts;                           //Ready to send handshake
signal between I2S_IN and Filter Block
    output      [31:0]              i2si_data;                          //Output audio data sent
to Filter Block

    output                          sync_sck;                           //Delayed and
synchronized digital audio bit clock
    output                          sync_sck_transition;                //Level to pulse
converter of sync_sck;

    input                           trig_fifo_overrun_clr;              //Signal to reset
ro_fifo_overrun

    output reg                      ro_fifo_overrun;                    //The FIFO buffer is full
and no more can be added to the buffer



    //Internal Variables
    wire                            sync_sck_transition;                //Wire leading to level
to pulse converter of serial clock
    wire                            sync_ws;                            //Wire connecting
synchronizer output ws to deserializer input ws
    wire                            sync_sd;                            //Wire connecting
synchronizer output sd to deserializer input sd
```

```
    wire        [31:0]              deserializer_data;                  //Wire connecting output
audio data from deserializer to input of mux
    wire                            deserializer_xfc;                   //Wire connecting output
xfc signal from deserializer to input of mux

    wire        [31:0]              bist_data;                          //Wire connecting BIST
output data to input of mux
    wire                            bist_xfc;                           //Wire connecting BIST
xfc signal to input of mux

    wire        [31:0]              fifo_data;                          //Wire connecting output
data from mux to FIFO
    wire                            fifo_xfc;                           //Wire connecting output
xfc signal from mux to FIFO
    wire                            fifo_rtr;                           //Wire connecting output
rtr signal from FIFO to ovverun



    synchronizer Synchronizer(
        .clk                    (clk),
        .rst_n                  (rst_n),
        ._sck                   (inp_sck),
        ._ws                    (inp_ws),
        ._sd                    (inp_sd),
        .sck_transition         (sync_sck_transition),
        .sck                    (sync_sck),
        .ws                     (sync_ws),
        .sd                     (sync_sd)
    );

    i2si_deserializer Deserializer(
        .clk                    (clk),
        .rst_n                  (rst_n),
        .sck_transition         (sync_sck_transition),
        .in_ws                  (sync_ws),
        .in_sd                  (sync_sd),
        .rf_i2si_en             (rf_i2si_en),
        .out_lft                (deserializer_data [31:16]),
        .out_rgt                (deserializer_data [15: 0]),
        .out_xfc                (deserializer_xfc)
    );

    i2si_bist_gen Bist(
        .clk                    (clk),
        .rst_n                  (rst_n),
        .sck_transition         (sync_sck_transition),
        .rf_bist_start_val      (rf_bist_start_val),
        .rf_bist_up_limit       (rf_bist_up_limit),
        .rf_bist_inc            (rf_bist_inc),
        .i2si_bist_out_data     (bist_data),
        .i2si_bist_out_xfc      (bist_xfc)
    );

    i2si_mux Mux(
        .sel                    (rf_mux_en),
        .in_0_data              (deserializer_data),
        .in_0_xfc               (deserializer_xfc),
        .in_1_data              (bist_data),
        .in_1_xfc               (bist_xfc),
        .mux_data               (fifo_data),
        .mux_xfc                (fifo_xfc)
    );

    fifo #(32, 3) i2si_Fifo(
        .clk                    (clk),
        .rst_n                  (rst_n),
        .fifo_inp_data          (fifo_data),
        .fifo_inp_rts           (fifo_xfc),
        .fifo_inp_rtr           (fifo_rtr),
```

```
        .fifo_out_data          (i2si_data),
        .fifo_out_rtr           (i2si_rtr),
        .fifo_out_rts           (i2si_rts)
    );


    //Define ovverun signal
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            ro_fifo_overrun <= 1'b0;
        else if(~fifo_rtr & deserializer_xfc)
            ro_fifo_overrun <= 1'b1;
        else if(trig_fifo_overrun_clr)
            ro_fifo_overrun <= 1'b0;
    end

endmodule
```

i2si_bist_gen.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2si_bist_gen.v
// Create Date:             10/13/2015
// Last Modification:       3/20/2016
// Author:                  Kevin Cao, Zachary Nelson
// Description: Creates a saw-tooth wave based on the bist register values
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module
i2si_bist_gen(clk,rst_n,sck_transition,rf_bist_start_val,rf_bist_inc,rf_bist_up_limit,i2si_bist_o
ut_data, i2si_bist_out_xfc);

    //Ports
    input                       clk;                                        //Master Clock
    input                       rst_n;                                      //Reset
    input                       sck_transition;                             //Serial Clock
Level to Pulse Converter

    input           [11:0]      rf_bist_start_val;                          //Start value
    input           [11:0]      rf_bist_up_limit;                           //Upper limit
    input           [ 7:0]      rf_bist_inc;                                //Increment
signal by this much

    output reg      [31:0]      i2si_bist_out_data;                         //Output data
    output wire                 i2si_bist_out_xfc;                          //Transfer
Complete


    //Internal Variables
    reg                 bist_active;                                        //Defines if BIST
generator is active
    reg    [ 4:0]       sck_count;                                          //Serial clock
counter




    //define xfc signal as high after bist_out_data increments
    assign i2si_bist_out_xfc = bist_active && sck_count == 5'd31 && sck_transition;



    //serial clock counter. Helps define when bist_out_data when sck_count reaches 31
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            sck_count <= 5'd31;
        else if(sck_transition)
           sck_count <= sck_count + 1'b1;
    end

    //defines if bist generator becomes active
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            bist_active <= 1'b0;
        else if(sck_count == 5'd31 && sck_transition)
        begin
            if(!bist_active)
                bist_active <= 1'b1;
        end
    end

    //increments bist_out_data from the start value to the upper limit and then resets to the
start value again
```

```
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            i2si_bist_out_data[15: 0] <=  16'd0;
            i2si_bist_out_data[31:16] <= ~16'd0;
                    end
        else if (sck_count == 5'd31 && sck_transition)
        begin
            //If bist_active is just starting
            if(!bist_active)
                            begin
                //Output signal = start value
                i2si_bist_out_data[15: 0] <=  {rf_bist_start_val, 4'b0000};
                i2si_bist_out_data[31:16] <= ~{rf_bist_start_val, 4'b0000};
            end
            else if($signed(i2si_bist_out_data[15:0]) >= $signed({rf_bist_up_limit, 4'b0000}))
            begin
                //Signal goes back to start value
                i2si_bist_out_data[15:0]  <=  {rf_bist_start_val, 4'b0000};
                i2si_bist_out_data[31:16] <= ~{rf_bist_start_val, 4'b0000};

            end
            //If the signal is within normal range
            //Increment the signal
            else
            begin
                i2si_bist_out_data[15: 0] <=   i2si_bist_out_data[15:0] + {rf_bist_inc, 4'b0000};
                i2si_bist_out_data[31:16] <= ~(i2si_bist_out_data[15:0] + {rf_bist_inc,
4'b0000});

           end
        end
    end

endmodule
```

i2si_deserializer.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2si_deserializer
// Create Date:             9/13/2015
// Last Modification:       11/9/2015
// Author:                  Kevin Cao
//
//Description: The deserializer takes in serial data and converts it to parallel data that is
outputted to the mux and fifo sub-blocks.
//
//              Deserializer needs to first be resetted in order to function properly,
initializing all values to 0.
//              Once all values are initialized to 0, certain conditions must be met for the
deserializer to become active.
//              To become active, rst_n first needs to go from high to low, and then ws needs to
go from high to low.
//              Then when sck_transition high, the deserializer becomes active.
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module i2si_deserializer(clk, rst_n, sck_transition, in_ws, in_sd, rf_i2si_en, out_lft, out_rgt,
out_xfc);

    //Ports
    input                                   clk;                        //Master clock
    input                                   rst_n;                      //Reset
    input                                   sck_transition;             //Sck transitions from 0
-> 1. Helps tell the deserializer when to perform certain actions
    input                                   in_ws;                      //Word select: defines if
left or right channel is being read from. 0 = Left Channel, 1 = Right Channel
    input                                   in_sd;                      //Digital audio serial
data
    input                                   rf_i2si_en;                 //Enabled bit that helps
define if the deserializer is active or idle
    output reg      [15:0]                  out_lft;                    //Parallel output data of
left channel
    output reg      [15:0]                  out_rgt;                    //Parallel output data of
right channel
    output reg                              out_xfc;                    //Transfer Complete

    //Internal Variables
    reg             [ 1:0]                  rst_n_vec;                  //Used to check when
rst_n goes from low to high and to trigger armed1
    reg                                     armed1;                     //First signal that helps
define idle and active
    reg                                     armed2;                     //Second signal that
helps define idle and active
    reg                                     active;                     //Defines if the
deserializer is active or not
    reg                                     ws_d;                       //Delayed signal of in_ws
    reg                                     in_left;                    //Defines when the
deserializer should read in the left channel
    reg                                     in_left_delay;              //Delayed signal to help
define pre_xfc and out_xfc

    wire                                    ws_delay;                   //Delayed signal of ws
    wire                                    ws_transition;              //Check if ws goes from 1
-> 0 when en = 1
    wire                                    pre_xfc;                    //Unsynchronized transfer
complete signal



    //delay ws signal
    //used to help create ws_transition to define the deserializer as active
    always @(posedge clk or negedge rst_n)
    begin
```

```
        if(!rst_n)
            ws_d <= 1'b0;
        else if(sck_transition)
            ws_d <= in_ws;
    end

    //Re-assigning ws to be more readable
    assign ws_delay = ws_d;

    //ws_transition becomes high when ws goes from 1 -> 0
    //used to help define if deserializer is active
    assign ws_transition = !in_ws && ws_delay;

    //Used to help define active when rst_n goes from low to high
    always @(posedge clk)
    begin
        rst_n_vec[0] <= rst_n;
        rst_n_vec[1] <= rst_n_vec[0];
    end

    //Intermediate step to help define active
    //checks if rst_n goes from high to low
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            armed1 <= 1'b0;
        else if(!rst_n_vec[1] && rst_n_vec[0])
            armed1 <= 1'b1;
        else if(ws_transition)
            armed1 <= 1'b0;
    end

    //Intermediate step to help define active
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            armed2 <= 1'b0;
        else if(armed1 && ws_transition)
            armed2 <= 1'b1;
        else if(sck_transition)
            armed2 <= 1'b0;
    end

    //Defines when deserializer is idle or active
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            active <= 1'b0;
        else if(!rf_i2si_en)
            active <= 1'b0;
        else if(armed2 && sck_transition)
            active <= 1'b1;
    end


    //Tells deserializer when to start reading in data from left channel
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            in_left <= 1'b1;
        else if (!in_ws && sck_transition && active)
            in_left <= 1'b1;
        else if (in_ws && sck_transition && active)
            in_left <= 1'b0;
    end

    //Used to help trigger
out_xfc


    always @(posedge clk or negedge rst_n)
```

```
    begin
        if(!rst_n)
            in_left_delay <= 1'b0;
        else
        begin
            in_left_delay <= in_left;
        end
    end

    //Triggers out_xfc when ws[3] goes from high to low
    //In other words when the system is done reading in the right channel
    //and begins reading in the left channel, trigger
xfc.


    assign pre_xfc = in_left && !in_left_delay;

    //synchronizing xfc with master clock
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            out_xfc <= 1'b0;
        else
            out_xfc <= pre_xfc;
    end


    //Store data into either the left or right channel when
    //the deserializer is active and when sck_transition is high
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            out_lft[15:0] <= 16'b0;
            out_rgt[15:0] <= 16'b0;
        end
        else if(active)
        begin
            if(sck_transition)
            begin
                if (in_left)
                begin
                    out_lft[15:1] <= out_lft[14:0];
                    out_lft[0]    <= in_sd;
                end
                else
                begin
                    out_rgt[15:1] <= out_rgt[14:0];
                    out_rgt[0]    <= in_sd;
                end
            end
        end
    end

endmodule
```

i2si_mux.v:

```
////////////////////////////////////////////////////////////////////////////////////////////////
///////////////
// Module Name:             i2si_mux.v
// Create Date:             10/13/2015
// Last Modification:       1/23/2016
// Author:                  Kevin Cao
//
//Description: Multiplexer that outputs either the BIST or Deserializer input data and xfc
signals
////////////////////////////////////////////////////////////////////////////////////////////////
///////////////

`timescale 1ns / 1ps

module i2si_mux(in_0_data, in_0_xfc, in_1_data, in_1_xfc, sel, mux_data, mux_xfc);

    //Ports
    input           [31:0]              in_0_data;                  //First input data value
    input                               in_0_xfc;                   //First input xfc value
    input           [31:0]              in_1_data;                  //Second input data value
    input                               in_1_xfc;                   //Second input xfc value
    input                               sel;                        //select input to select
either input 0 or 1

    output reg      [31:0]              mux_data;                   //mux data output value
    output reg                          mux_xfc;                    //mux xfc output value


    always @ (sel or in_0_data or in_0_xfc or in_1_data or in_1_xfc)
    begin
        if (sel == 1'b0)
        begin
            mux_data <= in_0_data;
            mux_xfc  <= in_0_xfc;
        end
        else
        begin
            mux_data <= in_1_data;
            mux_xfc  <= in_1_xfc;
        end
    end

endmodule
```

i2si_synchronizer.v:

```
////////////////////////////////////////////////////////////////////////////////
//
// Module Name:             synchronizer.v
// Create Date:             10/20/2015
// Last Modification:       3/20/2016
// Author:                  Kevin Cao
//
//Description:      Delays and synchronizes the sck, sd, and ws signals with master clock
//                      sck: 2 clk cycles
//                      sd : 4 clk cycles
//                      ws : 4 clk cycles
//
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module synchronizer(clk, rst_n, _sck, sck, sck_transition, _sd, sd, _ws, ws
    );

    //Ports
    input               clk;                        //Master clock
    input               rst_n;                      //Reset
    input               _sck;                       //Non-delayed and non-synchronized sck signal
    input               _sd;                        //Non-delayed and non-synchronized serial
data signal
    input               _ws;                        //Non-delayed and non-synchronized word
select signal

    output              sck;                        //Delayed and Synchronized sck
    output              sck_transition;             //Signal that represents when sck goes from
low to high. Helps define when particular actions should occur
    output              sd;                         //Delayed and Synchronized serial data signal
    output              ws;                         //Delayed and Synchronized word select signal


    //Internal Variables
    reg     [2:0]       sck_vec;
    reg     [3:0]       sd_vec;
    reg     [3:0]       ws_vec;

    wire                sck_delay;                  //Delayed sck signal that helps define
sck_transition



    //Delay sck by 2 clk cycles and synchronize with clk
    //sck[1] = sck synchronized with clk
    //sck[2] = sck delay signal to help create sck_transition
    //Delay and synchronize ws signal by 4 clock cycles
    //ws[3] is the synchronized signal
    //Delay and synchronize sd by 4 clock cycles
    //sd[3] is the synchronized signal
    always @(posedge clk or negedge rst_n)
    begin
        if (!rst_n)
         begin
            sck_vec <= 3'b000;
            ws_vec  <= 4'b0000;
            sd_vec  <= 3'b000;
        end
        else
        begin
            sck_vec[0]   <= _sck;
            sck_vec[2:1] <= sck_vec[1:0];

            ws_vec[0]    <= _ws;
            ws_vec[3:1]  <= ws_vec[2:0];
```

```
            sd_vec[0]    <= _sd;
            sd_vec[3:1]  <= sd_vec[2:0];
        end
    end

    //Re-assigning sck to be more readable
    assign sck = sck_vec[1];

    //Create additional sck delay signal to detect rising edge
    assign sck_delay = sck_vec[2];

    //Defines sck_transition as high when sck transitions from low to high
    //helps define when particular actions should occur
    assign sck_transition = sck && !sck_delay;

    //Re-assigning sd to be more readable
    assign sd = sd_vec[3];

    //Re-assigning ws to be more readable
    assign ws = ws_vec[3];

endmodule
```

i2s_out.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             i2s_out.v
// Create Date:             10/13/2015
// Last Modification:       3/25/2015
// Author:                  Kevin Cao
// Description:             Top module of I2S Out Interface
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module i2s_out(                 clk, rst_n,
                                i2so_sync_sck, i2so_sck_transition,
                                filt_rts, filt_data, filt_rtr,
                                i2so_ws, i2so_sd, i2so_sck,
                                trig_fifo_underrun,
                                ro_fifo_underrun
    );

    input                           clk;                                //Master clock
    input                           rst_n;                              //Reset

    input                           i2so_sync_sck;                      //Serial clock
    input                           i2so_sck_transition;                //Serial clock level to
pulse converter

    output                          i2so_sck;
    output                          i2so_ws;                            //Word select - selects
what audio channel is being read. 0 = left channel, 1 = right channel
    output                          i2so_sd;                            //Digital audio serial
data

    input                           filt_rts;                           //Ready to send
handshake signal between Filter and I2S_OUT Block
    output                          filt_rtr;                           //Ready to read
handshake signal between Filter and I2S_OUT Block
    input       [31:0]              filt_data;                          //Output audio data sent
from Filter Block to I2S_OUT Block

    input                           trig_fifo_underrun;                 //Signal to reset
ro_fifo_underrun

    output reg                      ro_fifo_underrun;                   //The FIFO buffer is not
full and no more data is available

    wire                            sck_out;                            //Wire connecting sck
synchronizer to sck deserializer  NOT NEEDED???
    wire                            i2so_sck_transition;                //Wire connecting
i2so_sck_transition signal to other blocks
    wire                            fifo_rts;                           //Ready to send
handshake signal between FIFO and Serializer
    wire                            fifo_rtr;                           //Ready to read
handshake signal between FIFO and Serializer
    wire        [31:0]              fifo_data;                          //Wire connecting 32 bit
audio data from FIFO to Serializer


    i2so_serializer Serializer(
        .clk                    (clk),
        .rst_n                  (rst_n),
        .sck_transition         (i2so_sck_transition),
        .filt_i2so_lft          (fifo_data [31:16]),
        .filt_i2so_rgt          (fifo_data [15: 0]),
        .filt_i2so_rts          (fifo_rts),
        .filt_i2so_rtr          (fifo_rtr),
        .i2so_ws                (i2so_ws),
        .i2so_sd                (i2so_sd)
    );
```

```
    fifo #(32, 3) i2so_Fifo(
        .clk                    (clk),
        .rst_n                  (rst_n),
        .fifo_inp_rts           (filt_rts),
        .fifo_inp_rtr           (filt_rtr),
        .fifo_inp_data          (filt_data),
        .fifo_out_rts           (fifo_rts),
        .fifo_out_rtr           (fifo_rtr),
        .fifo_out_data          (fifo_data)
    );


    assign i2so_sck = i2so_sync_sck;

    //Define underrun signal
    always @ (posedge clk or negedge rst_n)
    begin
        if (!rst_n)
            ro_fifo_underrun <= 0;
        else if (~fifo_rts & fifo_rtr)
            ro_fifo_underrun <= 1;
        else if (trig_fifo_underrun)
            ro_fifo_underrun <= 0;
    end


endmodule
```

i2so_serializer.v:

```
 ////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:24:34 10/20/2015
// Design Name:
// Module Name:    serializer
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module i2so_serializer(clk, rst_n, filt_i2so_rts, i2so_sd, i2so_ws, filt_i2so_lft, filt_i2so_rgt,
filt_i2so_rtr, sck_transition
    );

    input                       clk;                                //Master Clock
    input                       rst_n;                              //Reset
    input                       sck_transition;                     //Pulse when sck transitions
from low to high

    input                       filt_i2so_rts;                      //ready to send
    output wire                 filt_i2so_rtr;                      //Ready to receive
    input           [15:0]      filt_i2so_lft;                      //Left parallel digital audio
data
    input           [15:0]      filt_i2so_rgt;                      //Right parallel digital
audio data

    output reg                  i2so_sd;                            //i2s output serial data
    output reg                  i2so_ws;                            //i2s output word Select

    //Internal Variables
    reg                         serializer_active;
    reg                         filt_i2so_rts_delay;                //Delay signal of ready to
send
    reg             [15:0]      lft_data;                           //Captures the data of
filt_i2so_lft
    reg             [15:0]      rgt_data;                           //Captures the data of
filt_i2so_rgt
    reg                         LR;                                 //Left Right Counter: keeps
track of which parallel digital audio to read from
    reg             [ 3:0]      bit_count;                          //Bit Counter: keeps track of
which bit to read in
    wire                        filt_i2so_rts_transition;           //High when filt_i2so_rts
goes from low to high


    //Helps create filt_i2so_rts_transition signal to define when the serializer is in the active
state
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            filt_i2so_rts_delay <= 1'b0;
        else
            filt_i2so_rts_delay <= filt_i2so_rts;
    end

    assign filt_i2so_rts_transition = filt_i2so_rts && !filt_i2so_rts_delay;
```

```
    //Serializer becomes active when filt_i2so_rts transitions from low to high
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            serializer_active <= 0;
        else if(filt_i2so_rts_transition)
            serializer_active <= 1'b1;
    end


    //Tells the serializer to read from filt_i2so_lft or filt_i2so_rgt
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            LR <= 1'b1;
        else if(bit_count == 0 && sck_transition && serializer_active)
            LR <= ~LR;
    end

    assign filt_i2so_rtr = sck_transition && serializer_active && (bit_count == 0) && LR;

    //Capture data in filt_i2so_lft or filt_i2so_rgt during first filt_i2so_rts_transition or
during LR_transition
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            lft_data <= 0;
            rgt_data <= 0;
        end

        else if(serializer_active && filt_i2so_rtr)
        begin
            lft_data <= filt_i2so_lft;
            rgt_data <= filt_i2so_rgt;
        end
    end


    //Keeps track of which bit of the channel to read from to store in i2so_sd
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            bit_count <= 4'd0;
        else if(filt_i2so_rtr)
            bit_count <= 4'd15;
        else if(sck_transition && serializer_active)
              bit_count <= bit_count - 4'd1;
    end


    //Change ws when channel is on 15th bit or bit [1]
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            i2so_ws <= 1'b0;
        else if (serializer_active && bit_count == 4'd1 && sck_transition)
        begin
            i2so_ws <= ~i2so_ws;
        end
    end


    //Store bit data from filt_i2so_lft or filt_i2so_rgt into i2so_sd
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
            i2so_sd <= 1'b0;
        else if(serializer_active)
```

```
        begin
            if(LR == 1'b0)
            begin
                i2so_sd <= lft_data[bit_count];
            end

            else
            begin
                i2so_sd <= rgt_data[bit_count];
            end
        end
    end

endmodule
```

chip_reg.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             reg.v
// Create Date:             10/15/2015
// Last Modification:       3/16/2016
// Author:                  Julie Swift and Zachary Nelson
// Description: Provides read and write status to control/status registers
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module chip_reg(clk,rst_n,i2c_addr,i2c_wdata,i2c_xfc_write,i2c_op,
                ro_fifo_overrun,ro_fifo_underrun,ro_filter_ovf_flag,
                i2c_rdata,i2c_xfc_read,

rf_i2si_bist_en,rf_filter_shift,rf_filter_clip_en,rf_i2si_bist_start_val_a,rf_i2si_bist_start_val
_b,rf_i2si_bist_incr,rf_i2si_bist_upper_limit_a,rf_i2si_bist_upper_limit_b,rf_i2si_en,
                rf_filter_coeff0_a, rf_filter_coeff0_b,rf_filter_coeff1_a,
rf_filter_coeff1_b,rf_filter_coeff2_a, rf_filter_coeff2_b,rf_filter_coeff3_a,
rf_filter_coeff3_b,rf_filter_coeff4_a, rf_filter_coeff4_b,rf_filter_coeff5_a,
rf_filter_coeff5_b,rf_filter_coeff6_a, rf_filter_coeff6_b,rf_filter_coeff7_a,
rf_filter_coeff7_b,rf_filter_coeff8_a, rf_filter_coeff8_b,rf_filter_coeff9_a,
rf_filter_coeff9_b,rf_filter_coeff10_a, rf_filter_coeff10_b,rf_filter_coeff11_a,
rf_filter_coeff11_b,rf_filter_coeff12_a, rf_filter_coeff12_b,rf_filter_coeff13_a,
rf_filter_coeff13_b,rf_filter_coeff14_a, rf_filter_coeff14_b,rf_filter_coeff15_a,
rf_filter_coeff15_b,rf_filter_coeff16_a, rf_filter_coeff16_b,rf_filter_coeff17_a,
rf_filter_coeff17_b,rf_filter_coeff18_a, rf_filter_coeff18_b,rf_filter_coeff19_a,
rf_filter_coeff19_b,rf_filter_coeff20_a, rf_filter_coeff20_b,rf_filter_coeff21_a,
rf_filter_coeff21_b,rf_filter_coeff22_a, rf_filter_coeff22_b,rf_filter_coeff23_a,
rf_filter_coeff23_b,rf_filter_coeff24_a, rf_filter_coeff24_b,rf_filter_coeff25_a,
rf_filter_coeff25_b,rf_filter_coeff26_a, rf_filter_coeff26_b,rf_filter_coeff27_a,
rf_filter_coeff27_b,rf_filter_coeff28_a, rf_filter_coeff28_b,rf_filter_coeff29_a,
rf_filter_coeff29_b,rf_filter_coeff30_a, rf_filter_coeff30_b,rf_filter_coeff31_a,
rf_filter_coeff31_b,rf_filter_coeff32_a, rf_filter_coeff32_b,rf_filter_coeff33_a,
rf_filter_coeff33_b,rf_filter_coeff34_a, rf_filter_coeff34_b,rf_filter_coeff35_a,
rf_filter_coeff35_b,rf_filter_coeff36_a, rf_filter_coeff36_b,rf_filter_coeff37_a,
rf_filter_coeff37_b,rf_filter_coeff38_a, rf_filter_coeff38_b,rf_filter_coeff39_a,
rf_filter_coeff39_b,rf_filter_coeff40_a, rf_filter_coeff40_b,rf_filter_coeff41_a,
rf_filter_coeff41_b,rf_filter_coeff42_a, rf_filter_coeff42_b,rf_filter_coeff43_a,
rf_filter_coeff43_b,rf_filter_coeff44_a, rf_filter_coeff44_b,rf_filter_coeff45_a,
rf_filter_coeff45_b,rf_filter_coeff46_a, rf_filter_coeff46_b,rf_filter_coeff47_a,
rf_filter_coeff47_b,rf_filter_coeff48_a, rf_filter_coeff48_b,rf_filter_coeff49_a,
rf_filter_coeff49_b,rf_filter_coeff50_a, rf_filter_coeff50_b,rf_filter_coeff51_a,
rf_filter_coeff51_b,rf_filter_coeff52_a, rf_filter_coeff52_b,rf_filter_coeff53_a,
rf_filter_coeff53_b,rf_filter_coeff54_a, rf_filter_coeff54_b,rf_filter_coeff55_a,
rf_filter_coeff55_b,rf_filter_coeff56_a, rf_filter_coeff56_b,rf_filter_coeff57_a,
rf_filter_coeff57_b,rf_filter_coeff58_a, rf_filter_coeff58_b,rf_filter_coeff59_a,
rf_filter_coeff59_b,rf_filter_coeff60_a, rf_filter_coeff60_b,rf_filter_coeff61_a,
rf_filter_coeff61_b,rf_filter_coeff62_a, rf_filter_coeff62_b,rf_filter_coeff63_a,
rf_filter_coeff63_b,rf_filter_coeff64_a, rf_filter_coeff64_b,rf_filter_coeff65_a,
rf_filter_coeff65_b,rf_filter_coeff66_a, rf_filter_coeff66_b,rf_filter_coeff67_a,
rf_filter_coeff67_b,rf_filter_coeff68_a, rf_filter_coeff68_b,rf_filter_coeff69_a,
rf_filter_coeff69_b,rf_filter_coeff70_a, rf_filter_coeff70_b,rf_filter_coeff71_a,
rf_filter_coeff71_b,rf_filter_coeff72_a, rf_filter_coeff72_b,rf_filter_coeff73_a,
rf_filter_coeff73_b,rf_filter_coeff74_a, rf_filter_coeff74_b,rf_filter_coeff75_a,
rf_filter_coeff75_b,rf_filter_coeff76_a, rf_filter_coeff76_b,rf_filter_coeff77_a,
rf_filter_coeff77_b,rf_filter_coeff78_a, rf_filter_coeff78_b,rf_filter_coeff79_a,
rf_filter_coeff79_b,rf_filter_coeff80_a, rf_filter_coeff80_b,rf_filter_coeff81_a,
rf_filter_coeff81_b,rf_filter_coeff82_a, rf_filter_coeff82_b,rf_filter_coeff83_a,
rf_filter_coeff83_b,rf_filter_coeff84_a, rf_filter_coeff84_b,rf_filter_coeff85_a,
rf_filter_coeff85_b,rf_filter_coeff86_a, rf_filter_coeff86_b,rf_filter_coeff87_a,
rf_filter_coeff87_b,rf_filter_coeff88_a, rf_filter_coeff88_b,rf_filter_coeff89_a,
rf_filter_coeff89_b,rf_filter_coeff90_a, rf_filter_coeff90_b,rf_filter_coeff91_a,
rf_filter_coeff91_b,rf_filter_coeff92_a, rf_filter_coeff92_b,rf_filter_coeff93_a,
rf_filter_coeff93_b,rf_filter_coeff94_a, rf_filter_coeff94_b,rf_filter_coeff95_a,
rf_filter_coeff95_b,rf_filter_coeff96_a, rf_filter_coeff96_b,rf_filter_coeff97_a,
rf_filter_coeff97_b,rf_filter_coeff98_a, rf_filter_coeff98_b,rf_filter_coeff99_a,
rf_filter_coeff99_b,rf_filter_coeff100_a, rf_filter_coeff100_b,rf_filter_coeff101_a,
rf_filter_coeff101_b,rf_filter_coeff102_a, rf_filter_coeff102_b,rf_filter_coeff103_a,
```

rf_filter_coeff103_b,rf_filter_coeff104_a, rf_filter_coeff104_b,rf_filter_coeff105_a,
rf_filter_coeff105_b,rf_filter_coeff106_a, rf_filter_coeff106_b,rf_filter_coeff107_a,
rf_filter_coeff107_b,rf_filter_coeff108_a, rf_filter_coeff108_b,rf_filter_coeff109_a,
rf_filter_coeff109_b,rf_filter_coeff110_a, rf_filter_coeff110_b,rf_filter_coeff111_a,
rf_filter_coeff111_b,rf_filter_coeff112_a, rf_filter_coeff112_b,rf_filter_coeff113_a,
rf_filter_coeff113_b,rf_filter_coeff114_a, rf_filter_coeff114_b,rf_filter_coeff115_a,
rf_filter_coeff115_b,rf_filter_coeff116_a, rf_filter_coeff116_b,rf_filter_coeff117_a,
rf_filter_coeff117_b,rf_filter_coeff118_a, rf_filter_coeff118_b,rf_filter_coeff119_a,
rf_filter_coeff119_b,rf_filter_coeff120_a, rf_filter_coeff120_b,rf_filter_coeff121_a,
rf_filter_coeff121_b,rf_filter_coeff122_a, rf_filter_coeff122_b,rf_filter_coeff123_a,
rf_filter_coeff123_b,rf_filter_coeff124_a, rf_filter_coeff124_b,rf_filter_coeff125_a,
rf_filter_coeff125_b,rf_filter_coeff126_a, rf_filter_coeff126_b,rf_filter_coeff127_a,
rf_filter_coeff127_b,rf_filter_coeff128_a, rf_filter_coeff128_b,rf_filter_coeff129_a,
rf_filter_coeff129_b,rf_filter_coeff130_a, rf_filter_coeff130_b,rf_filter_coeff131_a,
rf_filter_coeff131_b,rf_filter_coeff132_a, rf_filter_coeff132_b,rf_filter_coeff133_a,
rf_filter_coeff133_b,rf_filter_coeff134_a, rf_filter_coeff134_b,rf_filter_coeff135_a,
rf_filter_coeff135_b,rf_filter_coeff136_a, rf_filter_coeff136_b,rf_filter_coeff137_a,
rf_filter_coeff137_b,rf_filter_coeff138_a, rf_filter_coeff138_b,rf_filter_coeff139_a,
rf_filter_coeff139_b,rf_filter_coeff140_a, rf_filter_coeff140_b,rf_filter_coeff141_a,
rf_filter_coeff141_b,rf_filter_coeff142_a, rf_filter_coeff142_b,rf_filter_coeff143_a,
rf_filter_coeff143_b,rf_filter_coeff144_a, rf_filter_coeff144_b,rf_filter_coeff145_a,
rf_filter_coeff145_b,rf_filter_coeff146_a, rf_filter_coeff146_b,rf_filter_coeff147_a,
rf_filter_coeff147_b,rf_filter_coeff148_a, rf_filter_coeff148_b,rf_filter_coeff149_a,
rf_filter_coeff149_b,rf_filter_coeff150_a, rf_filter_coeff150_b,rf_filter_coeff151_a,
rf_filter_coeff151_b,rf_filter_coeff152_a, rf_filter_coeff152_b,rf_filter_coeff153_a,
rf_filter_coeff153_b,rf_filter_coeff154_a, rf_filter_coeff154_b,rf_filter_coeff155_a,
rf_filter_coeff155_b,rf_filter_coeff156_a, rf_filter_coeff156_b,rf_filter_coeff157_a,
rf_filter_coeff157_b,rf_filter_coeff158_a, rf_filter_coeff158_b,rf_filter_coeff159_a,
rf_filter_coeff159_b,rf_filter_coeff160_a, rf_filter_coeff160_b,rf_filter_coeff161_a,
rf_filter_coeff161_b,rf_filter_coeff162_a, rf_filter_coeff162_b,rf_filter_coeff163_a,
rf_filter_coeff163_b,rf_filter_coeff164_a, rf_filter_coeff164_b,rf_filter_coeff165_a,
rf_filter_coeff165_b,rf_filter_coeff166_a, rf_filter_coeff166_b,rf_filter_coeff167_a,
rf_filter_coeff167_b,rf_filter_coeff168_a, rf_filter_coeff168_b,rf_filter_coeff169_a,
rf_filter_coeff169_b,rf_filter_coeff170_a, rf_filter_coeff170_b,rf_filter_coeff171_a,
rf_filter_coeff171_b,rf_filter_coeff172_a, rf_filter_coeff172_b,rf_filter_coeff173_a,
rf_filter_coeff173_b,rf_filter_coeff174_a, rf_filter_coeff174_b,rf_filter_coeff175_a,
rf_filter_coeff175_b,rf_filter_coeff176_a, rf_filter_coeff176_b,rf_filter_coeff177_a,
rf_filter_coeff177_b,rf_filter_coeff178_a, rf_filter_coeff178_b,rf_filter_coeff179_a,
rf_filter_coeff179_b,rf_filter_coeff180_a, rf_filter_coeff180_b,rf_filter_coeff181_a,
rf_filter_coeff181_b,rf_filter_coeff182_a, rf_filter_coeff182_b,rf_filter_coeff183_a,
rf_filter_coeff183_b,rf_filter_coeff184_a, rf_filter_coeff184_b,rf_filter_coeff185_a,
rf_filter_coeff185_b,rf_filter_coeff186_a, rf_filter_coeff186_b,rf_filter_coeff187_a,
rf_filter_coeff187_b,rf_filter_coeff188_a, rf_filter_coeff188_b,rf_filter_coeff189_a,
rf_filter_coeff189_b,rf_filter_coeff190_a, rf_filter_coeff190_b,rf_filter_coeff191_a,
rf_filter_coeff191_b,rf_filter_coeff192_a, rf_filter_coeff192_b,rf_filter_coeff193_a,
rf_filter_coeff193_b,rf_filter_coeff194_a, rf_filter_coeff194_b,rf_filter_coeff195_a,
rf_filter_coeff195_b,rf_filter_coeff196_a, rf_filter_coeff196_b,rf_filter_coeff197_a,
rf_filter_coeff197_b,rf_filter_coeff198_a, rf_filter_coeff198_b,rf_filter_coeff199_a,
rf_filter_coeff199_b,rf_filter_coeff200_a, rf_filter_coeff200_b,rf_filter_coeff201_a,
rf_filter_coeff201_b,rf_filter_coeff202_a, rf_filter_coeff202_b,rf_filter_coeff203_a,
rf_filter_coeff203_b,rf_filter_coeff204_a, rf_filter_coeff204_b,rf_filter_coeff205_a,
rf_filter_coeff205_b,rf_filter_coeff206_a, rf_filter_coeff206_b,rf_filter_coeff207_a,
rf_filter_coeff207_b,rf_filter_coeff208_a, rf_filter_coeff208_b,rf_filter_coeff209_a,
rf_filter_coeff209_b,rf_filter_coeff210_a, rf_filter_coeff210_b,rf_filter_coeff211_a,
rf_filter_coeff211_b,rf_filter_coeff212_a, rf_filter_coeff212_b,rf_filter_coeff213_a,
rf_filter_coeff213_b,rf_filter_coeff214_a, rf_filter_coeff214_b,rf_filter_coeff215_a,
rf_filter_coeff215_b,rf_filter_coeff216_a, rf_filter_coeff216_b,rf_filter_coeff217_a,
rf_filter_coeff217_b,rf_filter_coeff218_a, rf_filter_coeff218_b,rf_filter_coeff219_a,
rf_filter_coeff219_b,rf_filter_coeff220_a, rf_filter_coeff220_b,rf_filter_coeff221_a,
rf_filter_coeff221_b,rf_filter_coeff222_a, rf_filter_coeff222_b,rf_filter_coeff223_a,
rf_filter_coeff223_b,rf_filter_coeff224_a, rf_filter_coeff224_b,rf_filter_coeff225_a,
rf_filter_coeff225_b,rf_filter_coeff226_a, rf_filter_coeff226_b,rf_filter_coeff227_a,
rf_filter_coeff227_b,rf_filter_coeff228_a, rf_filter_coeff228_b,rf_filter_coeff229_a,
rf_filter_coeff229_b,rf_filter_coeff230_a, rf_filter_coeff230_b,rf_filter_coeff231_a,
rf_filter_coeff231_b,rf_filter_coeff232_a, rf_filter_coeff232_b,rf_filter_coeff233_a,
rf_filter_coeff233_b,rf_filter_coeff234_a, rf_filter_coeff234_b,rf_filter_coeff235_a,
rf_filter_coeff235_b,rf_filter_coeff236_a, rf_filter_coeff236_b,rf_filter_coeff237_a,
rf_filter_coeff237_b,rf_filter_coeff238_a, rf_filter_coeff238_b,rf_filter_coeff239_a,
rf_filter_coeff239_b,rf_filter_coeff240_a, rf_filter_coeff240_b,rf_filter_coeff241_a,
rf_filter_coeff241_b,rf_filter_coeff242_a, rf_filter_coeff242_b,rf_filter_coeff243_a,
rf_filter_coeff243_b,rf_filter_coeff244_a, rf_filter_coeff244_b,rf_filter_coeff245_a,

rf_filter_coeff245_b,rf_filter_coeff246_a, rf_filter_coeff246_b,rf_filter_coeff247_a,
rf_filter_coeff247_b,rf_filter_coeff248_a, rf_filter_coeff248_b,rf_filter_coeff249_a,
rf_filter_coeff249_b,rf_filter_coeff250_a, rf_filter_coeff250_b,rf_filter_coeff251_a,
rf_filter_coeff251_b,rf_filter_coeff252_a, rf_filter_coeff252_b,rf_filter_coeff253_a,
rf_filter_coeff253_b,rf_filter_coeff254_a, rf_filter_coeff254_b,rf_filter_coeff255_a,
rf_filter_coeff255_b,rf_filter_coeff256_a, rf_filter_coeff256_b,rf_filter_coeff257_a,
rf_filter_coeff257_b,rf_filter_coeff258_a, rf_filter_coeff258_b,rf_filter_coeff259_a,
rf_filter_coeff259_b,rf_filter_coeff260_a, rf_filter_coeff260_b,rf_filter_coeff261_a,
rf_filter_coeff261_b,rf_filter_coeff262_a, rf_filter_coeff262_b,rf_filter_coeff263_a,
rf_filter_coeff263_b,rf_filter_coeff264_a, rf_filter_coeff264_b,rf_filter_coeff265_a,
rf_filter_coeff265_b,rf_filter_coeff266_a, rf_filter_coeff266_b,rf_filter_coeff267_a,
rf_filter_coeff267_b,rf_filter_coeff268_a, rf_filter_coeff268_b,rf_filter_coeff269_a,
rf_filter_coeff269_b,rf_filter_coeff270_a, rf_filter_coeff270_b,rf_filter_coeff271_a,
rf_filter_coeff271_b,rf_filter_coeff272_a, rf_filter_coeff272_b,rf_filter_coeff273_a,
rf_filter_coeff273_b,rf_filter_coeff274_a, rf_filter_coeff274_b,rf_filter_coeff275_a,
rf_filter_coeff275_b,rf_filter_coeff276_a, rf_filter_coeff276_b,rf_filter_coeff277_a,
rf_filter_coeff277_b,rf_filter_coeff278_a, rf_filter_coeff278_b,rf_filter_coeff279_a,
rf_filter_coeff279_b,rf_filter_coeff280_a, rf_filter_coeff280_b,rf_filter_coeff281_a,
rf_filter_coeff281_b,rf_filter_coeff282_a, rf_filter_coeff282_b,rf_filter_coeff283_a,
rf_filter_coeff283_b,rf_filter_coeff284_a, rf_filter_coeff284_b,rf_filter_coeff285_a,
rf_filter_coeff285_b,rf_filter_coeff286_a, rf_filter_coeff286_b,rf_filter_coeff287_a,
rf_filter_coeff287_b,rf_filter_coeff288_a, rf_filter_coeff288_b,rf_filter_coeff289_a,
rf_filter_coeff289_b,rf_filter_coeff290_a, rf_filter_coeff290_b,rf_filter_coeff291_a,
rf_filter_coeff291_b,rf_filter_coeff292_a, rf_filter_coeff292_b,rf_filter_coeff293_a,
rf_filter_coeff293_b,rf_filter_coeff294_a, rf_filter_coeff294_b,rf_filter_coeff295_a,
rf_filter_coeff295_b,rf_filter_coeff296_a, rf_filter_coeff296_b,rf_filter_coeff297_a,
rf_filter_coeff297_b,rf_filter_coeff298_a, rf_filter_coeff298_b,rf_filter_coeff299_a,
rf_filter_coeff299_b,rf_filter_coeff300_a, rf_filter_coeff300_b,rf_filter_coeff301_a,
rf_filter_coeff301_b,rf_filter_coeff302_a, rf_filter_coeff302_b,rf_filter_coeff303_a,
rf_filter_coeff303_b,rf_filter_coeff304_a, rf_filter_coeff304_b,rf_filter_coeff305_a,
rf_filter_coeff305_b,rf_filter_coeff306_a, rf_filter_coeff306_b,rf_filter_coeff307_a,
rf_filter_coeff307_b,rf_filter_coeff308_a, rf_filter_coeff308_b,rf_filter_coeff309_a,
rf_filter_coeff309_b,rf_filter_coeff310_a, rf_filter_coeff310_b,rf_filter_coeff311_a,
rf_filter_coeff311_b,rf_filter_coeff312_a, rf_filter_coeff312_b,rf_filter_coeff313_a,
rf_filter_coeff313_b,rf_filter_coeff314_a, rf_filter_coeff314_b,rf_filter_coeff315_a,
rf_filter_coeff315_b,rf_filter_coeff316_a, rf_filter_coeff316_b,rf_filter_coeff317_a,
rf_filter_coeff317_b,rf_filter_coeff318_a, rf_filter_coeff318_b,rf_filter_coeff319_a,
rf_filter_coeff319_b,rf_filter_coeff320_a, rf_filter_coeff320_b,rf_filter_coeff321_a,
rf_filter_coeff321_b,rf_filter_coeff322_a, rf_filter_coeff322_b,rf_filter_coeff323_a,
rf_filter_coeff323_b,rf_filter_coeff324_a, rf_filter_coeff324_b,rf_filter_coeff325_a,
rf_filter_coeff325_b,rf_filter_coeff326_a, rf_filter_coeff326_b,rf_filter_coeff327_a,
rf_filter_coeff327_b,rf_filter_coeff328_a, rf_filter_coeff328_b,rf_filter_coeff329_a,
rf_filter_coeff329_b,rf_filter_coeff330_a, rf_filter_coeff330_b,rf_filter_coeff331_a,
rf_filter_coeff331_b,rf_filter_coeff332_a, rf_filter_coeff332_b,rf_filter_coeff333_a,
rf_filter_coeff333_b,rf_filter_coeff334_a, rf_filter_coeff334_b,rf_filter_coeff335_a,
rf_filter_coeff335_b,rf_filter_coeff336_a, rf_filter_coeff336_b,rf_filter_coeff337_a,
rf_filter_coeff337_b,rf_filter_coeff338_a, rf_filter_coeff338_b,rf_filter_coeff339_a,
rf_filter_coeff339_b,rf_filter_coeff340_a, rf_filter_coeff340_b,rf_filter_coeff341_a,
rf_filter_coeff341_b,rf_filter_coeff342_a, rf_filter_coeff342_b,rf_filter_coeff343_a,
rf_filter_coeff343_b,rf_filter_coeff344_a, rf_filter_coeff344_b,rf_filter_coeff345_a,
rf_filter_coeff345_b,rf_filter_coeff346_a, rf_filter_coeff346_b,rf_filter_coeff347_a,
rf_filter_coeff347_b,rf_filter_coeff348_a, rf_filter_coeff348_b,rf_filter_coeff349_a,
rf_filter_coeff349_b,rf_filter_coeff350_a, rf_filter_coeff350_b,rf_filter_coeff351_a,
rf_filter_coeff351_b,rf_filter_coeff352_a, rf_filter_coeff352_b,rf_filter_coeff353_a,
rf_filter_coeff353_b,rf_filter_coeff354_a, rf_filter_coeff354_b,rf_filter_coeff355_a,
rf_filter_coeff355_b,rf_filter_coeff356_a, rf_filter_coeff356_b,rf_filter_coeff357_a,
rf_filter_coeff357_b,rf_filter_coeff358_a, rf_filter_coeff358_b,rf_filter_coeff359_a,
rf_filter_coeff359_b,rf_filter_coeff360_a, rf_filter_coeff360_b,rf_filter_coeff361_a,
rf_filter_coeff361_b,rf_filter_coeff362_a, rf_filter_coeff362_b,rf_filter_coeff363_a,
rf_filter_coeff363_b,rf_filter_coeff364_a, rf_filter_coeff364_b,rf_filter_coeff365_a,
rf_filter_coeff365_b,rf_filter_coeff366_a, rf_filter_coeff366_b,rf_filter_coeff367_a,
rf_filter_coeff367_b,rf_filter_coeff368_a, rf_filter_coeff368_b,rf_filter_coeff369_a,
rf_filter_coeff369_b,rf_filter_coeff370_a, rf_filter_coeff370_b,rf_filter_coeff371_a,
rf_filter_coeff371_b,rf_filter_coeff372_a, rf_filter_coeff372_b,rf_filter_coeff373_a,
rf_filter_coeff373_b,rf_filter_coeff374_a, rf_filter_coeff374_b,rf_filter_coeff375_a,
rf_filter_coeff375_b,rf_filter_coeff376_a, rf_filter_coeff376_b,rf_filter_coeff377_a,
rf_filter_coeff377_b,rf_filter_coeff378_a, rf_filter_coeff378_b,rf_filter_coeff379_a,
rf_filter_coeff379_b,rf_filter_coeff380_a, rf_filter_coeff380_b,rf_filter_coeff381_a,
rf_filter_coeff381_b,rf_filter_coeff382_a, rf_filter_coeff382_b,rf_filter_coeff383_a,
rf_filter_coeff383_b,rf_filter_coeff384_a, rf_filter_coeff384_b,rf_filter_coeff385_a,
rf_filter_coeff385_b,rf_filter_coeff386_a, rf_filter_coeff386_b,rf_filter_coeff387_a,

```
rf_filter_coeff387_b,rf_filter_coeff388_a, rf_filter_coeff388_b,rf_filter_coeff389_a,
rf_filter_coeff389_b,rf_filter_coeff390_a, rf_filter_coeff390_b,rf_filter_coeff391_a,
rf_filter_coeff391_b,rf_filter_coeff392_a, rf_filter_coeff392_b,rf_filter_coeff393_a,
rf_filter_coeff393_b,rf_filter_coeff394_a, rf_filter_coeff394_b,rf_filter_coeff395_a,
rf_filter_coeff395_b,rf_filter_coeff396_a, rf_filter_coeff396_b,rf_filter_coeff397_a,
rf_filter_coeff397_b,rf_filter_coeff398_a, rf_filter_coeff398_b,rf_filter_coeff399_a,
rf_filter_coeff399_b,rf_filter_coeff400_a, rf_filter_coeff400_b,rf_filter_coeff401_a,
rf_filter_coeff401_b,rf_filter_coeff402_a, rf_filter_coeff402_b,rf_filter_coeff403_a,
rf_filter_coeff403_b,rf_filter_coeff404_a, rf_filter_coeff404_b,rf_filter_coeff405_a,
rf_filter_coeff405_b,rf_filter_coeff406_a, rf_filter_coeff406_b,rf_filter_coeff407_a,
rf_filter_coeff407_b,rf_filter_coeff408_a, rf_filter_coeff408_b,rf_filter_coeff409_a,
rf_filter_coeff409_b,rf_filter_coeff410_a, rf_filter_coeff410_b,rf_filter_coeff411_a,
rf_filter_coeff411_b,rf_filter_coeff412_a, rf_filter_coeff412_b,rf_filter_coeff413_a,
rf_filter_coeff413_b,rf_filter_coeff414_a, rf_filter_coeff414_b,rf_filter_coeff415_a,
rf_filter_coeff415_b,rf_filter_coeff416_a, rf_filter_coeff416_b,rf_filter_coeff417_a,
rf_filter_coeff417_b,rf_filter_coeff418_a, rf_filter_coeff418_b,rf_filter_coeff419_a,
rf_filter_coeff419_b,rf_filter_coeff420_a, rf_filter_coeff420_b,rf_filter_coeff421_a,
rf_filter_coeff421_b,rf_filter_coeff422_a, rf_filter_coeff422_b,rf_filter_coeff423_a,
rf_filter_coeff423_b,rf_filter_coeff424_a, rf_filter_coeff424_b,rf_filter_coeff425_a,
rf_filter_coeff425_b,rf_filter_coeff426_a, rf_filter_coeff426_b,rf_filter_coeff427_a,
rf_filter_coeff427_b,rf_filter_coeff428_a, rf_filter_coeff428_b,rf_filter_coeff429_a,
rf_filter_coeff429_b,rf_filter_coeff430_a, rf_filter_coeff430_b,rf_filter_coeff431_a,
rf_filter_coeff431_b,rf_filter_coeff432_a, rf_filter_coeff432_b,rf_filter_coeff433_a,
rf_filter_coeff433_b,rf_filter_coeff434_a, rf_filter_coeff434_b,rf_filter_coeff435_a,
rf_filter_coeff435_b,rf_filter_coeff436_a, rf_filter_coeff436_b,rf_filter_coeff437_a,
rf_filter_coeff437_b,rf_filter_coeff438_a, rf_filter_coeff438_b,rf_filter_coeff439_a,
rf_filter_coeff439_b,rf_filter_coeff440_a, rf_filter_coeff440_b,rf_filter_coeff441_a,
rf_filter_coeff441_b,rf_filter_coeff442_a, rf_filter_coeff442_b,rf_filter_coeff443_a,
rf_filter_coeff443_b,rf_filter_coeff444_a, rf_filter_coeff444_b,rf_filter_coeff445_a,
rf_filter_coeff445_b,rf_filter_coeff446_a, rf_filter_coeff446_b,rf_filter_coeff447_a,
rf_filter_coeff447_b,rf_filter_coeff448_a, rf_filter_coeff448_b,rf_filter_coeff449_a,
rf_filter_coeff449_b,rf_filter_coeff450_a, rf_filter_coeff450_b,rf_filter_coeff451_a,
rf_filter_coeff451_b,rf_filter_coeff452_a, rf_filter_coeff452_b,rf_filter_coeff453_a,
rf_filter_coeff453_b,rf_filter_coeff454_a, rf_filter_coeff454_b,rf_filter_coeff455_a,
rf_filter_coeff455_b,rf_filter_coeff456_a, rf_filter_coeff456_b,rf_filter_coeff457_a,
rf_filter_coeff457_b,rf_filter_coeff458_a, rf_filter_coeff458_b,rf_filter_coeff459_a,
rf_filter_coeff459_b,rf_filter_coeff460_a, rf_filter_coeff460_b,rf_filter_coeff461_a,
rf_filter_coeff461_b,rf_filter_coeff462_a, rf_filter_coeff462_b,rf_filter_coeff463_a,
rf_filter_coeff463_b,rf_filter_coeff464_a, rf_filter_coeff464_b,rf_filter_coeff465_a,
rf_filter_coeff465_b,rf_filter_coeff466_a, rf_filter_coeff466_b,rf_filter_coeff467_a,
rf_filter_coeff467_b,rf_filter_coeff468_a, rf_filter_coeff468_b,rf_filter_coeff469_a,
rf_filter_coeff469_b,rf_filter_coeff470_a, rf_filter_coeff470_b,rf_filter_coeff471_a,
rf_filter_coeff471_b,rf_filter_coeff472_a, rf_filter_coeff472_b,rf_filter_coeff473_a,
rf_filter_coeff473_b,rf_filter_coeff474_a, rf_filter_coeff474_b,rf_filter_coeff475_a,
rf_filter_coeff475_b,rf_filter_coeff476_a, rf_filter_coeff476_b,rf_filter_coeff477_a,
rf_filter_coeff477_b,rf_filter_coeff478_a, rf_filter_coeff478_b,rf_filter_coeff479_a,
rf_filter_coeff479_b,rf_filter_coeff480_a, rf_filter_coeff480_b,rf_filter_coeff481_a,
rf_filter_coeff481_b,rf_filter_coeff482_a, rf_filter_coeff482_b,rf_filter_coeff483_a,
rf_filter_coeff483_b,rf_filter_coeff484_a, rf_filter_coeff484_b,rf_filter_coeff485_a,
rf_filter_coeff485_b,rf_filter_coeff486_a, rf_filter_coeff486_b,rf_filter_coeff487_a,
rf_filter_coeff487_b,rf_filter_coeff488_a, rf_filter_coeff488_b,rf_filter_coeff489_a,
rf_filter_coeff489_b,rf_filter_coeff490_a, rf_filter_coeff490_b,rf_filter_coeff491_a,
rf_filter_coeff491_b,rf_filter_coeff492_a, rf_filter_coeff492_b,rf_filter_coeff493_a,
rf_filter_coeff493_b,rf_filter_coeff494_a, rf_filter_coeff494_b,rf_filter_coeff495_a,
rf_filter_coeff495_b,rf_filter_coeff496_a, rf_filter_coeff496_b,rf_filter_coeff497_a,
rf_filter_coeff497_b,rf_filter_coeff498_a, rf_filter_coeff498_b,rf_filter_coeff499_a,
rf_filter_coeff499_b,rf_filter_coeff500_a, rf_filter_coeff500_b,rf_filter_coeff501_a,
rf_filter_coeff501_b,rf_filter_coeff502_a, rf_filter_coeff502_b,rf_filter_coeff503_a,
rf_filter_coeff503_b,rf_filter_coeff504_a, rf_filter_coeff504_b,rf_filter_coeff505_a,
rf_filter_coeff505_b,rf_filter_coeff506_a, rf_filter_coeff506_b,rf_filter_coeff507_a,
rf_filter_coeff507_b,rf_filter_coeff508_a, rf_filter_coeff508_b,rf_filter_coeff509_a,
rf_filter_coeff509_b,rf_filter_coeff510_a, rf_filter_coeff510_b,rf_filter_coeff511_a,
rf_filter_coeff511_b,
                trig_fifo_overrun,trig_fifo_underrun,trig_filter_ovf_flag_clear);

    // REGISTER INTERFACES
    //-----------------------------------------------------------------------------
    //-----------------------------------------------------------------------------
    // Inputs (General)
    input                       clk;                                        // master clock
    input                       rst_n;                                      // reset
```

```
     input          [10:0]      i2c_addr;                                   // register address
     input          [ 7:0]      i2c_wdata;                                  // data to be written
for a write op
     input                      i2c_xfc_write;                              // write data
transfer complete
     input                      i2c_op;                                     // 1- write, 0- read

     // Inputs (RO Signals)
     //input ro_chip_id;                                                    // fixed chip ID
     //input ro_revision_id;                                                // fixed revision ID
     input                      ro_fifo_overrun;                            // I2S input audio
FIFO overrun
     input                      ro_fifo_underrun;                           // I2S output audio
FIFO underrun
     input                      ro_filter_ovf_flag;                         // filter overflow
flag

     // Outputs (General)
     output wire    [ 7:0]      i2c_rdata;                                  // read return data
     output wire                i2c_xfc_read;                               // read data transfer
complete

     // Outputs (RF Signals)
     output wire                rf_i2si_bist_en;                            // 0- deserializer
signal 1- BIST signal
     output wire    [ 2:0]      rf_filter_shift;                            // number of bit
postions to shift after filter accumulator
     output wire                rf_filter_clip_en;                          // 0- no clipping 1-
performs clipping
     output wire    [ 7:0]      rf_i2si_bist_start_val_a;                   // start value of
BIST signal
     output wire    [ 3:0]      rf_i2si_bist_start_val_b;                   // start value of
BIST signal
     output wire    [ 7:0]      rf_i2si_bist_incr;                          // BIST signal
increment value
     output wire    [ 7:0]      rf_i2si_bist_upper_limit_a;                 // upper limit of
BIST signal
     output wire    [ 3:0]      rf_i2si_bist_upper_limit_b;                 // upper limit of
BIST signal
     output wire                rf_i2si_en;                                 // enable bit for I2S
input deserializer
     output wire    [ 7:0]      rf_filter_coeff0_a, rf_filter_coeff0_b,rf_filter_coeff1_a,
rf_filter_coeff1_b,rf_filter_coeff2_a, rf_filter_coeff2_b,rf_filter_coeff3_a,
rf_filter_coeff3_b,rf_filter_coeff4_a, rf_filter_coeff4_b,rf_filter_coeff5_a,
rf_filter_coeff5_b,rf_filter_coeff6_a, rf_filter_coeff6_b,rf_filter_coeff7_a,
rf_filter_coeff7_b,rf_filter_coeff8_a, rf_filter_coeff8_b,rf_filter_coeff9_a,
rf_filter_coeff9_b,rf_filter_coeff10_a, rf_filter_coeff10_b,rf_filter_coeff11_a,
rf_filter_coeff11_b,rf_filter_coeff12_a, rf_filter_coeff12_b,rf_filter_coeff13_a,
rf_filter_coeff13_b,rf_filter_coeff14_a, rf_filter_coeff14_b,rf_filter_coeff15_a,
rf_filter_coeff15_b,rf_filter_coeff16_a, rf_filter_coeff16_b,rf_filter_coeff17_a,
rf_filter_coeff17_b,rf_filter_coeff18_a, rf_filter_coeff18_b,rf_filter_coeff19_a,
rf_filter_coeff19_b,rf_filter_coeff20_a, rf_filter_coeff20_b,rf_filter_coeff21_a,
rf_filter_coeff21_b,rf_filter_coeff22_a, rf_filter_coeff22_b,rf_filter_coeff23_a,
rf_filter_coeff23_b,rf_filter_coeff24_a, rf_filter_coeff24_b,rf_filter_coeff25_a,
rf_filter_coeff25_b,rf_filter_coeff26_a, rf_filter_coeff26_b,rf_filter_coeff27_a,
rf_filter_coeff27_b,rf_filter_coeff28_a, rf_filter_coeff28_b,rf_filter_coeff29_a,
rf_filter_coeff29_b,rf_filter_coeff30_a, rf_filter_coeff30_b,rf_filter_coeff31_a,
rf_filter_coeff31_b,rf_filter_coeff32_a, rf_filter_coeff32_b,rf_filter_coeff33_a,
rf_filter_coeff33_b,rf_filter_coeff34_a, rf_filter_coeff34_b,rf_filter_coeff35_a,
rf_filter_coeff35_b,rf_filter_coeff36_a, rf_filter_coeff36_b,rf_filter_coeff37_a,
rf_filter_coeff37_b,rf_filter_coeff38_a, rf_filter_coeff38_b,rf_filter_coeff39_a,
rf_filter_coeff39_b,rf_filter_coeff40_a, rf_filter_coeff40_b,rf_filter_coeff41_a,
rf_filter_coeff41_b,rf_filter_coeff42_a, rf_filter_coeff42_b,rf_filter_coeff43_a,
rf_filter_coeff43_b,rf_filter_coeff44_a, rf_filter_coeff44_b,rf_filter_coeff45_a,
rf_filter_coeff45_b,rf_filter_coeff46_a, rf_filter_coeff46_b,rf_filter_coeff47_a,
rf_filter_coeff47_b,rf_filter_coeff48_a, rf_filter_coeff48_b,rf_filter_coeff49_a,
rf_filter_coeff49_b,rf_filter_coeff50_a, rf_filter_coeff50_b,rf_filter_coeff51_a,
rf_filter_coeff51_b,rf_filter_coeff52_a, rf_filter_coeff52_b,rf_filter_coeff53_a,
rf_filter_coeff53_b,rf_filter_coeff54_a, rf_filter_coeff54_b,rf_filter_coeff55_a,
rf_filter_coeff55_b,rf_filter_coeff56_a, rf_filter_coeff56_b,rf_filter_coeff57_a,
rf_filter_coeff57_b,rf_filter_coeff58_a, rf_filter_coeff58_b,rf_filter_coeff59_a,
```

rf_filter_coeff59_b,rf_filter_coeff60_a, rf_filter_coeff60_b,rf_filter_coeff61_a,
rf_filter_coeff61_b,rf_filter_coeff62_a, rf_filter_coeff62_b,rf_filter_coeff63_a,
rf_filter_coeff63_b,rf_filter_coeff64_a, rf_filter_coeff64_b,rf_filter_coeff65_a,
rf_filter_coeff65_b,rf_filter_coeff66_a, rf_filter_coeff66_b,rf_filter_coeff67_a,
rf_filter_coeff67_b,rf_filter_coeff68_a, rf_filter_coeff68_b,rf_filter_coeff69_a,
rf_filter_coeff69_b,rf_filter_coeff70_a, rf_filter_coeff70_b,rf_filter_coeff71_a,
rf_filter_coeff71_b,rf_filter_coeff72_a, rf_filter_coeff72_b,rf_filter_coeff73_a,
rf_filter_coeff73_b,rf_filter_coeff74_a, rf_filter_coeff74_b,rf_filter_coeff75_a,
rf_filter_coeff75_b,rf_filter_coeff76_a, rf_filter_coeff76_b,rf_filter_coeff77_a,
rf_filter_coeff77_b,rf_filter_coeff78_a, rf_filter_coeff78_b,rf_filter_coeff79_a,
rf_filter_coeff79_b,rf_filter_coeff80_a, rf_filter_coeff80_b,rf_filter_coeff81_a,
rf_filter_coeff81_b,rf_filter_coeff82_a, rf_filter_coeff82_b,rf_filter_coeff83_a,
rf_filter_coeff83_b,rf_filter_coeff84_a, rf_filter_coeff84_b,rf_filter_coeff85_a,
rf_filter_coeff85_b,rf_filter_coeff86_a, rf_filter_coeff86_b,rf_filter_coeff87_a,
rf_filter_coeff87_b,rf_filter_coeff88_a, rf_filter_coeff88_b,rf_filter_coeff89_a,
rf_filter_coeff89_b,rf_filter_coeff90_a, rf_filter_coeff90_b,rf_filter_coeff91_a,
rf_filter_coeff91_b,rf_filter_coeff92_a, rf_filter_coeff92_b,rf_filter_coeff93_a,
rf_filter_coeff93_b,rf_filter_coeff94_a, rf_filter_coeff94_b,rf_filter_coeff95_a,
rf_filter_coeff95_b,rf_filter_coeff96_a, rf_filter_coeff96_b,rf_filter_coeff97_a,
rf_filter_coeff97_b,rf_filter_coeff98_a, rf_filter_coeff98_b,rf_filter_coeff99_a,
rf_filter_coeff99_b,rf_filter_coeff100_a, rf_filter_coeff100_b,rf_filter_coeff101_a,
rf_filter_coeff101_b,rf_filter_coeff102_a, rf_filter_coeff102_b,rf_filter_coeff103_a,
rf_filter_coeff103_b,rf_filter_coeff104_a, rf_filter_coeff104_b,rf_filter_coeff105_a,
rf_filter_coeff105_b,rf_filter_coeff106_a, rf_filter_coeff106_b,rf_filter_coeff107_a,
rf_filter_coeff107_b,rf_filter_coeff108_a, rf_filter_coeff108_b,rf_filter_coeff109_a,
rf_filter_coeff109_b,rf_filter_coeff110_a, rf_filter_coeff110_b,rf_filter_coeff111_a,
rf_filter_coeff111_b,rf_filter_coeff112_a, rf_filter_coeff112_b,rf_filter_coeff113_a,
rf_filter_coeff113_b,rf_filter_coeff114_a, rf_filter_coeff114_b,rf_filter_coeff115_a,
rf_filter_coeff115_b,rf_filter_coeff116_a, rf_filter_coeff116_b,rf_filter_coeff117_a,
rf_filter_coeff117_b,rf_filter_coeff118_a, rf_filter_coeff118_b,rf_filter_coeff119_a,
rf_filter_coeff119_b,rf_filter_coeff120_a, rf_filter_coeff120_b,rf_filter_coeff121_a,
rf_filter_coeff121_b,rf_filter_coeff122_a, rf_filter_coeff122_b,rf_filter_coeff123_a,
rf_filter_coeff123_b,rf_filter_coeff124_a, rf_filter_coeff124_b,rf_filter_coeff125_a,
rf_filter_coeff125_b,rf_filter_coeff126_a, rf_filter_coeff126_b,rf_filter_coeff127_a,
rf_filter_coeff127_b,rf_filter_coeff128_a, rf_filter_coeff128_b,rf_filter_coeff129_a,
rf_filter_coeff129_b,rf_filter_coeff130_a, rf_filter_coeff130_b,rf_filter_coeff131_a,
rf_filter_coeff131_b,rf_filter_coeff132_a, rf_filter_coeff132_b,rf_filter_coeff133_a,
rf_filter_coeff133_b,rf_filter_coeff134_a, rf_filter_coeff134_b,rf_filter_coeff135_a,
rf_filter_coeff135_b,rf_filter_coeff136_a, rf_filter_coeff136_b,rf_filter_coeff137_a,
rf_filter_coeff137_b,rf_filter_coeff138_a, rf_filter_coeff138_b,rf_filter_coeff139_a,
rf_filter_coeff139_b,rf_filter_coeff140_a, rf_filter_coeff140_b,rf_filter_coeff141_a,
rf_filter_coeff141_b,rf_filter_coeff142_a, rf_filter_coeff142_b,rf_filter_coeff143_a,
rf_filter_coeff143_b,rf_filter_coeff144_a, rf_filter_coeff144_b,rf_filter_coeff145_a,
rf_filter_coeff145_b,rf_filter_coeff146_a, rf_filter_coeff146_b,rf_filter_coeff147_a,
rf_filter_coeff147_b,rf_filter_coeff148_a, rf_filter_coeff148_b,rf_filter_coeff149_a,
rf_filter_coeff149_b,rf_filter_coeff150_a, rf_filter_coeff150_b,rf_filter_coeff151_a,
rf_filter_coeff151_b,rf_filter_coeff152_a, rf_filter_coeff152_b,rf_filter_coeff153_a,
rf_filter_coeff153_b,rf_filter_coeff154_a, rf_filter_coeff154_b,rf_filter_coeff155_a,
rf_filter_coeff155_b,rf_filter_coeff156_a, rf_filter_coeff156_b,rf_filter_coeff157_a,
rf_filter_coeff157_b,rf_filter_coeff158_a, rf_filter_coeff158_b,rf_filter_coeff159_a,
rf_filter_coeff159_b,rf_filter_coeff160_a, rf_filter_coeff160_b,rf_filter_coeff161_a,
rf_filter_coeff161_b,rf_filter_coeff162_a, rf_filter_coeff162_b,rf_filter_coeff163_a,
rf_filter_coeff163_b,rf_filter_coeff164_a, rf_filter_coeff164_b,rf_filter_coeff165_a,
rf_filter_coeff165_b,rf_filter_coeff166_a, rf_filter_coeff166_b,rf_filter_coeff167_a,
rf_filter_coeff167_b,rf_filter_coeff168_a, rf_filter_coeff168_b,rf_filter_coeff169_a,
rf_filter_coeff169_b,rf_filter_coeff170_a, rf_filter_coeff170_b,rf_filter_coeff171_a,
rf_filter_coeff171_b,rf_filter_coeff172_a, rf_filter_coeff172_b,rf_filter_coeff173_a,
rf_filter_coeff173_b,rf_filter_coeff174_a, rf_filter_coeff174_b,rf_filter_coeff175_a,
rf_filter_coeff175_b,rf_filter_coeff176_a, rf_filter_coeff176_b,rf_filter_coeff177_a,
rf_filter_coeff177_b,rf_filter_coeff178_a, rf_filter_coeff178_b,rf_filter_coeff179_a,
rf_filter_coeff179_b,rf_filter_coeff180_a, rf_filter_coeff180_b,rf_filter_coeff181_a,
rf_filter_coeff181_b,rf_filter_coeff182_a, rf_filter_coeff182_b,rf_filter_coeff183_a,
rf_filter_coeff183_b,rf_filter_coeff184_a, rf_filter_coeff184_b,rf_filter_coeff185_a,
rf_filter_coeff185_b,rf_filter_coeff186_a, rf_filter_coeff186_b,rf_filter_coeff187_a,
rf_filter_coeff187_b,rf_filter_coeff188_a, rf_filter_coeff188_b,rf_filter_coeff189_a,
rf_filter_coeff189_b,rf_filter_coeff190_a, rf_filter_coeff190_b,rf_filter_coeff191_a,
rf_filter_coeff191_b,rf_filter_coeff192_a, rf_filter_coeff192_b,rf_filter_coeff193_a,
rf_filter_coeff193_b,rf_filter_coeff194_a, rf_filter_coeff194_b,rf_filter_coeff195_a,
rf_filter_coeff195_b,rf_filter_coeff196_a, rf_filter_coeff196_b,rf_filter_coeff197_a,
rf_filter_coeff197_b,rf_filter_coeff198_a, rf_filter_coeff198_b,rf_filter_coeff199_a,
rf_filter_coeff199_b,rf_filter_coeff200_a, rf_filter_coeff200_b,rf_filter_coeff201_a,

rf_filter_coeff201_b,rf_filter_coeff202_a, rf_filter_coeff202_b,rf_filter_coeff203_a,
rf_filter_coeff203_b,rf_filter_coeff204_a, rf_filter_coeff204_b,rf_filter_coeff205_a,
rf_filter_coeff205_b,rf_filter_coeff206_a, rf_filter_coeff206_b,rf_filter_coeff207_a,
rf_filter_coeff207_b,rf_filter_coeff208_a, rf_filter_coeff208_b,rf_filter_coeff209_a,
rf_filter_coeff209_b,rf_filter_coeff210_a, rf_filter_coeff210_b,rf_filter_coeff211_a,
rf_filter_coeff211_b,rf_filter_coeff212_a, rf_filter_coeff212_b,rf_filter_coeff213_a,
rf_filter_coeff213_b,rf_filter_coeff214_a, rf_filter_coeff214_b,rf_filter_coeff215_a,
rf_filter_coeff215_b,rf_filter_coeff216_a, rf_filter_coeff216_b,rf_filter_coeff217_a,
rf_filter_coeff217_b,rf_filter_coeff218_a, rf_filter_coeff218_b,rf_filter_coeff219_a,
rf_filter_coeff219_b,rf_filter_coeff220_a, rf_filter_coeff220_b,rf_filter_coeff221_a,
rf_filter_coeff221_b,rf_filter_coeff222_a, rf_filter_coeff222_b,rf_filter_coeff223_a,
rf_filter_coeff223_b,rf_filter_coeff224_a, rf_filter_coeff224_b,rf_filter_coeff225_a,
rf_filter_coeff225_b,rf_filter_coeff226_a, rf_filter_coeff226_b,rf_filter_coeff227_a,
rf_filter_coeff227_b,rf_filter_coeff228_a, rf_filter_coeff228_b,rf_filter_coeff229_a,
rf_filter_coeff229_b,rf_filter_coeff230_a, rf_filter_coeff230_b,rf_filter_coeff231_a,
rf_filter_coeff231_b,rf_filter_coeff232_a, rf_filter_coeff232_b,rf_filter_coeff233_a,
rf_filter_coeff233_b,rf_filter_coeff234_a, rf_filter_coeff234_b,rf_filter_coeff235_a,
rf_filter_coeff235_b,rf_filter_coeff236_a, rf_filter_coeff236_b,rf_filter_coeff237_a,
rf_filter_coeff237_b,rf_filter_coeff238_a, rf_filter_coeff238_b,rf_filter_coeff239_a,
rf_filter_coeff239_b,rf_filter_coeff240_a, rf_filter_coeff240_b,rf_filter_coeff241_a,
rf_filter_coeff241_b,rf_filter_coeff242_a, rf_filter_coeff242_b,rf_filter_coeff243_a,
rf_filter_coeff243_b,rf_filter_coeff244_a, rf_filter_coeff244_b,rf_filter_coeff245_a,
rf_filter_coeff245_b,rf_filter_coeff246_a, rf_filter_coeff246_b,rf_filter_coeff247_a,
rf_filter_coeff247_b,rf_filter_coeff248_a, rf_filter_coeff248_b,rf_filter_coeff249_a,
rf_filter_coeff249_b,rf_filter_coeff250_a, rf_filter_coeff250_b,rf_filter_coeff251_a,
rf_filter_coeff251_b,rf_filter_coeff252_a, rf_filter_coeff252_b,rf_filter_coeff253_a,
rf_filter_coeff253_b,rf_filter_coeff254_a, rf_filter_coeff254_b,rf_filter_coeff255_a,
rf_filter_coeff255_b,rf_filter_coeff256_a, rf_filter_coeff256_b,rf_filter_coeff257_a,
rf_filter_coeff257_b,rf_filter_coeff258_a, rf_filter_coeff258_b,rf_filter_coeff259_a,
rf_filter_coeff259_b,rf_filter_coeff260_a, rf_filter_coeff260_b,rf_filter_coeff261_a,
rf_filter_coeff261_b,rf_filter_coeff262_a, rf_filter_coeff262_b,rf_filter_coeff263_a,
rf_filter_coeff263_b,rf_filter_coeff264_a, rf_filter_coeff264_b,rf_filter_coeff265_a,
rf_filter_coeff265_b,rf_filter_coeff266_a, rf_filter_coeff266_b,rf_filter_coeff267_a,
rf_filter_coeff267_b,rf_filter_coeff268_a, rf_filter_coeff268_b,rf_filter_coeff269_a,
rf_filter_coeff269_b,rf_filter_coeff270_a, rf_filter_coeff270_b,rf_filter_coeff271_a,
rf_filter_coeff271_b,rf_filter_coeff272_a, rf_filter_coeff272_b,rf_filter_coeff273_a,
rf_filter_coeff273_b,rf_filter_coeff274_a, rf_filter_coeff274_b,rf_filter_coeff275_a,
rf_filter_coeff275_b,rf_filter_coeff276_a, rf_filter_coeff276_b,rf_filter_coeff277_a,
rf_filter_coeff277_b,rf_filter_coeff278_a, rf_filter_coeff278_b,rf_filter_coeff279_a,
rf_filter_coeff279_b,rf_filter_coeff280_a, rf_filter_coeff280_b,rf_filter_coeff281_a,
rf_filter_coeff281_b,rf_filter_coeff282_a, rf_filter_coeff282_b,rf_filter_coeff283_a,
rf_filter_coeff283_b,rf_filter_coeff284_a, rf_filter_coeff284_b,rf_filter_coeff285_a,
rf_filter_coeff285_b,rf_filter_coeff286_a, rf_filter_coeff286_b,rf_filter_coeff287_a,
rf_filter_coeff287_b,rf_filter_coeff288_a, rf_filter_coeff288_b,rf_filter_coeff289_a,
rf_filter_coeff289_b,rf_filter_coeff290_a, rf_filter_coeff290_b,rf_filter_coeff291_a,
rf_filter_coeff291_b,rf_filter_coeff292_a, rf_filter_coeff292_b,rf_filter_coeff293_a,
rf_filter_coeff293_b,rf_filter_coeff294_a, rf_filter_coeff294_b,rf_filter_coeff295_a,
rf_filter_coeff295_b,rf_filter_coeff296_a, rf_filter_coeff296_b,rf_filter_coeff297_a,
rf_filter_coeff297_b,rf_filter_coeff298_a, rf_filter_coeff298_b,rf_filter_coeff299_a,
rf_filter_coeff299_b,rf_filter_coeff300_a, rf_filter_coeff300_b,rf_filter_coeff301_a,
rf_filter_coeff301_b,rf_filter_coeff302_a, rf_filter_coeff302_b,rf_filter_coeff303_a,
rf_filter_coeff303_b,rf_filter_coeff304_a, rf_filter_coeff304_b,rf_filter_coeff305_a,
rf_filter_coeff305_b,rf_filter_coeff306_a, rf_filter_coeff306_b,rf_filter_coeff307_a,
rf_filter_coeff307_b,rf_filter_coeff308_a, rf_filter_coeff308_b,rf_filter_coeff309_a,
rf_filter_coeff309_b,rf_filter_coeff310_a, rf_filter_coeff310_b,rf_filter_coeff311_a,
rf_filter_coeff311_b,rf_filter_coeff312_a, rf_filter_coeff312_b,rf_filter_coeff313_a,
rf_filter_coeff313_b,rf_filter_coeff314_a, rf_filter_coeff314_b,rf_filter_coeff315_a,
rf_filter_coeff315_b,rf_filter_coeff316_a, rf_filter_coeff316_b,rf_filter_coeff317_a,
rf_filter_coeff317_b,rf_filter_coeff318_a, rf_filter_coeff318_b,rf_filter_coeff319_a,
rf_filter_coeff319_b,rf_filter_coeff320_a, rf_filter_coeff320_b,rf_filter_coeff321_a,
rf_filter_coeff321_b,rf_filter_coeff322_a, rf_filter_coeff322_b,rf_filter_coeff323_a,
rf_filter_coeff323_b,rf_filter_coeff324_a, rf_filter_coeff324_b,rf_filter_coeff325_a,
rf_filter_coeff325_b,rf_filter_coeff326_a, rf_filter_coeff326_b,rf_filter_coeff327_a,
rf_filter_coeff327_b,rf_filter_coeff328_a, rf_filter_coeff328_b,rf_filter_coeff329_a,
rf_filter_coeff329_b,rf_filter_coeff330_a, rf_filter_coeff330_b,rf_filter_coeff331_a,
rf_filter_coeff331_b,rf_filter_coeff332_a, rf_filter_coeff332_b,rf_filter_coeff333_a,
rf_filter_coeff333_b,rf_filter_coeff334_a, rf_filter_coeff334_b,rf_filter_coeff335_a,
rf_filter_coeff335_b,rf_filter_coeff336_a, rf_filter_coeff336_b,rf_filter_coeff337_a,
rf_filter_coeff337_b,rf_filter_coeff338_a, rf_filter_coeff338_b,rf_filter_coeff339_a,
rf_filter_coeff339_b,rf_filter_coeff340_a, rf_filter_coeff340_b,rf_filter_coeff341_a,
rf_filter_coeff341_b,rf_filter_coeff342_a, rf_filter_coeff342_b,rf_filter_coeff343_a,

rf_filter_coeff343_b,rf_filter_coeff344_a, rf_filter_coeff344_b,rf_filter_coeff345_a,
rf_filter_coeff345_b,rf_filter_coeff346_a, rf_filter_coeff346_b,rf_filter_coeff347_a,
rf_filter_coeff347_b,rf_filter_coeff348_a, rf_filter_coeff348_b,rf_filter_coeff349_a,
rf_filter_coeff349_b,rf_filter_coeff350_a, rf_filter_coeff350_b,rf_filter_coeff351_a,
rf_filter_coeff351_b,rf_filter_coeff352_a, rf_filter_coeff352_b,rf_filter_coeff353_a,
rf_filter_coeff353_b,rf_filter_coeff354_a, rf_filter_coeff354_b,rf_filter_coeff355_a,
rf_filter_coeff355_b,rf_filter_coeff356_a, rf_filter_coeff356_b,rf_filter_coeff357_a,
rf_filter_coeff357_b,rf_filter_coeff358_a, rf_filter_coeff358_b,rf_filter_coeff359_a,
rf_filter_coeff359_b,rf_filter_coeff360_a, rf_filter_coeff360_b,rf_filter_coeff361_a,
rf_filter_coeff361_b,rf_filter_coeff362_a, rf_filter_coeff362_b,rf_filter_coeff363_a,
rf_filter_coeff363_b,rf_filter_coeff364_a, rf_filter_coeff364_b,rf_filter_coeff365_a,
rf_filter_coeff365_b,rf_filter_coeff366_a, rf_filter_coeff366_b,rf_filter_coeff367_a,
rf_filter_coeff367_b,rf_filter_coeff368_a, rf_filter_coeff368_b,rf_filter_coeff369_a,
rf_filter_coeff369_b,rf_filter_coeff370_a, rf_filter_coeff370_b,rf_filter_coeff371_a,
rf_filter_coeff371_b,rf_filter_coeff372_a, rf_filter_coeff372_b,rf_filter_coeff373_a,
rf_filter_coeff373_b,rf_filter_coeff374_a, rf_filter_coeff374_b,rf_filter_coeff375_a,
rf_filter_coeff375_b,rf_filter_coeff376_a, rf_filter_coeff376_b,rf_filter_coeff377_a,
rf_filter_coeff377_b,rf_filter_coeff378_a, rf_filter_coeff378_b,rf_filter_coeff379_a,
rf_filter_coeff379_b,rf_filter_coeff380_a, rf_filter_coeff380_b,rf_filter_coeff381_a,
rf_filter_coeff381_b,rf_filter_coeff382_a, rf_filter_coeff382_b,rf_filter_coeff383_a,
rf_filter_coeff383_b,rf_filter_coeff384_a, rf_filter_coeff384_b,rf_filter_coeff385_a,
rf_filter_coeff385_b,rf_filter_coeff386_a, rf_filter_coeff386_b,rf_filter_coeff387_a,
rf_filter_coeff387_b,rf_filter_coeff388_a, rf_filter_coeff388_b,rf_filter_coeff389_a,
rf_filter_coeff389_b,rf_filter_coeff390_a, rf_filter_coeff390_b,rf_filter_coeff391_a,
rf_filter_coeff391_b,rf_filter_coeff392_a, rf_filter_coeff392_b,rf_filter_coeff393_a,
rf_filter_coeff393_b,rf_filter_coeff394_a, rf_filter_coeff394_b,rf_filter_coeff395_a,
rf_filter_coeff395_b,rf_filter_coeff396_a, rf_filter_coeff396_b,rf_filter_coeff397_a,
rf_filter_coeff397_b,rf_filter_coeff398_a, rf_filter_coeff398_b,rf_filter_coeff399_a,
rf_filter_coeff399_b,rf_filter_coeff400_a, rf_filter_coeff400_b,rf_filter_coeff401_a,
rf_filter_coeff401_b,rf_filter_coeff402_a, rf_filter_coeff402_b,rf_filter_coeff403_a,
rf_filter_coeff403_b,rf_filter_coeff404_a, rf_filter_coeff404_b,rf_filter_coeff405_a,
rf_filter_coeff405_b,rf_filter_coeff406_a, rf_filter_coeff406_b,rf_filter_coeff407_a,
rf_filter_coeff407_b,rf_filter_coeff408_a, rf_filter_coeff408_b,rf_filter_coeff409_a,
rf_filter_coeff409_b,rf_filter_coeff410_a, rf_filter_coeff410_b,rf_filter_coeff411_a,
rf_filter_coeff411_b,rf_filter_coeff412_a, rf_filter_coeff412_b,rf_filter_coeff413_a,
rf_filter_coeff413_b,rf_filter_coeff414_a, rf_filter_coeff414_b,rf_filter_coeff415_a,
rf_filter_coeff415_b,rf_filter_coeff416_a, rf_filter_coeff416_b,rf_filter_coeff417_a,
rf_filter_coeff417_b,rf_filter_coeff418_a, rf_filter_coeff418_b,rf_filter_coeff419_a,
rf_filter_coeff419_b,rf_filter_coeff420_a, rf_filter_coeff420_b,rf_filter_coeff421_a,
rf_filter_coeff421_b,rf_filter_coeff422_a, rf_filter_coeff422_b,rf_filter_coeff423_a,
rf_filter_coeff423_b,rf_filter_coeff424_a, rf_filter_coeff424_b,rf_filter_coeff425_a,
rf_filter_coeff425_b,rf_filter_coeff426_a, rf_filter_coeff426_b,rf_filter_coeff427_a,
rf_filter_coeff427_b,rf_filter_coeff428_a, rf_filter_coeff428_b,rf_filter_coeff429_a,
rf_filter_coeff429_b,rf_filter_coeff430_a, rf_filter_coeff430_b,rf_filter_coeff431_a,
rf_filter_coeff431_b,rf_filter_coeff432_a, rf_filter_coeff432_b,rf_filter_coeff433_a,
rf_filter_coeff433_b,rf_filter_coeff434_a, rf_filter_coeff434_b,rf_filter_coeff435_a,
rf_filter_coeff435_b,rf_filter_coeff436_a, rf_filter_coeff436_b,rf_filter_coeff437_a,
rf_filter_coeff437_b,rf_filter_coeff438_a, rf_filter_coeff438_b,rf_filter_coeff439_a,
rf_filter_coeff439_b,rf_filter_coeff440_a, rf_filter_coeff440_b,rf_filter_coeff441_a,
rf_filter_coeff441_b,rf_filter_coeff442_a, rf_filter_coeff442_b,rf_filter_coeff443_a,
rf_filter_coeff443_b,rf_filter_coeff444_a, rf_filter_coeff444_b,rf_filter_coeff445_a,
rf_filter_coeff445_b,rf_filter_coeff446_a, rf_filter_coeff446_b,rf_filter_coeff447_a,
rf_filter_coeff447_b,rf_filter_coeff448_a, rf_filter_coeff448_b,rf_filter_coeff449_a,
rf_filter_coeff449_b,rf_filter_coeff450_a, rf_filter_coeff450_b,rf_filter_coeff451_a,
rf_filter_coeff451_b,rf_filter_coeff452_a, rf_filter_coeff452_b,rf_filter_coeff453_a,
rf_filter_coeff453_b,rf_filter_coeff454_a, rf_filter_coeff454_b,rf_filter_coeff455_a,
rf_filter_coeff455_b,rf_filter_coeff456_a, rf_filter_coeff456_b,rf_filter_coeff457_a,
rf_filter_coeff457_b,rf_filter_coeff458_a, rf_filter_coeff458_b,rf_filter_coeff459_a,
rf_filter_coeff459_b,rf_filter_coeff460_a, rf_filter_coeff460_b,rf_filter_coeff461_a,
rf_filter_coeff461_b,rf_filter_coeff462_a, rf_filter_coeff462_b,rf_filter_coeff463_a,
rf_filter_coeff463_b,rf_filter_coeff464_a, rf_filter_coeff464_b,rf_filter_coeff465_a,
rf_filter_coeff465_b,rf_filter_coeff466_a, rf_filter_coeff466_b,rf_filter_coeff467_a,
rf_filter_coeff467_b,rf_filter_coeff468_a, rf_filter_coeff468_b,rf_filter_coeff469_a,
rf_filter_coeff469_b,rf_filter_coeff470_a, rf_filter_coeff470_b,rf_filter_coeff471_a,
rf_filter_coeff471_b,rf_filter_coeff472_a, rf_filter_coeff472_b,rf_filter_coeff473_a,
rf_filter_coeff473_b,rf_filter_coeff474_a, rf_filter_coeff474_b,rf_filter_coeff475_a,
rf_filter_coeff475_b,rf_filter_coeff476_a, rf_filter_coeff476_b,rf_filter_coeff477_a,
rf_filter_coeff477_b,rf_filter_coeff478_a, rf_filter_coeff478_b,rf_filter_coeff479_a,
rf_filter_coeff479_b,rf_filter_coeff480_a, rf_filter_coeff480_b,rf_filter_coeff481_a,
rf_filter_coeff481_b,rf_filter_coeff482_a, rf_filter_coeff482_b,rf_filter_coeff483_a,
rf_filter_coeff483_b,rf_filter_coeff484_a, rf_filter_coeff484_b,rf_filter_coeff485_a,

```
rf_filter_coeff485_b,rf_filter_coeff486_a, rf_filter_coeff486_b,rf_filter_coeff487_a,
rf_filter_coeff487_b,rf_filter_coeff488_a, rf_filter_coeff488_b,rf_filter_coeff489_a,
rf_filter_coeff489_b,rf_filter_coeff490_a, rf_filter_coeff490_b,rf_filter_coeff491_a,
rf_filter_coeff491_b,rf_filter_coeff492_a, rf_filter_coeff492_b,rf_filter_coeff493_a,
rf_filter_coeff493_b,rf_filter_coeff494_a, rf_filter_coeff494_b,rf_filter_coeff495_a,
rf_filter_coeff495_b,rf_filter_coeff496_a, rf_filter_coeff496_b,rf_filter_coeff497_a,
rf_filter_coeff497_b,rf_filter_coeff498_a, rf_filter_coeff498_b,rf_filter_coeff499_a,
rf_filter_coeff499_b,rf_filter_coeff500_a, rf_filter_coeff500_b,rf_filter_coeff501_a,
rf_filter_coeff501_b,rf_filter_coeff502_a, rf_filter_coeff502_b,rf_filter_coeff503_a,
rf_filter_coeff503_b,rf_filter_coeff504_a, rf_filter_coeff504_b,rf_filter_coeff505_a,
rf_filter_coeff505_b,rf_filter_coeff506_a, rf_filter_coeff506_b,rf_filter_coeff507_a,
rf_filter_coeff507_b,rf_filter_coeff508_a, rf_filter_coeff508_b,rf_filter_coeff509_a,
rf_filter_coeff509_b,rf_filter_coeff510_a, rf_filter_coeff510_b,rf_filter_coeff511_a,
rf_filter_coeff511_b;

     // Outputs (Triggers)
     output wire trig_fifo_overrun;                                          // signal to reset
I2S input FIFO overrun
     output wire trig_fifo_underrun;                                         // signal to reset
I2S output FIFO underrun
     output wire trig_filter_ovf_flag_clear;                                 // signal to reset
filter overflow flag
     //-----------------------------------------------------------------------------
     //-----------------------------------------------------------------------------

     // SUBMODULE CONNECTIONS
     //-----------------------------------------------------------------------------
     //-----------------------------------------------------------------------------

     //-----------------------------------------------------------------------------
     //-----------------------------------------------------------------------------


     // MODULE INSTANTIATION
     //-----------------------------------------------------------------------------
     //-----------------------------------------------------------------------------


    register Register(
        .rst_n                      (rst_n),                      // input: reset
        .clk                        (clk),                        // input: master clock
        .addr                       (i2c_addr),                   // input: register address
        .wdata                      (i2c_wdata),                  // input: write data
        .w_enable                   (i2c_op),                     // input: write enable
        .wxfc                       (i2c_xfc_write),              // input: write transfer
complete
        .rxfc                       (i2c_xfc_read),               // output: read transfer
complete
        .ro_fifo_underrun           (ro_fifo_underrun),           // input: I2S output audio FIFO
underrun
        .ro_fifo_overrun            (ro_fifo_overrun),            // input: I2S input audio FIFO
overrun
        .ro_filter_ovf_flag         (ro_filter_ovf_flag),         // input: filter overflow
status bit
        .rdata                      (i2c_rdata),                  // output: read return data
        .rf_i2si_bist_en            (rf_i2si_bist_en),            // output: 0- deserializer
signal 1- BIST signal
        .rf_filter_shift            (rf_filter_shift),            // output: number of bit
postions to shift after filter accumulator
        .rf_i2si_en                 (rf_i2si_en),                 // output: Enable bit for
Deserializer. 0 = inactive, 1 = active
        .rf_filter_clip_en          (rf_filter_clip_en),          // output: 0- no clipping 1-
performs clipping
        .rf_i2si_bist_start_val_a   (rf_i2si_bist_start_val_a),   // output: start value of BIST
signal
        .rf_i2si_bist_start_val_b   (rf_i2si_bist_start_val_b),   // output: start value of BIST
signal
        .rf_i2si_bist_incr          (rf_i2si_bist_incr),          // output: BIST signal
increment value
        .rf_i2si_bist_upper_limit_a (rf_i2si_bist_upper_limit_a), // output: upper limit of BIST
signal
```

```
        .rf_i2si_bist_upper_limit_b (rf_i2si_bist_upper_limit_b), // output: upper limit of BIST
signal
        .rf_filter_coeff0_a(rf_filter_coeff0_a),
        .rf_filter_coeff0_b(rf_filter_coeff0_b),
        .rf_filter_coeff1_a(rf_filter_coeff1_a),
        .rf_filter_coeff1_b(rf_filter_coeff1_b),
        .rf_filter_coeff2_a(rf_filter_coeff2_a),
        .rf_filter_coeff2_b(rf_filter_coeff2_b),
        .rf_filter_coeff3_a(rf_filter_coeff3_a),
        .rf_filter_coeff3_b(rf_filter_coeff3_b),
        .rf_filter_coeff4_a(rf_filter_coeff4_a),
        .rf_filter_coeff4_b(rf_filter_coeff4_b),
        .rf_filter_coeff5_a(rf_filter_coeff5_a),
        .rf_filter_coeff5_b(rf_filter_coeff5_b),
        .rf_filter_coeff6_a(rf_filter_coeff6_a),
        .rf_filter_coeff6_b(rf_filter_coeff6_b),
        .rf_filter_coeff7_a(rf_filter_coeff7_a),
        .rf_filter_coeff7_b(rf_filter_coeff7_b),
        .rf_filter_coeff8_a(rf_filter_coeff8_a),
        .rf_filter_coeff8_b(rf_filter_coeff8_b),
        .rf_filter_coeff9_a(rf_filter_coeff9_a),
        .rf_filter_coeff9_b(rf_filter_coeff9_b),
        .rf_filter_coeff10_a(rf_filter_coeff10_a),
        .rf_filter_coeff10_b(rf_filter_coeff10_b),
        .rf_filter_coeff11_a(rf_filter_coeff11_a),
        .rf_filter_coeff11_b(rf_filter_coeff11_b),
        .rf_filter_coeff12_a(rf_filter_coeff12_a),
        .rf_filter_coeff12_b(rf_filter_coeff12_b),
        .rf_filter_coeff13_a(rf_filter_coeff13_a),
        .rf_filter_coeff13_b(rf_filter_coeff13_b),
        .rf_filter_coeff14_a(rf_filter_coeff14_a),
        .rf_filter_coeff14_b(rf_filter_coeff14_b),
        .rf_filter_coeff15_a(rf_filter_coeff15_a),
        .rf_filter_coeff15_b(rf_filter_coeff15_b),
        .rf_filter_coeff16_a(rf_filter_coeff16_a),
        .rf_filter_coeff16_b(rf_filter_coeff16_b),
        .rf_filter_coeff17_a(rf_filter_coeff17_a),
        .rf_filter_coeff17_b(rf_filter_coeff17_b),
        .rf_filter_coeff18_a(rf_filter_coeff18_a),
        .rf_filter_coeff18_b(rf_filter_coeff18_b),
        .rf_filter_coeff19_a(rf_filter_coeff19_a),
        .rf_filter_coeff19_b(rf_filter_coeff19_b),
        .rf_filter_coeff20_a(rf_filter_coeff20_a),
        .rf_filter_coeff20_b(rf_filter_coeff20_b),
        .rf_filter_coeff21_a(rf_filter_coeff21_a),
        .rf_filter_coeff21_b(rf_filter_coeff21_b),
        .rf_filter_coeff22_a(rf_filter_coeff22_a),
        .rf_filter_coeff22_b(rf_filter_coeff22_b),
        .rf_filter_coeff23_a(rf_filter_coeff23_a),
        .rf_filter_coeff23_b(rf_filter_coeff23_b),
        .rf_filter_coeff24_a(rf_filter_coeff24_a),
        .rf_filter_coeff24_b(rf_filter_coeff24_b),
        .rf_filter_coeff25_a(rf_filter_coeff25_a),
        .rf_filter_coeff25_b(rf_filter_coeff25_b),
        .rf_filter_coeff26_a(rf_filter_coeff26_a),
        .rf_filter_coeff26_b(rf_filter_coeff26_b),
        .rf_filter_coeff27_a(rf_filter_coeff27_a),
        .rf_filter_coeff27_b(rf_filter_coeff27_b),
        .rf_filter_coeff28_a(rf_filter_coeff28_a),
        .rf_filter_coeff28_b(rf_filter_coeff28_b),
        .rf_filter_coeff29_a(rf_filter_coeff29_a),
        .rf_filter_coeff29_b(rf_filter_coeff29_b),
        .rf_filter_coeff30_a(rf_filter_coeff30_a),
        .rf_filter_coeff30_b(rf_filter_coeff30_b),
        .rf_filter_coeff31_a(rf_filter_coeff31_a),
        .rf_filter_coeff31_b(rf_filter_coeff31_b),
        .rf_filter_coeff32_a(rf_filter_coeff32_a),
        .rf_filter_coeff32_b(rf_filter_coeff32_b),
        .rf_filter_coeff33_a(rf_filter_coeff33_a),
        .rf_filter_coeff33_b(rf_filter_coeff33_b),
        .rf_filter_coeff34_a(rf_filter_coeff34_a),
```

```
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
```

```
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
```

```
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
```

```
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
```

```
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
```

```
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
```

```
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
```

```
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
```

```
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
```

```
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
```

```
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
```

```
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
```

```
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
.rf_filter_coeff479_a(rf_filter_coeff479_a),
.rf_filter_coeff479_b(rf_filter_coeff479_b),
.rf_filter_coeff480_a(rf_filter_coeff480_a),
.rf_filter_coeff480_b(rf_filter_coeff480_b),
.rf_filter_coeff481_a(rf_filter_coeff481_a),
.rf_filter_coeff481_b(rf_filter_coeff481_b),
.rf_filter_coeff482_a(rf_filter_coeff482_a),
.rf_filter_coeff482_b(rf_filter_coeff482_b),
.rf_filter_coeff483_a(rf_filter_coeff483_a),
.rf_filter_coeff483_b(rf_filter_coeff483_b),
.rf_filter_coeff484_a(rf_filter_coeff484_a),
.rf_filter_coeff484_b(rf_filter_coeff484_b),
.rf_filter_coeff485_a(rf_filter_coeff485_a),
.rf_filter_coeff485_b(rf_filter_coeff485_b),
.rf_filter_coeff486_a(rf_filter_coeff486_a),
.rf_filter_coeff486_b(rf_filter_coeff486_b),
.rf_filter_coeff487_a(rf_filter_coeff487_a),
.rf_filter_coeff487_b(rf_filter_coeff487_b),
.rf_filter_coeff488_a(rf_filter_coeff488_a),
.rf_filter_coeff488_b(rf_filter_coeff488_b),
.rf_filter_coeff489_a(rf_filter_coeff489_a),
.rf_filter_coeff489_b(rf_filter_coeff489_b),
.rf_filter_coeff490_a(rf_filter_coeff490_a),
.rf_filter_coeff490_b(rf_filter_coeff490_b),
.rf_filter_coeff491_a(rf_filter_coeff491_a),
.rf_filter_coeff491_b(rf_filter_coeff491_b),
.rf_filter_coeff492_a(rf_filter_coeff492_a),
.rf_filter_coeff492_b(rf_filter_coeff492_b),
.rf_filter_coeff493_a(rf_filter_coeff493_a),
.rf_filter_coeff493_b(rf_filter_coeff493_b),
.rf_filter_coeff494_a(rf_filter_coeff494_a),
.rf_filter_coeff494_b(rf_filter_coeff494_b),
.rf_filter_coeff495_a(rf_filter_coeff495_a),
.rf_filter_coeff495_b(rf_filter_coeff495_b),
```

```
        .rf_filter_coeff496_a(rf_filter_coeff496_a),
        .rf_filter_coeff496_b(rf_filter_coeff496_b),
        .rf_filter_coeff497_a(rf_filter_coeff497_a),
        .rf_filter_coeff497_b(rf_filter_coeff497_b),
        .rf_filter_coeff498_a(rf_filter_coeff498_a),
        .rf_filter_coeff498_b(rf_filter_coeff498_b),
        .rf_filter_coeff499_a(rf_filter_coeff499_a),
        .rf_filter_coeff499_b(rf_filter_coeff499_b),
        .rf_filter_coeff500_a(rf_filter_coeff500_a),
        .rf_filter_coeff500_b(rf_filter_coeff500_b),
        .rf_filter_coeff501_a(rf_filter_coeff501_a),
        .rf_filter_coeff501_b(rf_filter_coeff501_b),
        .rf_filter_coeff502_a(rf_filter_coeff502_a),
        .rf_filter_coeff502_b(rf_filter_coeff502_b),
        .rf_filter_coeff503_a(rf_filter_coeff503_a),
        .rf_filter_coeff503_b(rf_filter_coeff503_b),
        .rf_filter_coeff504_a(rf_filter_coeff504_a),
        .rf_filter_coeff504_b(rf_filter_coeff504_b),
        .rf_filter_coeff505_a(rf_filter_coeff505_a),
        .rf_filter_coeff505_b(rf_filter_coeff505_b),
        .rf_filter_coeff506_a(rf_filter_coeff506_a),
        .rf_filter_coeff506_b(rf_filter_coeff506_b),
        .rf_filter_coeff507_a(rf_filter_coeff507_a),
        .rf_filter_coeff507_b(rf_filter_coeff507_b),
        .rf_filter_coeff508_a(rf_filter_coeff508_a),
        .rf_filter_coeff508_b(rf_filter_coeff508_b),
        .rf_filter_coeff509_a(rf_filter_coeff509_a),
        .rf_filter_coeff509_b(rf_filter_coeff509_b),
        .rf_filter_coeff510_a(rf_filter_coeff510_a),
        .rf_filter_coeff510_b(rf_filter_coeff510_b),
        .rf_filter_coeff511_a(rf_filter_coeff511_a),
        .rf_filter_coeff511_b(rf_filter_coeff511_b)
    );


    trig_generator Trig_Gen(
        .clk                        (clk),                      // input: master clock
        .rst_n                      (rst_n),                    // input: reset
        .address                    (i2c_addr),                 // input: register address
        .wdata                      (i2c_wdata),                // input: data to be written
for a write op
        .xfc                        (i2c_xfc_write),            // input: transfer complete
        .trig_i2si_fifo_overrun_clr (trig_fifo_overrun),        // output: signal to reset
I2S input FIFO overrun
        .trig_i2so_fifo_underrun_clr (trig_fifo_underrun),      // output: signal to reset
I2S output FIFO underrun
        .trig_filter_ovf_flag_clear (trig_filter_ovf_flag_clear) // output: signal to reset
filter output saturation
    );
   //-----------------------------------------------------------------------------
   //-----------------------------------------------------------------------------

endmodule
```

register.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             register.v
// Create Date:             10/10/2015
// Last Modification:       3/29/2016
// Author:                  Julie Swift and Zachary Nelson
// Description: This sub-block allocates the control and status bits of other blocks
//              to specific addresses that can be read from or written to
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module register(rst_n, clk, addr, wdata, w_enable, wxfc, rxfc, ro_fifo_underrun,
                ro_fifo_overrun,  rdata, rf_i2si_bist_en, rf_filter_shift, rf_i2si_en,
                rf_filter_clip_en, ro_filter_ovf_flag,
               // trig_fifo_overrun, trig_fifo_underrun, trig_filter_ovf_flag_clear,
                rf_i2si_bist_start_val_a,
                rf_i2si_bist_start_val_b, rf_i2si_bist_incr, rf_i2si_bist_upper_limit_a, rf_i2si_bist_upper_limit_b,
                rf_filter_coeff0_a, rf_filter_coeff0_b,rf_filter_coeff1_a, rf_filter_coeff1_b,rf_filter_coeff2_a,
rf_filter_coeff2_b,rf_filter_coeff3_a, rf_filter_coeff3_b,rf_filter_coeff4_a, rf_filter_coeff4_b,rf_filter_coeff5_a,
rf_filter_coeff5_b,rf_filter_coeff6_a, rf_filter_coeff6_b,rf_filter_coeff7_a, rf_filter_coeff7_b,rf_filter_coeff8_a,
rf_filter_coeff8_b,rf_filter_coeff9_a, rf_filter_coeff9_b,rf_filter_coeff10_a, rf_filter_coeff10_b,rf_filter_coeff11_a,
rf_filter_coeff11_b,rf_filter_coeff12_a, rf_filter_coeff12_b,rf_filter_coeff13_a, rf_filter_coeff13_b,rf_filter_coeff14_a,
rf_filter_coeff14_b,rf_filter_coeff15_a, rf_filter_coeff15_b,rf_filter_coeff16_a, rf_filter_coeff16_b,rf_filter_coeff17_a,
rf_filter_coeff17_b,rf_filter_coeff18_a, rf_filter_coeff18_b,rf_filter_coeff19_a, rf_filter_coeff19_b,rf_filter_coeff20_a,
rf_filter_coeff20_b,rf_filter_coeff21_a, rf_filter_coeff21_b,rf_filter_coeff22_a, rf_filter_coeff22_b,rf_filter_coeff23_a,
rf_filter_coeff23_b,rf_filter_coeff24_a, rf_filter_coeff24_b,rf_filter_coeff25_a, rf_filter_coeff25_b,rf_filter_coeff26_a,
rf_filter_coeff26_b,rf_filter_coeff27_a, rf_filter_coeff27_b,rf_filter_coeff28_a, rf_filter_coeff28_b,rf_filter_coeff29_a,
rf_filter_coeff29_b,rf_filter_coeff30_a, rf_filter_coeff30_b,rf_filter_coeff31_a, rf_filter_coeff31_b,rf_filter_coeff32_a,
rf_filter_coeff32_b,rf_filter_coeff33_a, rf_filter_coeff33_b,rf_filter_coeff34_a, rf_filter_coeff34_b,rf_filter_coeff35_a,
rf_filter_coeff35_b,rf_filter_coeff36_a, rf_filter_coeff36_b,rf_filter_coeff37_a, rf_filter_coeff37_b,rf_filter_coeff38_a,
rf_filter_coeff38_b,rf_filter_coeff39_a, rf_filter_coeff39_b,rf_filter_coeff40_a, rf_filter_coeff40_b,rf_filter_coeff41_a,
rf_filter_coeff41_b,rf_filter_coeff42_a, rf_filter_coeff42_b,rf_filter_coeff43_a, rf_filter_coeff43_b,rf_filter_coeff44_a,
rf_filter_coeff44_b,rf_filter_coeff45_a, rf_filter_coeff45_b,rf_filter_coeff46_a, rf_filter_coeff46_b,rf_filter_coeff47_a,
rf_filter_coeff47_b,rf_filter_coeff48_a, rf_filter_coeff48_b,rf_filter_coeff49_a, rf_filter_coeff49_b,rf_filter_coeff50_a,
rf_filter_coeff50_b,rf_filter_coeff51_a, rf_filter_coeff51_b,rf_filter_coeff52_a, rf_filter_coeff52_b,rf_filter_coeff53_a,
rf_filter_coeff53_b,rf_filter_coeff54_a, rf_filter_coeff54_b,rf_filter_coeff55_a, rf_filter_coeff55_b,rf_filter_coeff56_a,
rf_filter_coeff56_b,rf_filter_coeff57_a, rf_filter_coeff57_b,rf_filter_coeff58_a, rf_filter_coeff58_b,rf_filter_coeff59_a,
rf_filter_coeff59_b,rf_filter_coeff60_a, rf_filter_coeff60_b,rf_filter_coeff61_a, rf_filter_coeff61_b,rf_filter_coeff62_a,
rf_filter_coeff62_b,rf_filter_coeff63_a, rf_filter_coeff63_b,rf_filter_coeff64_a, rf_filter_coeff64_b,rf_filter_coeff65_a,
rf_filter_coeff65_b,rf_filter_coeff66_a, rf_filter_coeff66_b,rf_filter_coeff67_a, rf_filter_coeff67_b,rf_filter_coeff68_a,
rf_filter_coeff68_b,rf_filter_coeff69_a, rf_filter_coeff69_b,rf_filter_coeff70_a, rf_filter_coeff70_b,rf_filter_coeff71_a,
rf_filter_coeff71_b,rf_filter_coeff72_a, rf_filter_coeff72_b,rf_filter_coeff73_a, rf_filter_coeff73_b,rf_filter_coeff74_a,
rf_filter_coeff74_b,rf_filter_coeff75_a, rf_filter_coeff75_b,rf_filter_coeff76_a, rf_filter_coeff76_b,rf_filter_coeff77_a,
rf_filter_coeff77_b,rf_filter_coeff78_a, rf_filter_coeff78_b,rf_filter_coeff79_a, rf_filter_coeff79_b,rf_filter_coeff80_a,
rf_filter_coeff80_b,rf_filter_coeff81_a, rf_filter_coeff81_b,rf_filter_coeff82_a, rf_filter_coeff82_b,rf_filter_coeff83_a,
rf_filter_coeff83_b,rf_filter_coeff84_a, rf_filter_coeff84_b,rf_filter_coeff85_a, rf_filter_coeff85_b,rf_filter_coeff86_a,
rf_filter_coeff86_b,rf_filter_coeff87_a, rf_filter_coeff87_b,rf_filter_coeff88_a, rf_filter_coeff88_b,rf_filter_coeff89_a,
rf_filter_coeff89_b,rf_filter_coeff90_a, rf_filter_coeff90_b,rf_filter_coeff91_a, rf_filter_coeff91_b,rf_filter_coeff92_a,
rf_filter_coeff92_b,rf_filter_coeff93_a, rf_filter_coeff93_b,rf_filter_coeff94_a, rf_filter_coeff94_b,rf_filter_coeff95_a,
rf_filter_coeff95_b,rf_filter_coeff96_a, rf_filter_coeff96_b,rf_filter_coeff97_a, rf_filter_coeff97_b,rf_filter_coeff98_a,
rf_filter_coeff98_b,rf_filter_coeff99_a, rf_filter_coeff99_b,rf_filter_coeff100_a, rf_filter_coeff100_b,rf_filter_coeff101_a,
rf_filter_coeff101_b,rf_filter_coeff102_a, rf_filter_coeff102_b,rf_filter_coeff103_a, rf_filter_coeff103_b,rf_filter_coeff104_a,
rf_filter_coeff104_b,rf_filter_coeff105_a, rf_filter_coeff105_b,rf_filter_coeff106_a, rf_filter_coeff106_b,rf_filter_coeff107_a,
rf_filter_coeff107_b,rf_filter_coeff108_a, rf_filter_coeff108_b,rf_filter_coeff109_a, rf_filter_coeff109_b,rf_filter_coeff110_a,
rf_filter_coeff110_b,rf_filter_coeff111_a, rf_filter_coeff111_b,rf_filter_coeff112_a, rf_filter_coeff112_b,rf_filter_coeff113_a,
rf_filter_coeff113_b,rf_filter_coeff114_a, rf_filter_coeff114_b,rf_filter_coeff115_a, rf_filter_coeff115_b,rf_filter_coeff116_a,
rf_filter_coeff116_b,rf_filter_coeff117_a, rf_filter_coeff117_b,rf_filter_coeff118_a, rf_filter_coeff118_b,rf_filter_coeff119_a,
rf_filter_coeff119_b,rf_filter_coeff120_a, rf_filter_coeff120_b,rf_filter_coeff121_a, rf_filter_coeff121_b,rf_filter_coeff122_a,
rf_filter_coeff122_b,rf_filter_coeff123_a, rf_filter_coeff123_b,rf_filter_coeff124_a, rf_filter_coeff124_b,rf_filter_coeff125_a,
rf_filter_coeff125_b,rf_filter_coeff126_a, rf_filter_coeff126_b,rf_filter_coeff127_a, rf_filter_coeff127_b,rf_filter_coeff128_a,
rf_filter_coeff128_b,rf_filter_coeff129_a, rf_filter_coeff129_b,rf_filter_coeff130_a, rf_filter_coeff130_b,rf_filter_coeff131_a,
rf_filter_coeff131_b,rf_filter_coeff132_a, rf_filter_coeff132_b,rf_filter_coeff133_a, rf_filter_coeff133_b,rf_filter_coeff134_a,
rf_filter_coeff134_b,rf_filter_coeff135_a, rf_filter_coeff135_b,rf_filter_coeff136_a, rf_filter_coeff136_b,rf_filter_coeff137_a,
rf_filter_coeff137_b,rf_filter_coeff138_a, rf_filter_coeff138_b,rf_filter_coeff139_a, rf_filter_coeff139_b,rf_filter_coeff140_a,
rf_filter_coeff140_b,rf_filter_coeff141_a, rf_filter_coeff141_b,rf_filter_coeff142_a, rf_filter_coeff142_b,rf_filter_coeff143_a,
rf_filter_coeff143_b,rf_filter_coeff144_a, rf_filter_coeff144_b,rf_filter_coeff145_a, rf_filter_coeff145_b,rf_filter_coeff146_a,
rf_filter_coeff146_b,rf_filter_coeff147_a, rf_filter_coeff147_b,rf_filter_coeff148_a, rf_filter_coeff148_b,rf_filter_coeff149_a,
rf_filter_coeff149_b,rf_filter_coeff150_a, rf_filter_coeff150_b,rf_filter_coeff151_a, rf_filter_coeff151_b,rf_filter_coeff152_a,
rf_filter_coeff152_b,rf_filter_coeff153_a, rf_filter_coeff153_b,rf_filter_coeff154_a, rf_filter_coeff154_b,rf_filter_coeff155_a,
rf_filter_coeff155_b,rf_filter_coeff156_a, rf_filter_coeff156_b,rf_filter_coeff157_a, rf_filter_coeff157_b,rf_filter_coeff158_a,
rf_filter_coeff158_b,rf_filter_coeff159_a, rf_filter_coeff159_b,rf_filter_coeff160_a, rf_filter_coeff160_b,rf_filter_coeff161_a,
rf_filter_coeff161_b,rf_filter_coeff162_a, rf_filter_coeff162_b,rf_filter_coeff163_a, rf_filter_coeff163_b,rf_filter_coeff164_a,
rf_filter_coeff164_b,rf_filter_coeff165_a, rf_filter_coeff165_b,rf_filter_coeff166_a, rf_filter_coeff166_b,rf_filter_coeff167_a,
rf_filter_coeff167_b,rf_filter_coeff168_a, rf_filter_coeff168_b,rf_filter_coeff169_a, rf_filter_coeff169_b,rf_filter_coeff170_a,
rf_filter_coeff170_b,rf_filter_coeff171_a, rf_filter_coeff171_b,rf_filter_coeff172_a, rf_filter_coeff172_b,rf_filter_coeff173_a,
rf_filter_coeff173_b,rf_filter_coeff174_a, rf_filter_coeff174_b,rf_filter_coeff175_a, rf_filter_coeff175_b,rf_filter_coeff176_a,
rf_filter_coeff176_b,rf_filter_coeff177_a, rf_filter_coeff177_b,rf_filter_coeff178_a, rf_filter_coeff178_b,rf_filter_coeff179_a,
rf_filter_coeff179_b,rf_filter_coeff180_a, rf_filter_coeff180_b,rf_filter_coeff181_a, rf_filter_coeff181_b,rf_filter_coeff182_a,
rf_filter_coeff182_b,rf_filter_coeff183_a, rf_filter_coeff183_b,rf_filter_coeff184_a, rf_filter_coeff184_b,rf_filter_coeff185_a,
rf_filter_coeff185_b,rf_filter_coeff186_a, rf_filter_coeff186_b,rf_filter_coeff187_a, rf_filter_coeff187_b,rf_filter_coeff188_a,
rf_filter_coeff188_b,rf_filter_coeff189_a, rf_filter_coeff189_b,rf_filter_coeff190_a, rf_filter_coeff190_b,rf_filter_coeff191_a,
rf_filter_coeff191_b,rf_filter_coeff192_a, rf_filter_coeff192_b,rf_filter_coeff193_a, rf_filter_coeff193_b,rf_filter_coeff194_a,
rf_filter_coeff194_b,rf_filter_coeff195_a, rf_filter_coeff195_b,rf_filter_coeff196_a, rf_filter_coeff196_b,rf_filter_coeff197_a,
rf_filter_coeff197_b,rf_filter_coeff198_a, rf_filter_coeff198_b,rf_filter_coeff199_a, rf_filter_coeff199_b,rf_filter_coeff200_a,
rf_filter_coeff200_b,rf_filter_coeff201_a, rf_filter_coeff201_b,rf_filter_coeff202_a, rf_filter_coeff202_b,rf_filter_coeff203_a,
rf_filter_coeff203_b,rf_filter_coeff204_a, rf_filter_coeff204_b,rf_filter_coeff205_a, rf_filter_coeff205_b,rf_filter_coeff206_a,
rf_filter_coeff206_b,rf_filter_coeff207_a, rf_filter_coeff207_b,rf_filter_coeff208_a, rf_filter_coeff208_b,rf_filter_coeff209_a,
rf_filter_coeff209_b,rf_filter_coeff210_a, rf_filter_coeff210_b,rf_filter_coeff211_a, rf_filter_coeff211_b,rf_filter_coeff212_a,
rf_filter_coeff212_b,rf_filter_coeff213_a, rf_filter_coeff213_b,rf_filter_coeff214_a, rf_filter_coeff214_b,rf_filter_coeff215_a,
rf_filter_coeff215_b,rf_filter_coeff216_a, rf_filter_coeff216_b,rf_filter_coeff217_a, rf_filter_coeff217_b,rf_filter_coeff218_a,
```

rf_filter_coeff218_b,rf_filter_coeff219_a, rf_filter_coeff219_b,rf_filter_coeff220_a, rf_filter_coeff220_b,rf_filter_coeff221_a,
rf_filter_coeff221_b,rf_filter_coeff222_a, rf_filter_coeff222_b,rf_filter_coeff223_a, rf_filter_coeff223_b,rf_filter_coeff224_a,
rf_filter_coeff224_b,rf_filter_coeff225_a, rf_filter_coeff225_b,rf_filter_coeff226_a, rf_filter_coeff226_b,rf_filter_coeff227_a,
rf_filter_coeff227_b,rf_filter_coeff228_a, rf_filter_coeff228_b,rf_filter_coeff229_a, rf_filter_coeff229_b,rf_filter_coeff230_a,
rf_filter_coeff230_b,rf_filter_coeff231_a, rf_filter_coeff231_b,rf_filter_coeff232_a, rf_filter_coeff232_b,rf_filter_coeff233_a,
rf_filter_coeff233_b,rf_filter_coeff234_a, rf_filter_coeff234_b,rf_filter_coeff235_a, rf_filter_coeff235_b,rf_filter_coeff236_a,
rf_filter_coeff236_b,rf_filter_coeff237_a, rf_filter_coeff237_b,rf_filter_coeff238_a, rf_filter_coeff238_b,rf_filter_coeff239_a,
rf_filter_coeff239_b,rf_filter_coeff240_a, rf_filter_coeff240_b,rf_filter_coeff241_a, rf_filter_coeff241_b,rf_filter_coeff242_a,
rf_filter_coeff242_b,rf_filter_coeff243_a, rf_filter_coeff243_b,rf_filter_coeff244_a, rf_filter_coeff244_b,rf_filter_coeff245_a,
rf_filter_coeff245_b,rf_filter_coeff246_a, rf_filter_coeff246_b,rf_filter_coeff247_a, rf_filter_coeff247_b,rf_filter_coeff248_a,
rf_filter_coeff248_b,rf_filter_coeff249_a, rf_filter_coeff249_b,rf_filter_coeff250_a, rf_filter_coeff250_b,rf_filter_coeff251_a,
rf_filter_coeff251_b,rf_filter_coeff252_a, rf_filter_coeff252_b,rf_filter_coeff253_a, rf_filter_coeff253_b,rf_filter_coeff254_a,
rf_filter_coeff254_b,rf_filter_coeff255_a, rf_filter_coeff255_b,rf_filter_coeff256_a, rf_filter_coeff256_b,rf_filter_coeff257_a,
rf_filter_coeff257_b,rf_filter_coeff258_a, rf_filter_coeff258_b,rf_filter_coeff259_a, rf_filter_coeff259_b,rf_filter_coeff260_a,
rf_filter_coeff260_b,rf_filter_coeff261_a, rf_filter_coeff261_b,rf_filter_coeff262_a, rf_filter_coeff262_b,rf_filter_coeff263_a,
rf_filter_coeff263_b,rf_filter_coeff264_a, rf_filter_coeff264_b,rf_filter_coeff265_a, rf_filter_coeff265_b,rf_filter_coeff266_a,
rf_filter_coeff266_b,rf_filter_coeff267_a, rf_filter_coeff267_b,rf_filter_coeff268_a, rf_filter_coeff268_b,rf_filter_coeff269_a,
rf_filter_coeff269_b,rf_filter_coeff270_a, rf_filter_coeff270_b,rf_filter_coeff271_a, rf_filter_coeff271_b,rf_filter_coeff272_a,
rf_filter_coeff272_b,rf_filter_coeff273_a, rf_filter_coeff273_b,rf_filter_coeff274_a, rf_filter_coeff274_b,rf_filter_coeff275_a,
rf_filter_coeff275_b,rf_filter_coeff276_a, rf_filter_coeff276_b,rf_filter_coeff277_a, rf_filter_coeff277_b,rf_filter_coeff278_a,
rf_filter_coeff278_b,rf_filter_coeff279_a, rf_filter_coeff279_b,rf_filter_coeff280_a, rf_filter_coeff280_b,rf_filter_coeff281_a,
rf_filter_coeff281_b,rf_filter_coeff282_a, rf_filter_coeff282_b,rf_filter_coeff283_a, rf_filter_coeff283_b,rf_filter_coeff284_a,
rf_filter_coeff284_b,rf_filter_coeff285_a, rf_filter_coeff285_b,rf_filter_coeff286_a, rf_filter_coeff286_b,rf_filter_coeff287_a,
rf_filter_coeff287_b,rf_filter_coeff288_a, rf_filter_coeff288_b,rf_filter_coeff289_a, rf_filter_coeff289_b,rf_filter_coeff290_a,
rf_filter_coeff290_b,rf_filter_coeff291_a, rf_filter_coeff291_b,rf_filter_coeff292_a, rf_filter_coeff292_b,rf_filter_coeff293_a,
rf_filter_coeff293_b,rf_filter_coeff294_a, rf_filter_coeff294_b,rf_filter_coeff295_a, rf_filter_coeff295_b,rf_filter_coeff296_a,
rf_filter_coeff296_b,rf_filter_coeff297_a, rf_filter_coeff297_b,rf_filter_coeff298_a, rf_filter_coeff298_b,rf_filter_coeff299_a,
rf_filter_coeff299_b,rf_filter_coeff300_a, rf_filter_coeff300_b,rf_filter_coeff301_a, rf_filter_coeff301_b,rf_filter_coeff302_a,
rf_filter_coeff302_b,rf_filter_coeff303_a, rf_filter_coeff303_b,rf_filter_coeff304_a, rf_filter_coeff304_b,rf_filter_coeff305_a,
rf_filter_coeff305_b,rf_filter_coeff306_a, rf_filter_coeff306_b,rf_filter_coeff307_a, rf_filter_coeff307_b,rf_filter_coeff308_a,
rf_filter_coeff308_b,rf_filter_coeff309_a, rf_filter_coeff309_b,rf_filter_coeff310_a, rf_filter_coeff310_b,rf_filter_coeff311_a,
rf_filter_coeff311_b,rf_filter_coeff312_a, rf_filter_coeff312_b,rf_filter_coeff313_a, rf_filter_coeff313_b,rf_filter_coeff314_a,
rf_filter_coeff314_b,rf_filter_coeff315_a, rf_filter_coeff315_b,rf_filter_coeff316_a, rf_filter_coeff316_b,rf_filter_coeff317_a,
rf_filter_coeff317_b,rf_filter_coeff318_a, rf_filter_coeff318_b,rf_filter_coeff319_a, rf_filter_coeff319_b,rf_filter_coeff320_a,
rf_filter_coeff320_b,rf_filter_coeff321_a, rf_filter_coeff321_b,rf_filter_coeff322_a, rf_filter_coeff322_b,rf_filter_coeff323_a,
rf_filter_coeff323_b,rf_filter_coeff324_a, rf_filter_coeff324_b,rf_filter_coeff325_a, rf_filter_coeff325_b,rf_filter_coeff326_a,
rf_filter_coeff326_b,rf_filter_coeff327_a, rf_filter_coeff327_b,rf_filter_coeff328_a, rf_filter_coeff328_b,rf_filter_coeff329_a,
rf_filter_coeff329_b,rf_filter_coeff330_a, rf_filter_coeff330_b,rf_filter_coeff331_a, rf_filter_coeff331_b,rf_filter_coeff332_a,
rf_filter_coeff332_b,rf_filter_coeff333_a, rf_filter_coeff333_b,rf_filter_coeff334_a, rf_filter_coeff334_b,rf_filter_coeff335_a,
rf_filter_coeff335_b,rf_filter_coeff336_a, rf_filter_coeff336_b,rf_filter_coeff337_a, rf_filter_coeff337_b,rf_filter_coeff338_a,
rf_filter_coeff338_b,rf_filter_coeff339_a, rf_filter_coeff339_b,rf_filter_coeff340_a, rf_filter_coeff340_b,rf_filter_coeff341_a,
rf_filter_coeff341_b,rf_filter_coeff342_a, rf_filter_coeff342_b,rf_filter_coeff343_a, rf_filter_coeff343_b,rf_filter_coeff344_a,
rf_filter_coeff344_b,rf_filter_coeff345_a, rf_filter_coeff345_b,rf_filter_coeff346_a, rf_filter_coeff346_b,rf_filter_coeff347_a,
rf_filter_coeff347_b,rf_filter_coeff348_a, rf_filter_coeff348_b,rf_filter_coeff349_a, rf_filter_coeff349_b,rf_filter_coeff350_a,
rf_filter_coeff350_b,rf_filter_coeff351_a, rf_filter_coeff351_b,rf_filter_coeff352_a, rf_filter_coeff352_b,rf_filter_coeff353_a,
rf_filter_coeff353_b,rf_filter_coeff354_a, rf_filter_coeff354_b,rf_filter_coeff355_a, rf_filter_coeff355_b,rf_filter_coeff356_a,
rf_filter_coeff356_b,rf_filter_coeff357_a, rf_filter_coeff357_b,rf_filter_coeff358_a, rf_filter_coeff358_b,rf_filter_coeff359_a,
rf_filter_coeff359_b,rf_filter_coeff360_a, rf_filter_coeff360_b,rf_filter_coeff361_a, rf_filter_coeff361_b,rf_filter_coeff362_a,
rf_filter_coeff362_b,rf_filter_coeff363_a, rf_filter_coeff363_b,rf_filter_coeff364_a, rf_filter_coeff364_b,rf_filter_coeff365_a,
rf_filter_coeff365_b,rf_filter_coeff366_a, rf_filter_coeff366_b,rf_filter_coeff367_a, rf_filter_coeff367_b,rf_filter_coeff368_a,
rf_filter_coeff368_b,rf_filter_coeff369_a, rf_filter_coeff369_b,rf_filter_coeff370_a, rf_filter_coeff370_b,rf_filter_coeff371_a,
rf_filter_coeff371_b,rf_filter_coeff372_a, rf_filter_coeff372_b,rf_filter_coeff373_a, rf_filter_coeff373_b,rf_filter_coeff374_a,
rf_filter_coeff374_b,rf_filter_coeff375_a, rf_filter_coeff375_b,rf_filter_coeff376_a, rf_filter_coeff376_b,rf_filter_coeff377_a,
rf_filter_coeff377_b,rf_filter_coeff378_a, rf_filter_coeff378_b,rf_filter_coeff379_a, rf_filter_coeff379_b,rf_filter_coeff380_a,
rf_filter_coeff380_b,rf_filter_coeff381_a, rf_filter_coeff381_b,rf_filter_coeff382_a, rf_filter_coeff382_b,rf_filter_coeff383_a,
rf_filter_coeff383_b,rf_filter_coeff384_a, rf_filter_coeff384_b,rf_filter_coeff385_a, rf_filter_coeff385_b,rf_filter_coeff386_a,
rf_filter_coeff386_b,rf_filter_coeff387_a, rf_filter_coeff387_b,rf_filter_coeff388_a, rf_filter_coeff388_b,rf_filter_coeff389_a,
rf_filter_coeff389_b,rf_filter_coeff390_a, rf_filter_coeff390_b,rf_filter_coeff391_a, rf_filter_coeff391_b,rf_filter_coeff392_a,
rf_filter_coeff392_b,rf_filter_coeff393_a, rf_filter_coeff393_b,rf_filter_coeff394_a, rf_filter_coeff394_b,rf_filter_coeff395_a,
rf_filter_coeff395_b,rf_filter_coeff396_a, rf_filter_coeff396_b,rf_filter_coeff397_a, rf_filter_coeff397_b,rf_filter_coeff398_a,
rf_filter_coeff398_b,rf_filter_coeff399_a, rf_filter_coeff399_b,rf_filter_coeff400_a, rf_filter_coeff400_b,rf_filter_coeff401_a,
rf_filter_coeff401_b,rf_filter_coeff402_a, rf_filter_coeff402_b,rf_filter_coeff403_a, rf_filter_coeff403_b,rf_filter_coeff404_a,
rf_filter_coeff404_b,rf_filter_coeff405_a, rf_filter_coeff405_b,rf_filter_coeff406_a, rf_filter_coeff406_b,rf_filter_coeff407_a,
rf_filter_coeff407_b,rf_filter_coeff408_a, rf_filter_coeff408_b,rf_filter_coeff409_a, rf_filter_coeff409_b,rf_filter_coeff410_a,
rf_filter_coeff410_b,rf_filter_coeff411_a, rf_filter_coeff411_b,rf_filter_coeff412_a, rf_filter_coeff412_b,rf_filter_coeff413_a,
rf_filter_coeff413_b,rf_filter_coeff414_a, rf_filter_coeff414_b,rf_filter_coeff415_a, rf_filter_coeff415_b,rf_filter_coeff416_a,
rf_filter_coeff416_b,rf_filter_coeff417_a, rf_filter_coeff417_b,rf_filter_coeff418_a, rf_filter_coeff418_b,rf_filter_coeff419_a,
rf_filter_coeff419_b,rf_filter_coeff420_a, rf_filter_coeff420_b,rf_filter_coeff421_a, rf_filter_coeff421_b,rf_filter_coeff422_a,
rf_filter_coeff422_b,rf_filter_coeff423_a, rf_filter_coeff423_b,rf_filter_coeff424_a, rf_filter_coeff424_b,rf_filter_coeff425_a,
rf_filter_coeff425_b,rf_filter_coeff426_a, rf_filter_coeff426_b,rf_filter_coeff427_a, rf_filter_coeff427_b,rf_filter_coeff428_a,
rf_filter_coeff428_b,rf_filter_coeff429_a, rf_filter_coeff429_b,rf_filter_coeff430_a, rf_filter_coeff430_b,rf_filter_coeff431_a,
rf_filter_coeff431_b,rf_filter_coeff432_a, rf_filter_coeff432_b,rf_filter_coeff433_a, rf_filter_coeff433_b,rf_filter_coeff434_a,
rf_filter_coeff434_b,rf_filter_coeff435_a, rf_filter_coeff435_b,rf_filter_coeff436_a, rf_filter_coeff436_b,rf_filter_coeff437_a,
rf_filter_coeff437_b,rf_filter_coeff438_a, rf_filter_coeff438_b,rf_filter_coeff439_a, rf_filter_coeff439_b,rf_filter_coeff440_a,
rf_filter_coeff440_b,rf_filter_coeff441_a, rf_filter_coeff441_b,rf_filter_coeff442_a, rf_filter_coeff442_b,rf_filter_coeff443_a,
rf_filter_coeff443_b,rf_filter_coeff444_a, rf_filter_coeff444_b,rf_filter_coeff445_a, rf_filter_coeff445_b,rf_filter_coeff446_a,
rf_filter_coeff446_b,rf_filter_coeff447_a, rf_filter_coeff447_b,rf_filter_coeff448_a, rf_filter_coeff448_b,rf_filter_coeff449_a,
rf_filter_coeff449_b,rf_filter_coeff450_a, rf_filter_coeff450_b,rf_filter_coeff451_a, rf_filter_coeff451_b,rf_filter_coeff452_a,
rf_filter_coeff452_b,rf_filter_coeff453_a, rf_filter_coeff453_b,rf_filter_coeff454_a, rf_filter_coeff454_b,rf_filter_coeff455_a,
rf_filter_coeff455_b,rf_filter_coeff456_a, rf_filter_coeff456_b,rf_filter_coeff457_a, rf_filter_coeff457_b,rf_filter_coeff458_a,
rf_filter_coeff458_b,rf_filter_coeff459_a, rf_filter_coeff459_b,rf_filter_coeff460_a, rf_filter_coeff460_b,rf_filter_coeff461_a,
rf_filter_coeff461_b,rf_filter_coeff462_a, rf_filter_coeff462_b,rf_filter_coeff463_a, rf_filter_coeff463_b,rf_filter_coeff464_a,
rf_filter_coeff464_b,rf_filter_coeff465_a, rf_filter_coeff465_b,rf_filter_coeff466_a, rf_filter_coeff466_b,rf_filter_coeff467_a,
rf_filter_coeff467_b,rf_filter_coeff468_a, rf_filter_coeff468_b,rf_filter_coeff469_a, rf_filter_coeff469_b,rf_filter_coeff470_a,
rf_filter_coeff470_b,rf_filter_coeff471_a, rf_filter_coeff471_b,rf_filter_coeff472_a, rf_filter_coeff472_b,rf_filter_coeff473_a,
rf_filter_coeff473_b,rf_filter_coeff474_a, rf_filter_coeff474_b,rf_filter_coeff475_a, rf_filter_coeff475_b,rf_filter_coeff476_a,
rf_filter_coeff476_b,rf_filter_coeff477_a, rf_filter_coeff477_b,rf_filter_coeff478_a, rf_filter_coeff478_b,rf_filter_coeff479_a,
rf_filter_coeff479_b,rf_filter_coeff480_a, rf_filter_coeff480_b,rf_filter_coeff481_a, rf_filter_coeff481_b,rf_filter_coeff482_a,
rf_filter_coeff482_b,rf_filter_coeff483_a, rf_filter_coeff483_b,rf_filter_coeff484_a, rf_filter_coeff484_b,rf_filter_coeff485_a,
rf_filter_coeff485_b,rf_filter_coeff486_a, rf_filter_coeff486_b,rf_filter_coeff487_a, rf_filter_coeff487_b,rf_filter_coeff488_a,
rf_filter_coeff488_b,rf_filter_coeff489_a, rf_filter_coeff489_b,rf_filter_coeff490_a, rf_filter_coeff490_b,rf_filter_coeff491_a,
rf_filter_coeff491_b,rf_filter_coeff492_a, rf_filter_coeff492_b,rf_filter_coeff493_a, rf_filter_coeff493_b,rf_filter_coeff494_a,
rf_filter_coeff494_b,rf_filter_coeff495_a, rf_filter_coeff495_b,rf_filter_coeff496_a, rf_filter_coeff496_b,rf_filter_coeff497_a,
rf_filter_coeff497_b,rf_filter_coeff498_a, rf_filter_coeff498_b,rf_filter_coeff499_a, rf_filter_coeff499_b,rf_filter_coeff500_a,
rf_filter_coeff500_b,rf_filter_coeff501_a, rf_filter_coeff501_b,rf_filter_coeff502_a, rf_filter_coeff502_b,rf_filter_coeff503_a,

```
rf_filter_coeff503_b,rf_filter_coeff504_a, rf_filter_coeff504_b,rf_filter_coeff505_a, rf_filter_coeff505_b,rf_filter_coeff506_a,
rf_filter_coeff506_b,rf_filter_coeff507_a, rf_filter_coeff507_b,rf_filter_coeff508_a, rf_filter_coeff508_b,rf_filter_coeff509_a,
rf_filter_coeff509_b,rf_filter_coeff510_a, rf_filter_coeff510_b,rf_filter_coeff511_a, rf_filter_coeff511_b);

    // Inputs
    input                       clk;                                                                                    // master
clock
    input                       rst_n;                                                                                     // reset
    input       [10:0]          addr;                                                                                // 11 bit register
address
    input       [ 7:0]          wdata;                                                                               // write data
    input                       w_enable;                                                                                // write
enable
    input                       wxfc;                                                                                    // write
transfer complete
    input                       ro_fifo_underrun;                                                                        // I2S
output audio FIFO underrun
    input                       ro_fifo_overrun;                                                                         // I2S
input audio FIFO overru
            input                       ro_filter_ovf_flag;
// Filter status bit

    // Outputs
            output reg
rxfc;                                                                                                                  // read
transfer complete
    output reg  [ 7:0]          rdata;                                                                   // read data
    output reg                  rf_i2si_bist_en;                                                  // 0- deserializer signal 1- BIST
signal
    output reg  [ 2:0]          rf_filter_shift;                                                         // number of bit postions to
shift after filter accumulator
    output reg                  rf_filter_clip_en;                                                // 0- no clipping 1- performs
clipping
    output reg                  rf_i2si_en;                                         // Enable bit for Deserializer. 0 = inactive, 1 = active
    reg                         trig_fifo_overrun;                                                // signal to reset I2S input FIFO
overrun
    reg                         trig_fifo_underrun;                                               // signal to reset I2S output FIFO
underrun
    reg                         trig_filter_ovf_flag_clear;
    output reg  [ 7:0]          rf_i2si_bist_start_val_a;                   // start value of BIST signal
    output reg  [ 3:0]          rf_i2si_bist_start_val_b;                   // start value of BIST signal
    output reg  [ 7:0]          rf_i2si_bist_incr;                          // BIST signal increment value
    output reg  [ 7:0]          rf_i2si_bist_upper_limit_a;                 // upper limit of BIST signal
    output reg  [ 3:0]          rf_i2si_bist_upper_limit_b;                 // upper limit of BIST signal
            output reg  [ 7:0]          rf_filter_coeff0_a, rf_filter_coeff0_b,rf_filter_coeff1_a,
rf_filter_coeff1_b,rf_filter_coeff2_a, rf_filter_coeff2_b,rf_filter_coeff3_a, rf_filter_coeff3_b,rf_filter_coeff4_a,
rf_filter_coeff4_b,rf_filter_coeff5_a, rf_filter_coeff5_b,rf_filter_coeff6_a, rf_filter_coeff6_b,rf_filter_coeff7_a,
rf_filter_coeff7_b,rf_filter_coeff8_a, rf_filter_coeff8_b,rf_filter_coeff9_a, rf_filter_coeff9_b,rf_filter_coeff10_a,
rf_filter_coeff10_b,rf_filter_coeff11_a, rf_filter_coeff11_b,rf_filter_coeff12_a, rf_filter_coeff12_b,rf_filter_coeff13_a,
rf_filter_coeff13_b,rf_filter_coeff14_a, rf_filter_coeff14_b,rf_filter_coeff15_a, rf_filter_coeff15_b,rf_filter_coeff16_a,
rf_filter_coeff16_b,rf_filter_coeff17_a, rf_filter_coeff17_b,rf_filter_coeff18_a, rf_filter_coeff18_b,rf_filter_coeff19_a,
rf_filter_coeff19_b,rf_filter_coeff20_a, rf_filter_coeff20_b,rf_filter_coeff21_a, rf_filter_coeff21_b,rf_filter_coeff22_a,
rf_filter_coeff22_b,rf_filter_coeff23_a, rf_filter_coeff23_b,rf_filter_coeff24_a, rf_filter_coeff24_b,rf_filter_coeff25_a,
rf_filter_coeff25_b,rf_filter_coeff26_a, rf_filter_coeff26_b,rf_filter_coeff27_a, rf_filter_coeff27_b,rf_filter_coeff28_a,
rf_filter_coeff28_b,rf_filter_coeff29_a, rf_filter_coeff29_b,rf_filter_coeff30_a, rf_filter_coeff30_b,rf_filter_coeff31_a,
rf_filter_coeff31_b,rf_filter_coeff32_a, rf_filter_coeff32_b,rf_filter_coeff33_a, rf_filter_coeff33_b,rf_filter_coeff34_a,
rf_filter_coeff34_b,rf_filter_coeff35_a, rf_filter_coeff35_b,rf_filter_coeff36_a, rf_filter_coeff36_b,rf_filter_coeff37_a,
rf_filter_coeff37_b,rf_filter_coeff38_a, rf_filter_coeff38_b,rf_filter_coeff39_a, rf_filter_coeff39_b,rf_filter_coeff40_a,
rf_filter_coeff40_b,rf_filter_coeff41_a, rf_filter_coeff41_b,rf_filter_coeff42_a, rf_filter_coeff42_b,rf_filter_coeff43_a,
rf_filter_coeff43_b,rf_filter_coeff44_a, rf_filter_coeff44_b,rf_filter_coeff45_a, rf_filter_coeff45_b,rf_filter_coeff46_a,
rf_filter_coeff46_b,rf_filter_coeff47_a, rf_filter_coeff47_b,rf_filter_coeff48_a, rf_filter_coeff48_b,rf_filter_coeff49_a,
rf_filter_coeff49_b,rf_filter_coeff50_a, rf_filter_coeff50_b,rf_filter_coeff51_a, rf_filter_coeff51_b,rf_filter_coeff52_a,
rf_filter_coeff52_b,rf_filter_coeff53_a, rf_filter_coeff53_b,rf_filter_coeff54_a, rf_filter_coeff54_b,rf_filter_coeff55_a,
rf_filter_coeff55_b,rf_filter_coeff56_a, rf_filter_coeff56_b,rf_filter_coeff57_a, rf_filter_coeff57_b,rf_filter_coeff58_a,
rf_filter_coeff58_b,rf_filter_coeff59_a, rf_filter_coeff59_b,rf_filter_coeff60_a, rf_filter_coeff60_b,rf_filter_coeff61_a,
rf_filter_coeff61_b,rf_filter_coeff62_a, rf_filter_coeff62_b,rf_filter_coeff63_a, rf_filter_coeff63_b,rf_filter_coeff64_a,
rf_filter_coeff64_b,rf_filter_coeff65_a, rf_filter_coeff65_b,rf_filter_coeff66_a, rf_filter_coeff66_b,rf_filter_coeff67_a,
rf_filter_coeff67_b,rf_filter_coeff68_a, rf_filter_coeff68_b,rf_filter_coeff69_a, rf_filter_coeff69_b,rf_filter_coeff70_a,
rf_filter_coeff70_b,rf_filter_coeff71_a, rf_filter_coeff71_b,rf_filter_coeff72_a, rf_filter_coeff72_b,rf_filter_coeff73_a,
rf_filter_coeff73_b,rf_filter_coeff74_a, rf_filter_coeff74_b,rf_filter_coeff75_a, rf_filter_coeff75_b,rf_filter_coeff76_a,
rf_filter_coeff76_b,rf_filter_coeff77_a, rf_filter_coeff77_b,rf_filter_coeff78_a, rf_filter_coeff78_b,rf_filter_coeff79_a,
rf_filter_coeff79_b,rf_filter_coeff80_a, rf_filter_coeff80_b,rf_filter_coeff81_a, rf_filter_coeff81_b,rf_filter_coeff82_a,
rf_filter_coeff82_b,rf_filter_coeff83_a, rf_filter_coeff83_b,rf_filter_coeff84_a, rf_filter_coeff84_b,rf_filter_coeff85_a,
rf_filter_coeff85_b,rf_filter_coeff86_a, rf_filter_coeff86_b,rf_filter_coeff87_a, rf_filter_coeff87_b,rf_filter_coeff88_a,
rf_filter_coeff88_b,rf_filter_coeff89_a, rf_filter_coeff89_b,rf_filter_coeff90_a, rf_filter_coeff90_b,rf_filter_coeff91_a,
rf_filter_coeff91_b,rf_filter_coeff92_a, rf_filter_coeff92_b,rf_filter_coeff93_a, rf_filter_coeff93_b,rf_filter_coeff94_a,
rf_filter_coeff94_b,rf_filter_coeff95_a, rf_filter_coeff95_b,rf_filter_coeff96_a, rf_filter_coeff96_b,rf_filter_coeff97_a,
rf_filter_coeff97_b,rf_filter_coeff98_a, rf_filter_coeff98_b,rf_filter_coeff99_a, rf_filter_coeff99_b,rf_filter_coeff100_a,
rf_filter_coeff100_b,rf_filter_coeff101_a, rf_filter_coeff101_b,rf_filter_coeff102_a, rf_filter_coeff102_b,rf_filter_coeff103_a,
rf_filter_coeff103_b,rf_filter_coeff104_a, rf_filter_coeff104_b,rf_filter_coeff105_a, rf_filter_coeff105_b,rf_filter_coeff106_a,
rf_filter_coeff106_b,rf_filter_coeff107_a, rf_filter_coeff107_b,rf_filter_coeff108_a, rf_filter_coeff108_b,rf_filter_coeff109_a,
rf_filter_coeff109_b,rf_filter_coeff110_a, rf_filter_coeff110_b,rf_filter_coeff111_a, rf_filter_coeff111_b,rf_filter_coeff112_a,
rf_filter_coeff112_b,rf_filter_coeff113_a, rf_filter_coeff113_b,rf_filter_coeff114_a, rf_filter_coeff114_b,rf_filter_coeff115_a,
rf_filter_coeff115_b,rf_filter_coeff116_a, rf_filter_coeff116_b,rf_filter_coeff117_a, rf_filter_coeff117_b,rf_filter_coeff118_a,
rf_filter_coeff118_b,rf_filter_coeff119_a, rf_filter_coeff119_b,rf_filter_coeff120_a, rf_filter_coeff120_b,rf_filter_coeff121_a,
rf_filter_coeff121_b,rf_filter_coeff122_a, rf_filter_coeff122_b,rf_filter_coeff123_a, rf_filter_coeff123_b,rf_filter_coeff124_a,
rf_filter_coeff124_b,rf_filter_coeff125_a, rf_filter_coeff125_b,rf_filter_coeff126_a, rf_filter_coeff126_b,rf_filter_coeff127_a,
rf_filter_coeff127_b,rf_filter_coeff128_a, rf_filter_coeff128_b,rf_filter_coeff129_a, rf_filter_coeff129_b,rf_filter_coeff130_a,
rf_filter_coeff130_b,rf_filter_coeff131_a, rf_filter_coeff131_b,rf_filter_coeff132_a, rf_filter_coeff132_b,rf_filter_coeff133_a,
rf_filter_coeff133_b,rf_filter_coeff134_a, rf_filter_coeff134_b,rf_filter_coeff135_a, rf_filter_coeff135_b,rf_filter_coeff136_a,
rf_filter_coeff136_b,rf_filter_coeff137_a, rf_filter_coeff137_b,rf_filter_coeff138_a, rf_filter_coeff138_b,rf_filter_coeff139_a,
rf_filter_coeff139_b,rf_filter_coeff140_a, rf_filter_coeff140_b,rf_filter_coeff141_a, rf_filter_coeff141_b,rf_filter_coeff142_a,
rf_filter_coeff142_b,rf_filter_coeff143_a, rf_filter_coeff143_b,rf_filter_coeff144_a, rf_filter_coeff144_b,rf_filter_coeff145_a,
rf_filter_coeff145_b,rf_filter_coeff146_a, rf_filter_coeff146_b,rf_filter_coeff147_a, rf_filter_coeff147_b,rf_filter_coeff148_a,
rf_filter_coeff148_b,rf_filter_coeff149_a, rf_filter_coeff149_b,rf_filter_coeff150_a, rf_filter_coeff150_b,rf_filter_coeff151_a,
```

rf_filter_coeff151_b,rf_filter_coeff152_a, rf_filter_coeff152_b,rf_filter_coeff153_a, rf_filter_coeff153_b,rf_filter_coeff154_a,
rf_filter_coeff154_b,rf_filter_coeff155_a, rf_filter_coeff155_b,rf_filter_coeff156_a, rf_filter_coeff156_b,rf_filter_coeff157_a,
rf_filter_coeff157_b,rf_filter_coeff158_a, rf_filter_coeff158_b,rf_filter_coeff159_a, rf_filter_coeff159_b,rf_filter_coeff160_a,
rf_filter_coeff160_b,rf_filter_coeff161_a, rf_filter_coeff161_b,rf_filter_coeff162_a, rf_filter_coeff162_b,rf_filter_coeff163_a,
rf_filter_coeff163_b,rf_filter_coeff164_a, rf_filter_coeff164_b,rf_filter_coeff165_a, rf_filter_coeff165_b,rf_filter_coeff166_a,
rf_filter_coeff166_b,rf_filter_coeff167_a, rf_filter_coeff167_b,rf_filter_coeff168_a, rf_filter_coeff168_b,rf_filter_coeff169_a,
rf_filter_coeff169_b,rf_filter_coeff170_a, rf_filter_coeff170_b,rf_filter_coeff171_a, rf_filter_coeff171_b,rf_filter_coeff172_a,
rf_filter_coeff172_b,rf_filter_coeff173_a, rf_filter_coeff173_b,rf_filter_coeff174_a, rf_filter_coeff174_b,rf_filter_coeff175_a,
rf_filter_coeff175_b,rf_filter_coeff176_a, rf_filter_coeff176_b,rf_filter_coeff177_a, rf_filter_coeff177_b,rf_filter_coeff178_a,
rf_filter_coeff178_b,rf_filter_coeff179_a, rf_filter_coeff179_b,rf_filter_coeff180_a, rf_filter_coeff180_b,rf_filter_coeff181_a,
rf_filter_coeff181_b,rf_filter_coeff182_a, rf_filter_coeff182_b,rf_filter_coeff183_a, rf_filter_coeff183_b,rf_filter_coeff184_a,
rf_filter_coeff184_b,rf_filter_coeff185_a, rf_filter_coeff185_b,rf_filter_coeff186_a, rf_filter_coeff186_b,rf_filter_coeff187_a,
rf_filter_coeff187_b,rf_filter_coeff188_a, rf_filter_coeff188_b,rf_filter_coeff189_a, rf_filter_coeff189_b,rf_filter_coeff190_a,
rf_filter_coeff190_b,rf_filter_coeff191_a, rf_filter_coeff191_b,rf_filter_coeff192_a, rf_filter_coeff192_b,rf_filter_coeff193_a,
rf_filter_coeff193_b,rf_filter_coeff194_a, rf_filter_coeff194_b,rf_filter_coeff195_a, rf_filter_coeff195_b,rf_filter_coeff196_a,
rf_filter_coeff196_b,rf_filter_coeff197_a, rf_filter_coeff197_b,rf_filter_coeff198_a, rf_filter_coeff198_b,rf_filter_coeff199_a,
rf_filter_coeff199_b,rf_filter_coeff200_a, rf_filter_coeff200_b,rf_filter_coeff201_a, rf_filter_coeff201_b,rf_filter_coeff202_a,
rf_filter_coeff202_b,rf_filter_coeff203_a, rf_filter_coeff203_b,rf_filter_coeff204_a, rf_filter_coeff204_b,rf_filter_coeff205_a,
rf_filter_coeff205_b,rf_filter_coeff206_a, rf_filter_coeff206_b,rf_filter_coeff207_a, rf_filter_coeff207_b,rf_filter_coeff208_a,
rf_filter_coeff208_b,rf_filter_coeff209_a, rf_filter_coeff209_b,rf_filter_coeff210_a, rf_filter_coeff210_b,rf_filter_coeff211_a,
rf_filter_coeff211_b,rf_filter_coeff212_a, rf_filter_coeff212_b,rf_filter_coeff213_a, rf_filter_coeff213_b,rf_filter_coeff214_a,
rf_filter_coeff214_b,rf_filter_coeff215_a, rf_filter_coeff215_b,rf_filter_coeff216_a, rf_filter_coeff216_b,rf_filter_coeff217_a,
rf_filter_coeff217_b,rf_filter_coeff218_a, rf_filter_coeff218_b,rf_filter_coeff219_a, rf_filter_coeff219_b,rf_filter_coeff220_a,
rf_filter_coeff220_b,rf_filter_coeff221_a, rf_filter_coeff221_b,rf_filter_coeff222_a, rf_filter_coeff222_b,rf_filter_coeff223_a,
rf_filter_coeff223_b,rf_filter_coeff224_a, rf_filter_coeff224_b,rf_filter_coeff225_a, rf_filter_coeff225_b,rf_filter_coeff226_a,
rf_filter_coeff226_b,rf_filter_coeff227_a, rf_filter_coeff227_b,rf_filter_coeff228_a, rf_filter_coeff228_b,rf_filter_coeff229_a,
rf_filter_coeff229_b,rf_filter_coeff230_a, rf_filter_coeff230_b,rf_filter_coeff231_a, rf_filter_coeff231_b,rf_filter_coeff232_a,
rf_filter_coeff232_b,rf_filter_coeff233_a, rf_filter_coeff233_b,rf_filter_coeff234_a, rf_filter_coeff234_b,rf_filter_coeff235_a,
rf_filter_coeff235_b,rf_filter_coeff236_a, rf_filter_coeff236_b,rf_filter_coeff237_a, rf_filter_coeff237_b,rf_filter_coeff238_a,
rf_filter_coeff238_b,rf_filter_coeff239_a, rf_filter_coeff239_b,rf_filter_coeff240_a, rf_filter_coeff240_b,rf_filter_coeff241_a,
rf_filter_coeff241_b,rf_filter_coeff242_a, rf_filter_coeff242_b,rf_filter_coeff243_a, rf_filter_coeff243_b,rf_filter_coeff244_a,
rf_filter_coeff244_b,rf_filter_coeff245_a, rf_filter_coeff245_b,rf_filter_coeff246_a, rf_filter_coeff246_b,rf_filter_coeff247_a,
rf_filter_coeff247_b,rf_filter_coeff248_a, rf_filter_coeff248_b,rf_filter_coeff249_a, rf_filter_coeff249_b,rf_filter_coeff250_a,
rf_filter_coeff250_b,rf_filter_coeff251_a, rf_filter_coeff251_b,rf_filter_coeff252_a, rf_filter_coeff252_b,rf_filter_coeff253_a,
rf_filter_coeff253_b,rf_filter_coeff254_a, rf_filter_coeff254_b,rf_filter_coeff255_a, rf_filter_coeff255_b,rf_filter_coeff256_a,
rf_filter_coeff256_b,rf_filter_coeff257_a, rf_filter_coeff257_b,rf_filter_coeff258_a, rf_filter_coeff258_b,rf_filter_coeff259_a,
rf_filter_coeff259_b,rf_filter_coeff260_a, rf_filter_coeff260_b,rf_filter_coeff261_a, rf_filter_coeff261_b,rf_filter_coeff262_a,
rf_filter_coeff262_b,rf_filter_coeff263_a, rf_filter_coeff263_b,rf_filter_coeff264_a, rf_filter_coeff264_b,rf_filter_coeff265_a,
rf_filter_coeff265_b,rf_filter_coeff266_a, rf_filter_coeff266_b,rf_filter_coeff267_a, rf_filter_coeff267_b,rf_filter_coeff268_a,
rf_filter_coeff268_b,rf_filter_coeff269_a, rf_filter_coeff269_b,rf_filter_coeff270_a, rf_filter_coeff270_b,rf_filter_coeff271_a,
rf_filter_coeff271_b,rf_filter_coeff272_a, rf_filter_coeff272_b,rf_filter_coeff273_a, rf_filter_coeff273_b,rf_filter_coeff274_a,
rf_filter_coeff274_b,rf_filter_coeff275_a, rf_filter_coeff275_b,rf_filter_coeff276_a, rf_filter_coeff276_b,rf_filter_coeff277_a,
rf_filter_coeff277_b,rf_filter_coeff278_a, rf_filter_coeff278_b,rf_filter_coeff279_a, rf_filter_coeff279_b,rf_filter_coeff280_a,
rf_filter_coeff280_b,rf_filter_coeff281_a, rf_filter_coeff281_b,rf_filter_coeff282_a, rf_filter_coeff282_b,rf_filter_coeff283_a,
rf_filter_coeff283_b,rf_filter_coeff284_a, rf_filter_coeff284_b,rf_filter_coeff285_a, rf_filter_coeff285_b,rf_filter_coeff286_a,
rf_filter_coeff286_b,rf_filter_coeff287_a, rf_filter_coeff287_b,rf_filter_coeff288_a, rf_filter_coeff288_b,rf_filter_coeff289_a,
rf_filter_coeff289_b,rf_filter_coeff290_a, rf_filter_coeff290_b,rf_filter_coeff291_a, rf_filter_coeff291_b,rf_filter_coeff292_a,
rf_filter_coeff292_b,rf_filter_coeff293_a, rf_filter_coeff293_b,rf_filter_coeff294_a, rf_filter_coeff294_b,rf_filter_coeff295_a,
rf_filter_coeff295_b,rf_filter_coeff296_a, rf_filter_coeff296_b,rf_filter_coeff297_a, rf_filter_coeff297_b,rf_filter_coeff298_a,
rf_filter_coeff298_b,rf_filter_coeff299_a, rf_filter_coeff299_b,rf_filter_coeff300_a, rf_filter_coeff300_b,rf_filter_coeff301_a,
rf_filter_coeff301_b,rf_filter_coeff302_a, rf_filter_coeff302_b,rf_filter_coeff303_a, rf_filter_coeff303_b,rf_filter_coeff304_a,
rf_filter_coeff304_b,rf_filter_coeff305_a, rf_filter_coeff305_b,rf_filter_coeff306_a, rf_filter_coeff306_b,rf_filter_coeff307_a,
rf_filter_coeff307_b,rf_filter_coeff308_a, rf_filter_coeff308_b,rf_filter_coeff309_a, rf_filter_coeff309_b,rf_filter_coeff310_a,
rf_filter_coeff310_b,rf_filter_coeff311_a, rf_filter_coeff311_b,rf_filter_coeff312_a, rf_filter_coeff312_b,rf_filter_coeff313_a,
rf_filter_coeff313_b,rf_filter_coeff314_a, rf_filter_coeff314_b,rf_filter_coeff315_a, rf_filter_coeff315_b,rf_filter_coeff316_a,
rf_filter_coeff316_b,rf_filter_coeff317_a, rf_filter_coeff317_b,rf_filter_coeff318_a, rf_filter_coeff318_b,rf_filter_coeff319_a,
rf_filter_coeff319_b,rf_filter_coeff320_a, rf_filter_coeff320_b,rf_filter_coeff321_a, rf_filter_coeff321_b,rf_filter_coeff322_a,
rf_filter_coeff322_b,rf_filter_coeff323_a, rf_filter_coeff323_b,rf_filter_coeff324_a, rf_filter_coeff324_b,rf_filter_coeff325_a,
rf_filter_coeff325_b,rf_filter_coeff326_a, rf_filter_coeff326_b,rf_filter_coeff327_a, rf_filter_coeff327_b,rf_filter_coeff328_a,
rf_filter_coeff328_b,rf_filter_coeff329_a, rf_filter_coeff329_b,rf_filter_coeff330_a, rf_filter_coeff330_b,rf_filter_coeff331_a,
rf_filter_coeff331_b,rf_filter_coeff332_a, rf_filter_coeff332_b,rf_filter_coeff333_a, rf_filter_coeff333_b,rf_filter_coeff334_a,
rf_filter_coeff334_b,rf_filter_coeff335_a, rf_filter_coeff335_b,rf_filter_coeff336_a, rf_filter_coeff336_b,rf_filter_coeff337_a,
rf_filter_coeff337_b,rf_filter_coeff338_a, rf_filter_coeff338_b,rf_filter_coeff339_a, rf_filter_coeff339_b,rf_filter_coeff340_a,
rf_filter_coeff340_b,rf_filter_coeff341_a, rf_filter_coeff341_b,rf_filter_coeff342_a, rf_filter_coeff342_b,rf_filter_coeff343_a,
rf_filter_coeff343_b,rf_filter_coeff344_a, rf_filter_coeff344_b,rf_filter_coeff345_a, rf_filter_coeff345_b,rf_filter_coeff346_a,
rf_filter_coeff346_b,rf_filter_coeff347_a, rf_filter_coeff347_b,rf_filter_coeff348_a, rf_filter_coeff348_b,rf_filter_coeff349_a,
rf_filter_coeff349_b,rf_filter_coeff350_a, rf_filter_coeff350_b,rf_filter_coeff351_a, rf_filter_coeff351_b,rf_filter_coeff352_a,
rf_filter_coeff352_b,rf_filter_coeff353_a, rf_filter_coeff353_b,rf_filter_coeff354_a, rf_filter_coeff354_b,rf_filter_coeff355_a,
rf_filter_coeff355_b,rf_filter_coeff356_a, rf_filter_coeff356_b,rf_filter_coeff357_a, rf_filter_coeff357_b,rf_filter_coeff358_a,
rf_filter_coeff358_b,rf_filter_coeff359_a, rf_filter_coeff359_b,rf_filter_coeff360_a, rf_filter_coeff360_b,rf_filter_coeff361_a,
rf_filter_coeff361_b,rf_filter_coeff362_a, rf_filter_coeff362_b,rf_filter_coeff363_a, rf_filter_coeff363_b,rf_filter_coeff364_a,
rf_filter_coeff364_b,rf_filter_coeff365_a, rf_filter_coeff365_b,rf_filter_coeff366_a, rf_filter_coeff366_b,rf_filter_coeff367_a,
rf_filter_coeff367_b,rf_filter_coeff368_a, rf_filter_coeff368_b,rf_filter_coeff369_a, rf_filter_coeff369_b,rf_filter_coeff370_a,
rf_filter_coeff370_b,rf_filter_coeff371_a, rf_filter_coeff371_b,rf_filter_coeff372_a, rf_filter_coeff372_b,rf_filter_coeff373_a,
rf_filter_coeff373_b,rf_filter_coeff374_a, rf_filter_coeff374_b,rf_filter_coeff375_a, rf_filter_coeff375_b,rf_filter_coeff376_a,
rf_filter_coeff376_b,rf_filter_coeff377_a, rf_filter_coeff377_b,rf_filter_coeff378_a, rf_filter_coeff378_b,rf_filter_coeff379_a,
rf_filter_coeff379_b,rf_filter_coeff380_a, rf_filter_coeff380_b,rf_filter_coeff381_a, rf_filter_coeff381_b,rf_filter_coeff382_a,
rf_filter_coeff382_b,rf_filter_coeff383_a, rf_filter_coeff383_b,rf_filter_coeff384_a, rf_filter_coeff384_b,rf_filter_coeff385_a,
rf_filter_coeff385_b,rf_filter_coeff386_a, rf_filter_coeff386_b,rf_filter_coeff387_a, rf_filter_coeff387_b,rf_filter_coeff388_a,
rf_filter_coeff388_b,rf_filter_coeff389_a, rf_filter_coeff389_b,rf_filter_coeff390_a, rf_filter_coeff390_b,rf_filter_coeff391_a,
rf_filter_coeff391_b,rf_filter_coeff392_a, rf_filter_coeff392_b,rf_filter_coeff393_a, rf_filter_coeff393_b,rf_filter_coeff394_a,
rf_filter_coeff394_b,rf_filter_coeff395_a, rf_filter_coeff395_b,rf_filter_coeff396_a, rf_filter_coeff396_b,rf_filter_coeff397_a,
rf_filter_coeff397_b,rf_filter_coeff398_a, rf_filter_coeff398_b,rf_filter_coeff399_a, rf_filter_coeff399_b,rf_filter_coeff400_a,
rf_filter_coeff400_b,rf_filter_coeff401_a, rf_filter_coeff401_b,rf_filter_coeff402_a, rf_filter_coeff402_b,rf_filter_coeff403_a,
rf_filter_coeff403_b,rf_filter_coeff404_a, rf_filter_coeff404_b,rf_filter_coeff405_a, rf_filter_coeff405_b,rf_filter_coeff406_a,
rf_filter_coeff406_b,rf_filter_coeff407_a, rf_filter_coeff407_b,rf_filter_coeff408_a, rf_filter_coeff408_b,rf_filter_coeff409_a,
rf_filter_coeff409_b,rf_filter_coeff410_a, rf_filter_coeff410_b,rf_filter_coeff411_a, rf_filter_coeff411_b,rf_filter_coeff412_a,
rf_filter_coeff412_b,rf_filter_coeff413_a, rf_filter_coeff413_b,rf_filter_coeff414_a, rf_filter_coeff414_b,rf_filter_coeff415_a,
rf_filter_coeff415_b,rf_filter_coeff416_a, rf_filter_coeff416_b,rf_filter_coeff417_a, rf_filter_coeff417_b,rf_filter_coeff418_a,
rf_filter_coeff418_b,rf_filter_coeff419_a, rf_filter_coeff419_b,rf_filter_coeff420_a, rf_filter_coeff420_b,rf_filter_coeff421_a,
rf_filter_coeff421_b,rf_filter_coeff422_a, rf_filter_coeff422_b,rf_filter_coeff423_a, rf_filter_coeff423_b,rf_filter_coeff424_a,
rf_filter_coeff424_b,rf_filter_coeff425_a, rf_filter_coeff425_b,rf_filter_coeff426_a, rf_filter_coeff426_b,rf_filter_coeff427_a,
rf_filter_coeff427_b,rf_filter_coeff428_a, rf_filter_coeff428_b,rf_filter_coeff429_a, rf_filter_coeff429_b,rf_filter_coeff430_a,
rf_filter_coeff430_b,rf_filter_coeff431_a, rf_filter_coeff431_b,rf_filter_coeff432_a, rf_filter_coeff432_b,rf_filter_coeff433_a,
rf_filter_coeff433_b,rf_filter_coeff434_a, rf_filter_coeff434_b,rf_filter_coeff435_a, rf_filter_coeff435_b,rf_filter_coeff436_a,

```
rf_filter_coeff436_b,rf_filter_coeff437_a, rf_filter_coeff437_b,rf_filter_coeff438_a, rf_filter_coeff438_b,rf_filter_coeff439_a,
rf_filter_coeff439_b,rf_filter_coeff440_a, rf_filter_coeff440_b,rf_filter_coeff441_a, rf_filter_coeff441_b,rf_filter_coeff442_a,
rf_filter_coeff442_b,rf_filter_coeff443_a, rf_filter_coeff443_b,rf_filter_coeff444_a, rf_filter_coeff444_b,rf_filter_coeff445_a,
rf_filter_coeff445_b,rf_filter_coeff446_a, rf_filter_coeff446_b,rf_filter_coeff447_a, rf_filter_coeff447_b,rf_filter_coeff448_a,
rf_filter_coeff448_b,rf_filter_coeff449_a, rf_filter_coeff449_b,rf_filter_coeff450_a, rf_filter_coeff450_b,rf_filter_coeff451_a,
rf_filter_coeff451_b,rf_filter_coeff452_a, rf_filter_coeff452_b,rf_filter_coeff453_a, rf_filter_coeff453_b,rf_filter_coeff454_a,
rf_filter_coeff454_b,rf_filter_coeff455_a, rf_filter_coeff455_b,rf_filter_coeff456_a, rf_filter_coeff456_b,rf_filter_coeff457_a,
rf_filter_coeff457_b,rf_filter_coeff458_a, rf_filter_coeff458_b,rf_filter_coeff459_a, rf_filter_coeff459_b,rf_filter_coeff460_a,
rf_filter_coeff460_b,rf_filter_coeff461_a, rf_filter_coeff461_b,rf_filter_coeff462_a, rf_filter_coeff462_b,rf_filter_coeff463_a,
rf_filter_coeff463_b,rf_filter_coeff464_a, rf_filter_coeff464_b,rf_filter_coeff465_a, rf_filter_coeff465_b,rf_filter_coeff466_a,
rf_filter_coeff466_b,rf_filter_coeff467_a, rf_filter_coeff467_b,rf_filter_coeff468_a, rf_filter_coeff468_b,rf_filter_coeff469_a,
rf_filter_coeff469_b,rf_filter_coeff470_a, rf_filter_coeff470_b,rf_filter_coeff471_a, rf_filter_coeff471_b,rf_filter_coeff472_a,
rf_filter_coeff472_b,rf_filter_coeff473_a, rf_filter_coeff473_b,rf_filter_coeff474_a, rf_filter_coeff474_b,rf_filter_coeff475_a,
rf_filter_coeff475_b,rf_filter_coeff476_a, rf_filter_coeff476_b,rf_filter_coeff477_a, rf_filter_coeff477_b,rf_filter_coeff478_a,
rf_filter_coeff478_b,rf_filter_coeff479_a, rf_filter_coeff479_b,rf_filter_coeff480_a, rf_filter_coeff480_b,rf_filter_coeff481_a,
rf_filter_coeff481_b,rf_filter_coeff482_a, rf_filter_coeff482_b,rf_filter_coeff483_a, rf_filter_coeff483_b,rf_filter_coeff484_a,
rf_filter_coeff484_b,rf_filter_coeff485_a, rf_filter_coeff485_b,rf_filter_coeff486_a, rf_filter_coeff486_b,rf_filter_coeff487_a,
rf_filter_coeff487_b,rf_filter_coeff488_a, rf_filter_coeff488_b,rf_filter_coeff489_a, rf_filter_coeff489_b,rf_filter_coeff490_a,
rf_filter_coeff490_b,rf_filter_coeff491_a, rf_filter_coeff491_b,rf_filter_coeff492_a, rf_filter_coeff492_b,rf_filter_coeff493_a,
rf_filter_coeff493_b,rf_filter_coeff494_a, rf_filter_coeff494_b,rf_filter_coeff495_a, rf_filter_coeff495_b,rf_filter_coeff496_a,
rf_filter_coeff496_b,rf_filter_coeff497_a, rf_filter_coeff497_b,rf_filter_coeff498_a, rf_filter_coeff498_b,rf_filter_coeff499_a,
rf_filter_coeff499_b,rf_filter_coeff500_a, rf_filter_coeff500_b,rf_filter_coeff501_a, rf_filter_coeff501_b,rf_filter_coeff502_a,
rf_filter_coeff502_b,rf_filter_coeff503_a, rf_filter_coeff503_b,rf_filter_coeff504_a, rf_filter_coeff504_b,rf_filter_coeff505_a,
rf_filter_coeff505_b,rf_filter_coeff506_a, rf_filter_coeff506_b,rf_filter_coeff507_a, rf_filter_coeff507_b,rf_filter_coeff508_a,
rf_filter_coeff508_b,rf_filter_coeff509_a, rf_filter_coeff509_b,rf_filter_coeff510_a, rf_filter_coeff510_b,rf_filter_coeff511_a,
rf_filter_coeff511_b;

    always @(posedge clk or negedge rst_n)

        if (~rst_n)
        begin
            rf_i2si_bist_en <= 1'h0;
            rf_filter_shift <= 3'b000;// Nominal Shift Value is 12
            rf_filter_clip_en <= 1'h1;
            rf_i2si_en <= 1'h1;
            trig_fifo_overrun <= 1'h0; //NA?
            trig_fifo_underrun <= 1'h0; //NA?
            trig_filter_ovf_flag_clear <= 1'h0;
            rf_i2si_bist_incr <= 8'h010;
            rf_i2si_bist_start_val_a <= 8'h80;
            rf_i2si_bist_start_val_b <= 4'h0;
            rf_i2si_bist_upper_limit_a <= 8'h7f;
            rf_i2si_bist_upper_limit_b <= 4'hf;

            rf_filter_coeff0_a <= 8'b00010000;     //                        When Shift Value is 12
            rf_filter_coeff0_b <= 8'b00000000;     //                        1 is represented by 16'h0100;
            rf_filter_coeff1_a <= 8'h000;
            rf_filter_coeff1_b <= 8'h000;
            rf_filter_coeff2_a <= 8'h000;
            rf_filter_coeff2_b <= 8'h000;
            rf_filter_coeff3_a <= 8'h000;
            rf_filter_coeff3_b <= 8'h000;
            rf_filter_coeff4_a <= 8'h000;
            rf_filter_coeff4_b <= 8'h000;
            rf_filter_coeff5_a <= 8'h000;
            rf_filter_coeff5_b <= 8'h000;
            rf_filter_coeff6_a <= 8'h000;
            rf_filter_coeff6_b <= 8'h000;
            rf_filter_coeff7_a <= 8'h000;
            rf_filter_coeff7_b <= 8'h000;
            rf_filter_coeff8_a <= 8'h000;
            rf_filter_coeff8_b <= 8'h000;
            rf_filter_coeff9_a <= 8'h000;
            rf_filter_coeff9_b <= 8'h000;
            rf_filter_coeff10_a <= 8'h000;
            rf_filter_coeff10_b <= 8'h000;
            rf_filter_coeff11_a <= 8'h000;
            rf_filter_coeff11_b <= 8'h000;
            rf_filter_coeff12_a <= 8'h000;
            rf_filter_coeff12_b <= 8'h000;
            rf_filter_coeff13_a <= 8'h000;
            rf_filter_coeff13_b <= 8'h000;
            rf_filter_coeff14_a <= 8'h000;
            rf_filter_coeff14_b <= 8'h000;
            rf_filter_coeff15_a <= 8'h000;
            rf_filter_coeff15_b <= 8'h000;
            rf_filter_coeff16_a <= 8'h000;
            rf_filter_coeff16_b <= 8'h000;
            rf_filter_coeff17_a <= 8'h000;
            rf_filter_coeff17_b <= 8'h000;
            rf_filter_coeff18_a <= 8'h000;
            rf_filter_coeff18_b <= 8'h000;
            rf_filter_coeff19_a <= 8'h000;
            rf_filter_coeff19_b <= 8'h000;
            rf_filter_coeff20_a <= 8'h000;
            rf_filter_coeff20_b <= 8'h000;
            rf_filter_coeff21_a <= 8'h000;
            rf_filter_coeff21_b <= 8'h000;
            rf_filter_coeff22_a <= 8'h000;
            rf_filter_coeff22_b <= 8'h000;
            rf_filter_coeff23_a <= 8'h000;
            rf_filter_coeff23_b <= 8'h000;
            rf_filter_coeff24_a <= 8'h000;
            rf_filter_coeff24_b <= 8'h000;
            rf_filter_coeff25_a <= 8'h000;
```

```
rf_filter_coeff25_b <= 8'h000;
rf_filter_coeff26_a <= 8'h000;
rf_filter_coeff26_b <= 8'h000;
rf_filter_coeff27_a <= 8'h000;
rf_filter_coeff27_b <= 8'h000;
rf_filter_coeff28_a <= 8'h000;
rf_filter_coeff28_b <= 8'h000;
rf_filter_coeff29_a <= 8'h000;
rf_filter_coeff29_b <= 8'h000;
rf_filter_coeff30_a <= 8'h000;
rf_filter_coeff30_b <= 8'h000;
rf_filter_coeff31_a <= 8'h000;
rf_filter_coeff31_b <= 8'h000;
rf_filter_coeff32_a <= 8'h000;
rf_filter_coeff32_b <= 8'h000;
rf_filter_coeff33_a <= 8'h000;
rf_filter_coeff33_b <= 8'h000;
rf_filter_coeff34_a <= 8'h000;
rf_filter_coeff34_b <= 8'h000;
rf_filter_coeff35_a <= 8'h000;
rf_filter_coeff35_b <= 8'h000;
rf_filter_coeff36_a <= 8'h000;
rf_filter_coeff36_b <= 8'h000;
rf_filter_coeff37_a <= 8'h000;
rf_filter_coeff37_b <= 8'h000;
rf_filter_coeff38_a <= 8'h000;
rf_filter_coeff38_b <= 8'h000;
rf_filter_coeff39_a <= 8'h000;
rf_filter_coeff39_b <= 8'h000;
rf_filter_coeff40_a <= 8'h000;
rf_filter_coeff40_b <= 8'h000;
rf_filter_coeff41_a <= 8'h000;
rf_filter_coeff41_b <= 8'h000;
rf_filter_coeff42_a <= 8'h000;
rf_filter_coeff42_b <= 8'h000;
rf_filter_coeff43_a <= 8'h000;
rf_filter_coeff43_b <= 8'h000;
rf_filter_coeff44_a <= 8'h000;
rf_filter_coeff44_b <= 8'h000;
rf_filter_coeff45_a <= 8'h000;
rf_filter_coeff45_b <= 8'h000;
rf_filter_coeff46_a <= 8'h000;
rf_filter_coeff46_b <= 8'h000;
rf_filter_coeff47_a <= 8'h000;
rf_filter_coeff47_b <= 8'h000;
rf_filter_coeff48_a <= 8'h000;
rf_filter_coeff48_b <= 8'h000;
rf_filter_coeff49_a <= 8'h000;
rf_filter_coeff49_b <= 8'h000;
rf_filter_coeff50_a <= 8'h000;
rf_filter_coeff50_b <= 8'h000;
rf_filter_coeff51_a <= 8'h000;
rf_filter_coeff51_b <= 8'h000;
rf_filter_coeff52_a <= 8'h000;
rf_filter_coeff52_b <= 8'h000;
rf_filter_coeff53_a <= 8'h000;
rf_filter_coeff53_b <= 8'h000;
rf_filter_coeff54_a <= 8'h000;
rf_filter_coeff54_b <= 8'h000;
rf_filter_coeff55_a <= 8'h000;
rf_filter_coeff55_b <= 8'h000;
rf_filter_coeff56_a <= 8'h000;
rf_filter_coeff56_b <= 8'h000;
rf_filter_coeff57_a <= 8'h000;
rf_filter_coeff57_b <= 8'h000;
rf_filter_coeff58_a <= 8'h000;
rf_filter_coeff58_b <= 8'h000;
rf_filter_coeff59_a <= 8'h000;
rf_filter_coeff59_b <= 8'h000;
rf_filter_coeff60_a <= 8'h000;
rf_filter_coeff60_b <= 8'h000;
rf_filter_coeff61_a <= 8'h000;
rf_filter_coeff61_b <= 8'h000;
rf_filter_coeff62_a <= 8'h000;
rf_filter_coeff62_b <= 8'h000;
rf_filter_coeff63_a <= 8'h000;
rf_filter_coeff63_b <= 8'h000;
rf_filter_coeff64_a <= 8'h000;
rf_filter_coeff64_b <= 8'h000;
rf_filter_coeff65_a <= 8'h000;
rf_filter_coeff65_b <= 8'h000;
rf_filter_coeff66_a <= 8'h000;
rf_filter_coeff66_b <= 8'h000;
rf_filter_coeff67_a <= 8'h000;
rf_filter_coeff67_b <= 8'h000;
rf_filter_coeff68_a <= 8'h000;
rf_filter_coeff68_b <= 8'h000;
rf_filter_coeff69_a <= 8'h000;
rf_filter_coeff69_b <= 8'h000;
rf_filter_coeff70_a <= 8'h000;
rf_filter_coeff70_b <= 8'h000;
rf_filter_coeff71_a <= 8'h000;
rf_filter_coeff71_b <= 8'h000;
rf_filter_coeff72_a <= 8'h000;
rf_filter_coeff72_b <= 8'h000;
```

```
rf_filter_coeff73_a <= 8'h000;
rf_filter_coeff73_b <= 8'h000;
rf_filter_coeff74_a <= 8'h000;
rf_filter_coeff74_b <= 8'h000;
rf_filter_coeff75_a <= 8'h000;
rf_filter_coeff75_b <= 8'h000;
rf_filter_coeff76_a <= 8'h000;
rf_filter_coeff76_b <= 8'h000;
rf_filter_coeff77_a <= 8'h000;
rf_filter_coeff77_b <= 8'h000;
rf_filter_coeff78_a <= 8'h000;
rf_filter_coeff78_b <= 8'h000;
rf_filter_coeff79_a <= 8'h000;
rf_filter_coeff79_b <= 8'h000;
rf_filter_coeff80_a <= 8'h000;
rf_filter_coeff80_b <= 8'h000;
rf_filter_coeff81_a <= 8'h000;
rf_filter_coeff81_b <= 8'h000;
rf_filter_coeff82_a <= 8'h000;
rf_filter_coeff82_b <= 8'h000;
rf_filter_coeff83_a <= 8'h000;
rf_filter_coeff83_b <= 8'h000;
rf_filter_coeff84_a <= 8'h000;
rf_filter_coeff84_b <= 8'h000;
rf_filter_coeff85_a <= 8'h000;
rf_filter_coeff85_b <= 8'h000;
rf_filter_coeff86_a <= 8'h000;
rf_filter_coeff86_b <= 8'h000;
rf_filter_coeff87_a <= 8'h000;
rf_filter_coeff87_b <= 8'h000;
rf_filter_coeff88_a <= 8'h000;
rf_filter_coeff88_b <= 8'h000;
rf_filter_coeff89_a <= 8'h000;
rf_filter_coeff89_b <= 8'h000;
rf_filter_coeff90_a <= 8'h000;
rf_filter_coeff90_b <= 8'h000;
rf_filter_coeff91_a <= 8'h000;
rf_filter_coeff91_b <= 8'h000;
rf_filter_coeff92_a <= 8'h000;
rf_filter_coeff92_b <= 8'h000;
rf_filter_coeff93_a <= 8'h000;
rf_filter_coeff93_b <= 8'h000;
rf_filter_coeff94_a <= 8'h000;
rf_filter_coeff94_b <= 8'h000;
rf_filter_coeff95_a <= 8'h000;
rf_filter_coeff95_b <= 8'h000;
rf_filter_coeff96_a <= 8'h000;
rf_filter_coeff96_b <= 8'h000;
rf_filter_coeff97_a <= 8'h000;
rf_filter_coeff97_b <= 8'h000;
rf_filter_coeff98_a <= 8'h000;
rf_filter_coeff98_b <= 8'h000;
rf_filter_coeff99_a <= 8'h000;
rf_filter_coeff99_b <= 8'h000;
rf_filter_coeff100_a <= 8'h000;
rf_filter_coeff100_b <= 8'h000;
rf_filter_coeff101_a <= 8'h000;
rf_filter_coeff101_b <= 8'h000;
rf_filter_coeff102_a <= 8'h000;
rf_filter_coeff102_b <= 8'h000;
rf_filter_coeff103_a <= 8'h000;
rf_filter_coeff103_b <= 8'h000;
rf_filter_coeff104_a <= 8'h000;
rf_filter_coeff104_b <= 8'h000;
rf_filter_coeff105_a <= 8'h000;
rf_filter_coeff105_b <= 8'h000;
rf_filter_coeff106_a <= 8'h000;
rf_filter_coeff106_b <= 8'h000;
rf_filter_coeff107_a <= 8'h000;
rf_filter_coeff107_b <= 8'h000;
rf_filter_coeff108_a <= 8'h000;
rf_filter_coeff108_b <= 8'h000;
rf_filter_coeff109_a <= 8'h000;
rf_filter_coeff109_b <= 8'h000;
rf_filter_coeff110_a <= 8'h000;
rf_filter_coeff110_b <= 8'h000;
rf_filter_coeff111_a <= 8'h000;
rf_filter_coeff111_b <= 8'h000;
rf_filter_coeff112_a <= 8'h000;
rf_filter_coeff112_b <= 8'h000;
rf_filter_coeff113_a <= 8'h000;
rf_filter_coeff113_b <= 8'h000;
rf_filter_coeff114_a <= 8'h000;
rf_filter_coeff114_b <= 8'h000;
rf_filter_coeff115_a <= 8'h000;
rf_filter_coeff115_b <= 8'h000;
rf_filter_coeff116_a <= 8'h000;
rf_filter_coeff116_b <= 8'h000;
rf_filter_coeff117_a <= 8'h000;
rf_filter_coeff117_b <= 8'h000;
rf_filter_coeff118_a <= 8'h000;
rf_filter_coeff118_b <= 8'h000;
rf_filter_coeff119_a <= 8'h000;
rf_filter_coeff119_b <= 8'h000;
rf_filter_coeff120_a <= 8'h000;
```

```
rf_filter_coeff120_b <= 8'h000;
rf_filter_coeff121_a <= 8'h000;
rf_filter_coeff121_b <= 8'h000;
rf_filter_coeff122_a <= 8'h000;
rf_filter_coeff122_b <= 8'h000;
rf_filter_coeff123_a <= 8'h000;
rf_filter_coeff123_b <= 8'h000;
rf_filter_coeff124_a <= 8'h000;
rf_filter_coeff124_b <= 8'h000;
rf_filter_coeff125_a <= 8'h000;
rf_filter_coeff125_b <= 8'h000;
rf_filter_coeff126_a <= 8'h000;
rf_filter_coeff126_b <= 8'h000;
rf_filter_coeff127_a <= 8'h000;
rf_filter_coeff127_b <= 8'h000;
rf_filter_coeff128_a <= 8'h000;
rf_filter_coeff128_b <= 8'h000;
rf_filter_coeff129_a <= 8'h000;
rf_filter_coeff129_b <= 8'h000;
rf_filter_coeff130_a <= 8'h000;
rf_filter_coeff130_b <= 8'h000;
rf_filter_coeff131_a <= 8'h000;
rf_filter_coeff131_b <= 8'h000;
rf_filter_coeff132_a <= 8'h000;
rf_filter_coeff132_b <= 8'h000;
rf_filter_coeff133_a <= 8'h000;
rf_filter_coeff133_b <= 8'h000;
rf_filter_coeff134_a <= 8'h000;
rf_filter_coeff134_b <= 8'h000;
rf_filter_coeff135_a <= 8'h000;
rf_filter_coeff135_b <= 8'h000;
rf_filter_coeff136_a <= 8'h000;
rf_filter_coeff136_b <= 8'h000;
rf_filter_coeff137_a <= 8'h000;
rf_filter_coeff137_b <= 8'h000;
rf_filter_coeff138_a <= 8'h000;
rf_filter_coeff138_b <= 8'h000;
rf_filter_coeff139_a <= 8'h000;
rf_filter_coeff139_b <= 8'h000;
rf_filter_coeff140_a <= 8'h000;
rf_filter_coeff140_b <= 8'h000;
rf_filter_coeff141_a <= 8'h000;
rf_filter_coeff141_b <= 8'h000;
rf_filter_coeff142_a <= 8'h000;
rf_filter_coeff142_b <= 8'h000;
rf_filter_coeff143_a <= 8'h000;
rf_filter_coeff143_b <= 8'h000;
rf_filter_coeff144_a <= 8'h000;
rf_filter_coeff144_b <= 8'h000;
rf_filter_coeff145_a <= 8'h000;
rf_filter_coeff145_b <= 8'h000;
rf_filter_coeff146_a <= 8'h000;
rf_filter_coeff146_b <= 8'h000;
rf_filter_coeff147_a <= 8'h000;
rf_filter_coeff147_b <= 8'h000;
rf_filter_coeff148_a <= 8'h000;
rf_filter_coeff148_b <= 8'h000;
rf_filter_coeff149_a <= 8'h000;
rf_filter_coeff149_b <= 8'h000;
rf_filter_coeff150_a <= 8'h000;
rf_filter_coeff150_b <= 8'h000;
rf_filter_coeff151_a <= 8'h000;
rf_filter_coeff151_b <= 8'h000;
rf_filter_coeff152_a <= 8'h000;
rf_filter_coeff152_b <= 8'h000;
rf_filter_coeff153_a <= 8'h000;
rf_filter_coeff153_b <= 8'h000;
rf_filter_coeff154_a <= 8'h000;
rf_filter_coeff154_b <= 8'h000;
rf_filter_coeff155_a <= 8'h000;
rf_filter_coeff155_b <= 8'h000;
rf_filter_coeff156_a <= 8'h000;
rf_filter_coeff156_b <= 8'h000;
rf_filter_coeff157_a <= 8'h000;
rf_filter_coeff157_b <= 8'h000;
rf_filter_coeff158_a <= 8'h000;
rf_filter_coeff158_b <= 8'h000;
rf_filter_coeff159_a <= 8'h000;
rf_filter_coeff159_b <= 8'h000;
rf_filter_coeff160_a <= 8'h000;
rf_filter_coeff160_b <= 8'h000;
rf_filter_coeff161_a <= 8'h000;
rf_filter_coeff161_b <= 8'h000;
rf_filter_coeff162_a <= 8'h000;
rf_filter_coeff162_b <= 8'h000;
rf_filter_coeff163_a <= 8'h000;
rf_filter_coeff163_b <= 8'h000;
rf_filter_coeff164_a <= 8'h000;
rf_filter_coeff164_b <= 8'h000;
rf_filter_coeff165_a <= 8'h000;
rf_filter_coeff165_b <= 8'h000;
rf_filter_coeff166_a <= 8'h000;
rf_filter_coeff166_b <= 8'h000;
rf_filter_coeff167_a <= 8'h000;
rf_filter_coeff167_b <= 8'h000;
```

```
rf_filter_coeff168_a <= 8'h000;
rf_filter_coeff168_b <= 8'h000;
rf_filter_coeff169_a <= 8'h000;
rf_filter_coeff169_b <= 8'h000;
rf_filter_coeff170_a <= 8'h000;
rf_filter_coeff170_b <= 8'h000;
rf_filter_coeff171_a <= 8'h000;
rf_filter_coeff171_b <= 8'h000;
rf_filter_coeff172_a <= 8'h000;
rf_filter_coeff172_b <= 8'h000;
rf_filter_coeff173_a <= 8'h000;
rf_filter_coeff173_b <= 8'h000;
rf_filter_coeff174_a <= 8'h000;
rf_filter_coeff174_b <= 8'h000;
rf_filter_coeff175_a <= 8'h000;
rf_filter_coeff175_b <= 8'h000;
rf_filter_coeff176_a <= 8'h000;
rf_filter_coeff176_b <= 8'h000;
rf_filter_coeff177_a <= 8'h000;
rf_filter_coeff177_b <= 8'h000;
rf_filter_coeff178_a <= 8'h000;
rf_filter_coeff178_b <= 8'h000;
rf_filter_coeff179_a <= 8'h000;
rf_filter_coeff179_b <= 8'h000;
rf_filter_coeff180_a <= 8'h000;
rf_filter_coeff180_b <= 8'h000;
rf_filter_coeff181_a <= 8'h000;
rf_filter_coeff181_b <= 8'h000;
rf_filter_coeff182_a <= 8'h000;
rf_filter_coeff182_b <= 8'h000;
rf_filter_coeff183_a <= 8'h000;
rf_filter_coeff183_b <= 8'h000;
rf_filter_coeff184_a <= 8'h000;
rf_filter_coeff184_b <= 8'h000;
rf_filter_coeff185_a <= 8'h000;
rf_filter_coeff185_b <= 8'h000;
rf_filter_coeff186_a <= 8'h000;
rf_filter_coeff186_b <= 8'h000;
rf_filter_coeff187_a <= 8'h000;
rf_filter_coeff187_b <= 8'h000;
rf_filter_coeff188_a <= 8'h000;
rf_filter_coeff188_b <= 8'h000;
rf_filter_coeff189_a <= 8'h000;
rf_filter_coeff189_b <= 8'h000;
rf_filter_coeff190_a <= 8'h000;
rf_filter_coeff190_b <= 8'h000;
rf_filter_coeff191_a <= 8'h000;
rf_filter_coeff191_b <= 8'h000;
rf_filter_coeff192_a <= 8'h000;
rf_filter_coeff192_b <= 8'h000;
rf_filter_coeff193_a <= 8'h000;
rf_filter_coeff193_b <= 8'h000;
rf_filter_coeff194_a <= 8'h000;
rf_filter_coeff194_b <= 8'h000;
rf_filter_coeff195_a <= 8'h000;
rf_filter_coeff195_b <= 8'h000;
rf_filter_coeff196_a <= 8'h000;
rf_filter_coeff196_b <= 8'h000;
rf_filter_coeff197_a <= 8'h000;
rf_filter_coeff197_b <= 8'h000;
rf_filter_coeff198_a <= 8'h000;
rf_filter_coeff198_b <= 8'h000;
rf_filter_coeff199_a <= 8'h000;
rf_filter_coeff199_b <= 8'h000;
rf_filter_coeff200_a <= 8'h000;
rf_filter_coeff200_b <= 8'h000;
rf_filter_coeff201_a <= 8'h000;
rf_filter_coeff201_b <= 8'h000;
rf_filter_coeff202_a <= 8'h000;
rf_filter_coeff202_b <= 8'h000;
rf_filter_coeff203_a <= 8'h000;
rf_filter_coeff203_b <= 8'h000;
rf_filter_coeff204_a <= 8'h000;
rf_filter_coeff204_b <= 8'h000;
rf_filter_coeff205_a <= 8'h000;
rf_filter_coeff205_b <= 8'h000;
rf_filter_coeff206_a <= 8'h000;
rf_filter_coeff206_b <= 8'h000;
rf_filter_coeff207_a <= 8'h000;
rf_filter_coeff207_b <= 8'h000;
rf_filter_coeff208_a <= 8'h000;
rf_filter_coeff208_b <= 8'h000;
rf_filter_coeff209_a <= 8'h000;
rf_filter_coeff209_b <= 8'h000;
rf_filter_coeff210_a <= 8'h000;
rf_filter_coeff210_b <= 8'h000;
rf_filter_coeff211_a <= 8'h000;
rf_filter_coeff211_b <= 8'h000;
rf_filter_coeff212_a <= 8'h000;
rf_filter_coeff212_b <= 8'h000;
rf_filter_coeff213_a <= 8'h000;
rf_filter_coeff213_b <= 8'h000;
rf_filter_coeff214_a <= 8'h000;
rf_filter_coeff214_b <= 8'h000;
rf_filter_coeff215_a <= 8'h000;
```

```
rf_filter_coeff215_b <= 8'h000;
rf_filter_coeff216_a <= 8'h000;
rf_filter_coeff216_b <= 8'h000;
rf_filter_coeff217_a <= 8'h000;
rf_filter_coeff217_b <= 8'h000;
rf_filter_coeff218_a <= 8'h000;
rf_filter_coeff218_b <= 8'h000;
rf_filter_coeff219_a <= 8'h000;
rf_filter_coeff219_b <= 8'h000;
rf_filter_coeff220_a <= 8'h000;
rf_filter_coeff220_b <= 8'h000;
rf_filter_coeff221_a <= 8'h000;
rf_filter_coeff221_b <= 8'h000;
rf_filter_coeff222_a <= 8'h000;
rf_filter_coeff222_b <= 8'h000;
rf_filter_coeff223_a <= 8'h000;
rf_filter_coeff223_b <= 8'h000;
rf_filter_coeff224_a <= 8'h000;
rf_filter_coeff224_b <= 8'h000;
rf_filter_coeff225_a <= 8'h000;
rf_filter_coeff225_b <= 8'h000;
rf_filter_coeff226_a <= 8'h000;
rf_filter_coeff226_b <= 8'h000;
rf_filter_coeff227_a <= 8'h000;
rf_filter_coeff227_b <= 8'h000;
rf_filter_coeff228_a <= 8'h000;
rf_filter_coeff228_b <= 8'h000;
rf_filter_coeff229_a <= 8'h000;
rf_filter_coeff229_b <= 8'h000;
rf_filter_coeff230_a <= 8'h000;
rf_filter_coeff230_b <= 8'h000;
rf_filter_coeff231_a <= 8'h000;
rf_filter_coeff231_b <= 8'h000;
rf_filter_coeff232_a <= 8'h000;
rf_filter_coeff232_b <= 8'h000;
rf_filter_coeff233_a <= 8'h000;
rf_filter_coeff233_b <= 8'h000;
rf_filter_coeff234_a <= 8'h000;
rf_filter_coeff234_b <= 8'h000;
rf_filter_coeff235_a <= 8'h000;
rf_filter_coeff235_b <= 8'h000;
rf_filter_coeff236_a <= 8'h000;
rf_filter_coeff236_b <= 8'h000;
rf_filter_coeff237_a <= 8'h000;
rf_filter_coeff237_b <= 8'h000;
rf_filter_coeff238_a <= 8'h000;
rf_filter_coeff238_b <= 8'h000;
rf_filter_coeff239_a <= 8'h000;
rf_filter_coeff239_b <= 8'h000;
rf_filter_coeff240_a <= 8'h000;
rf_filter_coeff240_b <= 8'h000;
rf_filter_coeff241_a <= 8'h000;
rf_filter_coeff241_b <= 8'h000;
rf_filter_coeff242_a <= 8'h000;
rf_filter_coeff242_b <= 8'h000;
rf_filter_coeff243_a <= 8'h000;
rf_filter_coeff243_b <= 8'h000;
rf_filter_coeff244_a <= 8'h000;
rf_filter_coeff244_b <= 8'h000;
rf_filter_coeff245_a <= 8'h000;
rf_filter_coeff245_b <= 8'h000;
rf_filter_coeff246_a <= 8'h000;
rf_filter_coeff246_b <= 8'h000;
rf_filter_coeff247_a <= 8'h000;
rf_filter_coeff247_b <= 8'h000;
rf_filter_coeff248_a <= 8'h000;
rf_filter_coeff248_b <= 8'h000;
rf_filter_coeff249_a <= 8'h000;
rf_filter_coeff249_b <= 8'h000;
rf_filter_coeff250_a <= 8'h000;
rf_filter_coeff250_b <= 8'h000;
rf_filter_coeff251_a <= 8'h000;
rf_filter_coeff251_b <= 8'h000;
rf_filter_coeff252_a <= 8'h000;
rf_filter_coeff252_b <= 8'h000;
rf_filter_coeff253_a <= 8'h000;
rf_filter_coeff253_b <= 8'h000;
rf_filter_coeff254_a <= 8'h000;
rf_filter_coeff254_b <= 8'h000;
rf_filter_coeff255_a <= 8'h000;
rf_filter_coeff255_b <= 8'h000;
rf_filter_coeff256_a <= 8'h000;
rf_filter_coeff256_b <= 8'h000;
rf_filter_coeff257_a <= 8'h000;
rf_filter_coeff257_b <= 8'h000;
rf_filter_coeff258_a <= 8'h000;
rf_filter_coeff258_b <= 8'h000;
rf_filter_coeff259_a <= 8'h000;
rf_filter_coeff259_b <= 8'h000;
rf_filter_coeff260_a <= 8'h000;
rf_filter_coeff260_b <= 8'h000;
rf_filter_coeff261_a <= 8'h000;
rf_filter_coeff261_b <= 8'h000;
rf_filter_coeff262_a <= 8'h000;
rf_filter_coeff262_b <= 8'h000;
```

```
rf_filter_coeff263_a <= 8'h000;
rf_filter_coeff263_b <= 8'h000;
rf_filter_coeff264_a <= 8'h000;
rf_filter_coeff264_b <= 8'h000;
rf_filter_coeff265_a <= 8'h000;
rf_filter_coeff265_b <= 8'h000;
rf_filter_coeff266_a <= 8'h000;
rf_filter_coeff266_b <= 8'h000;
rf_filter_coeff267_a <= 8'h000;
rf_filter_coeff267_b <= 8'h000;
rf_filter_coeff268_a <= 8'h000;
rf_filter_coeff268_b <= 8'h000;
rf_filter_coeff269_a <= 8'h000;
rf_filter_coeff269_b <= 8'h000;
rf_filter_coeff270_a <= 8'h000;
rf_filter_coeff270_b <= 8'h000;
rf_filter_coeff271_a <= 8'h000;
rf_filter_coeff271_b <= 8'h000;
rf_filter_coeff272_a <= 8'h000;
rf_filter_coeff272_b <= 8'h000;
rf_filter_coeff273_a <= 8'h000;
rf_filter_coeff273_b <= 8'h000;
rf_filter_coeff274_a <= 8'h000;
rf_filter_coeff274_b <= 8'h000;
rf_filter_coeff275_a <= 8'h000;
rf_filter_coeff275_b <= 8'h000;
rf_filter_coeff276_a <= 8'h000;
rf_filter_coeff276_b <= 8'h000;
rf_filter_coeff277_a <= 8'h000;
rf_filter_coeff277_b <= 8'h000;
rf_filter_coeff278_a <= 8'h000;
rf_filter_coeff278_b <= 8'h000;
rf_filter_coeff279_a <= 8'h000;
rf_filter_coeff279_b <= 8'h000;
rf_filter_coeff280_a <= 8'h000;
rf_filter_coeff280_b <= 8'h000;
rf_filter_coeff281_a <= 8'h000;
rf_filter_coeff281_b <= 8'h000;
rf_filter_coeff282_a <= 8'h000;
rf_filter_coeff282_b <= 8'h000;
rf_filter_coeff283_a <= 8'h000;
rf_filter_coeff283_b <= 8'h000;
rf_filter_coeff284_a <= 8'h000;
rf_filter_coeff284_b <= 8'h000;
rf_filter_coeff285_a <= 8'h000;
rf_filter_coeff285_b <= 8'h000;
rf_filter_coeff286_a <= 8'h000;
rf_filter_coeff286_b <= 8'h000;
rf_filter_coeff287_a <= 8'h000;
rf_filter_coeff287_b <= 8'h000;
rf_filter_coeff288_a <= 8'h000;
rf_filter_coeff288_b <= 8'h000;
rf_filter_coeff289_a <= 8'h000;
rf_filter_coeff289_b <= 8'h000;
rf_filter_coeff290_a <= 8'h000;
rf_filter_coeff290_b <= 8'h000;
rf_filter_coeff291_a <= 8'h000;
rf_filter_coeff291_b <= 8'h000;
rf_filter_coeff292_a <= 8'h000;
rf_filter_coeff292_b <= 8'h000;
rf_filter_coeff293_a <= 8'h000;
rf_filter_coeff293_b <= 8'h000;
rf_filter_coeff294_a <= 8'h000;
rf_filter_coeff294_b <= 8'h000;
rf_filter_coeff295_a <= 8'h000;
rf_filter_coeff295_b <= 8'h000;
rf_filter_coeff296_a <= 8'h000;
rf_filter_coeff296_b <= 8'h000;
rf_filter_coeff297_a <= 8'h000;
rf_filter_coeff297_b <= 8'h000;
rf_filter_coeff298_a <= 8'h000;
rf_filter_coeff298_b <= 8'h000;
rf_filter_coeff299_a <= 8'h000;
rf_filter_coeff299_b <= 8'h000;
rf_filter_coeff300_a <= 8'h000;
rf_filter_coeff300_b <= 8'h000;
rf_filter_coeff301_a <= 8'h000;
rf_filter_coeff301_b <= 8'h000;
rf_filter_coeff302_a <= 8'h000;
rf_filter_coeff302_b <= 8'h000;
rf_filter_coeff303_a <= 8'h000;
rf_filter_coeff303_b <= 8'h000;
rf_filter_coeff304_a <= 8'h000;
rf_filter_coeff304_b <= 8'h000;
rf_filter_coeff305_a <= 8'h000;
rf_filter_coeff305_b <= 8'h000;
rf_filter_coeff306_a <= 8'h000;
rf_filter_coeff306_b <= 8'h000;
rf_filter_coeff307_a <= 8'h000;
rf_filter_coeff307_b <= 8'h000;
rf_filter_coeff308_a <= 8'h000;
rf_filter_coeff308_b <= 8'h000;
rf_filter_coeff309_a <= 8'h000;
rf_filter_coeff309_b <= 8'h000;
rf_filter_coeff310_a <= 8'h000;
```

```
rf_filter_coeff310_b <= 8'h000;
rf_filter_coeff311_a <= 8'h000;
rf_filter_coeff311_b <= 8'h000;
rf_filter_coeff312_a <= 8'h000;
rf_filter_coeff312_b <= 8'h000;
rf_filter_coeff313_a <= 8'h000;
rf_filter_coeff313_b <= 8'h000;
rf_filter_coeff314_a <= 8'h000;
rf_filter_coeff314_b <= 8'h000;
rf_filter_coeff315_a <= 8'h000;
rf_filter_coeff315_b <= 8'h000;
rf_filter_coeff316_a <= 8'h000;
rf_filter_coeff316_b <= 8'h000;
rf_filter_coeff317_a <= 8'h000;
rf_filter_coeff317_b <= 8'h000;
rf_filter_coeff318_a <= 8'h000;
rf_filter_coeff318_b <= 8'h000;
rf_filter_coeff319_a <= 8'h000;
rf_filter_coeff319_b <= 8'h000;
rf_filter_coeff320_a <= 8'h000;
rf_filter_coeff320_b <= 8'h000;
rf_filter_coeff321_a <= 8'h000;
rf_filter_coeff321_b <= 8'h000;
rf_filter_coeff322_a <= 8'h000;
rf_filter_coeff322_b <= 8'h000;
rf_filter_coeff323_a <= 8'h000;
rf_filter_coeff323_b <= 8'h000;
rf_filter_coeff324_a <= 8'h000;
rf_filter_coeff324_b <= 8'h000;
rf_filter_coeff325_a <= 8'h000;
rf_filter_coeff325_b <= 8'h000;
rf_filter_coeff326_a <= 8'h000;
rf_filter_coeff326_b <= 8'h000;
rf_filter_coeff327_a <= 8'h000;
rf_filter_coeff327_b <= 8'h000;
rf_filter_coeff328_a <= 8'h000;
rf_filter_coeff328_b <= 8'h000;
rf_filter_coeff329_a <= 8'h000;
rf_filter_coeff329_b <= 8'h000;
rf_filter_coeff330_a <= 8'h000;
rf_filter_coeff330_b <= 8'h000;
rf_filter_coeff331_a <= 8'h000;
rf_filter_coeff331_b <= 8'h000;
rf_filter_coeff332_a <= 8'h000;
rf_filter_coeff332_b <= 8'h000;
rf_filter_coeff333_a <= 8'h000;
rf_filter_coeff333_b <= 8'h000;
rf_filter_coeff334_a <= 8'h000;
rf_filter_coeff334_b <= 8'h000;
rf_filter_coeff335_a <= 8'h000;
rf_filter_coeff335_b <= 8'h000;
rf_filter_coeff336_a <= 8'h000;
rf_filter_coeff336_b <= 8'h000;
rf_filter_coeff337_a <= 8'h000;
rf_filter_coeff337_b <= 8'h000;
rf_filter_coeff338_a <= 8'h000;
rf_filter_coeff338_b <= 8'h000;
rf_filter_coeff339_a <= 8'h000;
rf_filter_coeff339_b <= 8'h000;
rf_filter_coeff340_a <= 8'h000;
rf_filter_coeff340_b <= 8'h000;
rf_filter_coeff341_a <= 8'h000;
rf_filter_coeff341_b <= 8'h000;
rf_filter_coeff342_a <= 8'h000;
rf_filter_coeff342_b <= 8'h000;
rf_filter_coeff343_a <= 8'h000;
rf_filter_coeff343_b <= 8'h000;
rf_filter_coeff344_a <= 8'h000;
rf_filter_coeff344_b <= 8'h000;
rf_filter_coeff345_a <= 8'h000;
rf_filter_coeff345_b <= 8'h000;
rf_filter_coeff346_a <= 8'h000;
rf_filter_coeff346_b <= 8'h000;
rf_filter_coeff347_a <= 8'h000;
rf_filter_coeff347_b <= 8'h000;
rf_filter_coeff348_a <= 8'h000;
rf_filter_coeff348_b <= 8'h000;
rf_filter_coeff349_a <= 8'h000;
rf_filter_coeff349_b <= 8'h000;
rf_filter_coeff350_a <= 8'h000;
rf_filter_coeff350_b <= 8'h000;
rf_filter_coeff351_a <= 8'h000;
rf_filter_coeff351_b <= 8'h000;
rf_filter_coeff352_a <= 8'h000;
rf_filter_coeff352_b <= 8'h000;
rf_filter_coeff353_a <= 8'h000;
rf_filter_coeff353_b <= 8'h000;
rf_filter_coeff354_a <= 8'h000;
rf_filter_coeff354_b <= 8'h000;
rf_filter_coeff355_a <= 8'h000;
rf_filter_coeff355_b <= 8'h000;
rf_filter_coeff356_a <= 8'h000;
rf_filter_coeff356_b <= 8'h000;
rf_filter_coeff357_a <= 8'h000;
rf_filter_coeff357_b <= 8'h000;
```

```
rf_filter_coeff358_a <= 8'h000;
rf_filter_coeff358_b <= 8'h000;
rf_filter_coeff359_a <= 8'h000;
rf_filter_coeff359_b <= 8'h000;
rf_filter_coeff360_a <= 8'h000;
rf_filter_coeff360_b <= 8'h000;
rf_filter_coeff361_a <= 8'h000;
rf_filter_coeff361_b <= 8'h000;
rf_filter_coeff362_a <= 8'h000;
rf_filter_coeff362_b <= 8'h000;
rf_filter_coeff363_a <= 8'h000;
rf_filter_coeff363_b <= 8'h000;
rf_filter_coeff364_a <= 8'h000;
rf_filter_coeff364_b <= 8'h000;
rf_filter_coeff365_a <= 8'h000;
rf_filter_coeff365_b <= 8'h000;
rf_filter_coeff366_a <= 8'h000;
rf_filter_coeff366_b <= 8'h000;
rf_filter_coeff367_a <= 8'h000;
rf_filter_coeff367_b <= 8'h000;
rf_filter_coeff368_a <= 8'h000;
rf_filter_coeff368_b <= 8'h000;
rf_filter_coeff369_a <= 8'h000;
rf_filter_coeff369_b <= 8'h000;
rf_filter_coeff370_a <= 8'h000;
rf_filter_coeff370_b <= 8'h000;
rf_filter_coeff371_a <= 8'h000;
rf_filter_coeff371_b <= 8'h000;
rf_filter_coeff372_a <= 8'h000;
rf_filter_coeff372_b <= 8'h000;
rf_filter_coeff373_a <= 8'h000;
rf_filter_coeff373_b <= 8'h000;
rf_filter_coeff374_a <= 8'h000;
rf_filter_coeff374_b <= 8'h000;
rf_filter_coeff375_a <= 8'h000;
rf_filter_coeff375_b <= 8'h000;
rf_filter_coeff376_a <= 8'h000;
rf_filter_coeff376_b <= 8'h000;
rf_filter_coeff377_a <= 8'h000;
rf_filter_coeff377_b <= 8'h000;
rf_filter_coeff378_a <= 8'h000;
rf_filter_coeff378_b <= 8'h000;
rf_filter_coeff379_a <= 8'h000;
rf_filter_coeff379_b <= 8'h000;
rf_filter_coeff380_a <= 8'h000;
rf_filter_coeff380_b <= 8'h000;
rf_filter_coeff381_a <= 8'h000;
rf_filter_coeff381_b <= 8'h000;
rf_filter_coeff382_a <= 8'h000;
rf_filter_coeff382_b <= 8'h000;
rf_filter_coeff383_a <= 8'h000;
rf_filter_coeff383_b <= 8'h000;
rf_filter_coeff384_a <= 8'h000;
rf_filter_coeff384_b <= 8'h000;
rf_filter_coeff385_a <= 8'h000;
rf_filter_coeff385_b <= 8'h000;
rf_filter_coeff386_a <= 8'h000;
rf_filter_coeff386_b <= 8'h000;
rf_filter_coeff387_a <= 8'h000;
rf_filter_coeff387_b <= 8'h000;
rf_filter_coeff388_a <= 8'h000;
rf_filter_coeff388_b <= 8'h000;
rf_filter_coeff389_a <= 8'h000;
rf_filter_coeff389_b <= 8'h000;
rf_filter_coeff390_a <= 8'h000;
rf_filter_coeff390_b <= 8'h000;
rf_filter_coeff391_a <= 8'h000;
rf_filter_coeff391_b <= 8'h000;
rf_filter_coeff392_a <= 8'h000;
rf_filter_coeff392_b <= 8'h000;
rf_filter_coeff393_a <= 8'h000;
rf_filter_coeff393_b <= 8'h000;
rf_filter_coeff394_a <= 8'h000;
rf_filter_coeff394_b <= 8'h000;
rf_filter_coeff395_a <= 8'h000;
rf_filter_coeff395_b <= 8'h000;
rf_filter_coeff396_a <= 8'h000;
rf_filter_coeff396_b <= 8'h000;
rf_filter_coeff397_a <= 8'h000;
rf_filter_coeff397_b <= 8'h000;
rf_filter_coeff398_a <= 8'h000;
rf_filter_coeff398_b <= 8'h000;
rf_filter_coeff399_a <= 8'h000;
rf_filter_coeff399_b <= 8'h000;
rf_filter_coeff400_a <= 8'h000;
rf_filter_coeff400_b <= 8'h000;
rf_filter_coeff401_a <= 8'h000;
rf_filter_coeff401_b <= 8'h000;
rf_filter_coeff402_a <= 8'h000;
rf_filter_coeff402_b <= 8'h000;
rf_filter_coeff403_a <= 8'h000;
rf_filter_coeff403_b <= 8'h000;
rf_filter_coeff404_a <= 8'h000;
rf_filter_coeff404_b <= 8'h000;
rf_filter_coeff405_a <= 8'h000;
```

```
rf_filter_coeff405_b <= 8'h000;
rf_filter_coeff406_a <= 8'h000;
rf_filter_coeff406_b <= 8'h000;
rf_filter_coeff407_a <= 8'h000;
rf_filter_coeff407_b <= 8'h000;
rf_filter_coeff408_a <= 8'h000;
rf_filter_coeff408_b <= 8'h000;
rf_filter_coeff409_a <= 8'h000;
rf_filter_coeff409_b <= 8'h000;
rf_filter_coeff410_a <= 8'h000;
rf_filter_coeff410_b <= 8'h000;
rf_filter_coeff411_a <= 8'h000;
rf_filter_coeff411_b <= 8'h000;
rf_filter_coeff412_a <= 8'h000;
rf_filter_coeff412_b <= 8'h000;
rf_filter_coeff413_a <= 8'h000;
rf_filter_coeff413_b <= 8'h000;
rf_filter_coeff414_a <= 8'h000;
rf_filter_coeff414_b <= 8'h000;
rf_filter_coeff415_a <= 8'h000;
rf_filter_coeff415_b <= 8'h000;
rf_filter_coeff416_a <= 8'h000;
rf_filter_coeff416_b <= 8'h000;
rf_filter_coeff417_a <= 8'h000;
rf_filter_coeff417_b <= 8'h000;
rf_filter_coeff418_a <= 8'h000;
rf_filter_coeff418_b <= 8'h000;
rf_filter_coeff419_a <= 8'h000;
rf_filter_coeff419_b <= 8'h000;
rf_filter_coeff420_a <= 8'h000;
rf_filter_coeff420_b <= 8'h000;
rf_filter_coeff421_a <= 8'h000;
rf_filter_coeff421_b <= 8'h000;
rf_filter_coeff422_a <= 8'h000;
rf_filter_coeff422_b <= 8'h000;
rf_filter_coeff423_a <= 8'h000;
rf_filter_coeff423_b <= 8'h000;
rf_filter_coeff424_a <= 8'h000;
rf_filter_coeff424_b <= 8'h000;
rf_filter_coeff425_a <= 8'h000;
rf_filter_coeff425_b <= 8'h000;
rf_filter_coeff426_a <= 8'h000;
rf_filter_coeff426_b <= 8'h000;
rf_filter_coeff427_a <= 8'h000;
rf_filter_coeff427_b <= 8'h000;
rf_filter_coeff428_a <= 8'h000;
rf_filter_coeff428_b <= 8'h000;
rf_filter_coeff429_a <= 8'h000;
rf_filter_coeff429_b <= 8'h000;
rf_filter_coeff430_a <= 8'h000;
rf_filter_coeff430_b <= 8'h000;
rf_filter_coeff431_a <= 8'h000;
rf_filter_coeff431_b <= 8'h000;
rf_filter_coeff432_a <= 8'h000;
rf_filter_coeff432_b <= 8'h000;
rf_filter_coeff433_a <= 8'h000;
rf_filter_coeff433_b <= 8'h000;
rf_filter_coeff434_a <= 8'h000;
rf_filter_coeff434_b <= 8'h000;
rf_filter_coeff435_a <= 8'h000;
rf_filter_coeff435_b <= 8'h000;
rf_filter_coeff436_a <= 8'h000;
rf_filter_coeff436_b <= 8'h000;
rf_filter_coeff437_a <= 8'h000;
rf_filter_coeff437_b <= 8'h000;
rf_filter_coeff438_a <= 8'h000;
rf_filter_coeff438_b <= 8'h000;
rf_filter_coeff439_a <= 8'h000;
rf_filter_coeff439_b <= 8'h000;
rf_filter_coeff440_a <= 8'h000;
rf_filter_coeff440_b <= 8'h000;
rf_filter_coeff441_a <= 8'h000;
rf_filter_coeff441_b <= 8'h000;
rf_filter_coeff442_a <= 8'h000;
rf_filter_coeff442_b <= 8'h000;
rf_filter_coeff443_a <= 8'h000;
rf_filter_coeff443_b <= 8'h000;
rf_filter_coeff444_a <= 8'h000;
rf_filter_coeff444_b <= 8'h000;
rf_filter_coeff445_a <= 8'h000;
rf_filter_coeff445_b <= 8'h000;
rf_filter_coeff446_a <= 8'h000;
rf_filter_coeff446_b <= 8'h000;
rf_filter_coeff447_a <= 8'h000;
rf_filter_coeff447_b <= 8'h000;
rf_filter_coeff448_a <= 8'h000;
rf_filter_coeff448_b <= 8'h000;
rf_filter_coeff449_a <= 8'h000;
rf_filter_coeff449_b <= 8'h000;
rf_filter_coeff450_a <= 8'h000;
rf_filter_coeff450_b <= 8'h000;
rf_filter_coeff451_a <= 8'h000;
rf_filter_coeff451_b <= 8'h000;
rf_filter_coeff452_a <= 8'h000;
rf_filter_coeff452_b <= 8'h000;
```

```
rf_filter_coeff453_a <= 8'h000;
rf_filter_coeff453_b <= 8'h000;
rf_filter_coeff454_a <= 8'h000;
rf_filter_coeff454_b <= 8'h000;
rf_filter_coeff455_a <= 8'h000;
rf_filter_coeff455_b <= 8'h000;
rf_filter_coeff456_a <= 8'h000;
rf_filter_coeff456_b <= 8'h000;
rf_filter_coeff457_a <= 8'h000;
rf_filter_coeff457_b <= 8'h000;
rf_filter_coeff458_a <= 8'h000;
rf_filter_coeff458_b <= 8'h000;
rf_filter_coeff459_a <= 8'h000;
rf_filter_coeff459_b <= 8'h000;
rf_filter_coeff460_a <= 8'h000;
rf_filter_coeff460_b <= 8'h000;
rf_filter_coeff461_a <= 8'h000;
rf_filter_coeff461_b <= 8'h000;
rf_filter_coeff462_a <= 8'h000;
rf_filter_coeff462_b <= 8'h000;
rf_filter_coeff463_a <= 8'h000;
rf_filter_coeff463_b <= 8'h000;
rf_filter_coeff464_a <= 8'h000;
rf_filter_coeff464_b <= 8'h000;
rf_filter_coeff465_a <= 8'h000;
rf_filter_coeff465_b <= 8'h000;
rf_filter_coeff466_a <= 8'h000;
rf_filter_coeff466_b <= 8'h000;
rf_filter_coeff467_a <= 8'h000;
rf_filter_coeff467_b <= 8'h000;
rf_filter_coeff468_a <= 8'h000;
rf_filter_coeff468_b <= 8'h000;
rf_filter_coeff469_a <= 8'h000;
rf_filter_coeff469_b <= 8'h000;
rf_filter_coeff470_a <= 8'h000;
rf_filter_coeff470_b <= 8'h000;
rf_filter_coeff471_a <= 8'h000;
rf_filter_coeff471_b <= 8'h000;
rf_filter_coeff472_a <= 8'h000;
rf_filter_coeff472_b <= 8'h000;
rf_filter_coeff473_a <= 8'h000;
rf_filter_coeff473_b <= 8'h000;
rf_filter_coeff474_a <= 8'h000;
rf_filter_coeff474_b <= 8'h000;
rf_filter_coeff475_a <= 8'h000;
rf_filter_coeff475_b <= 8'h000;
rf_filter_coeff476_a <= 8'h000;
rf_filter_coeff476_b <= 8'h000;
rf_filter_coeff477_a <= 8'h000;
rf_filter_coeff477_b <= 8'h000;
rf_filter_coeff478_a <= 8'h000;
rf_filter_coeff478_b <= 8'h000;
rf_filter_coeff479_a <= 8'h000;
rf_filter_coeff479_b <= 8'h000;
rf_filter_coeff480_a <= 8'h000;
rf_filter_coeff480_b <= 8'h000;
rf_filter_coeff481_a <= 8'h000;
rf_filter_coeff481_b <= 8'h000;
rf_filter_coeff482_a <= 8'h000;
rf_filter_coeff482_b <= 8'h000;
rf_filter_coeff483_a <= 8'h000;
rf_filter_coeff483_b <= 8'h000;
rf_filter_coeff484_a <= 8'h000;
rf_filter_coeff484_b <= 8'h000;
rf_filter_coeff485_a <= 8'h000;
rf_filter_coeff485_b <= 8'h000;
rf_filter_coeff486_a <= 8'h000;
rf_filter_coeff486_b <= 8'h000;
rf_filter_coeff487_a <= 8'h000;
rf_filter_coeff487_b <= 8'h000;
rf_filter_coeff488_a <= 8'h000;
rf_filter_coeff488_b <= 8'h000;
rf_filter_coeff489_a <= 8'h000;
rf_filter_coeff489_b <= 8'h000;
rf_filter_coeff490_a <= 8'h000;
rf_filter_coeff490_b <= 8'h000;
rf_filter_coeff491_a <= 8'h000;
rf_filter_coeff491_b <= 8'h000;
rf_filter_coeff492_a <= 8'h000;
rf_filter_coeff492_b <= 8'h000;
rf_filter_coeff493_a <= 8'h000;
rf_filter_coeff493_b <= 8'h000;
rf_filter_coeff494_a <= 8'h000;
rf_filter_coeff494_b <= 8'h000;
rf_filter_coeff495_a <= 8'h000;
rf_filter_coeff495_b <= 8'h000;
rf_filter_coeff496_a <= 8'h000;
rf_filter_coeff496_b <= 8'h000;
rf_filter_coeff497_a <= 8'h000;
rf_filter_coeff497_b <= 8'h000;
rf_filter_coeff498_a <= 8'h000;
rf_filter_coeff498_b <= 8'h000;
rf_filter_coeff499_a <= 8'h000;
rf_filter_coeff499_b <= 8'h000;
rf_filter_coeff500_a <= 8'h000;
```

```
            rf_filter_coeff500_b <= 8'h000;
            rf_filter_coeff501_a <= 8'h000;
            rf_filter_coeff501_b <= 8'h000;
            rf_filter_coeff502_a <= 8'h000;
            rf_filter_coeff502_b <= 8'h000;
            rf_filter_coeff503_a <= 8'h000;
            rf_filter_coeff503_b <= 8'h000;
            rf_filter_coeff504_a <= 8'h000;
            rf_filter_coeff504_b <= 8'h000;
            rf_filter_coeff505_a <= 8'h000;
            rf_filter_coeff505_b <= 8'h000;
            rf_filter_coeff506_a <= 8'h000;
            rf_filter_coeff506_b <= 8'h000;
            rf_filter_coeff507_a <= 8'h000;
            rf_filter_coeff507_b <= 8'h000;
            rf_filter_coeff508_a <= 8'h000;
            rf_filter_coeff508_b <= 8'h000;
            rf_filter_coeff509_a <= 8'h000;
            rf_filter_coeff509_b <= 8'h000;
            rf_filter_coeff510_a <= 8'h000;
            rf_filter_coeff510_b <= 8'h000;
            rf_filter_coeff511_a <= 8'h000;
            rf_filter_coeff511_b <= 8'h000;
        end
        else if (wxfc && w_enable)
            begin
                // Given the address, the signals are assigned to their correlated bits of data
                case(addr)
                    11'h004: begin
                        rf_i2si_bist_en <= wdata[0];
                        rf_filter_shift <= wdata[3:1];
                        rf_filter_clip_en <= wdata[4];
                        rf_i2si_en <= wdata[5];
                        end
                    /*11'h008: begin
                                                                                                                                ro_fifo_overrun <= wdata
[1];
                                                                                                                                ro_fifo_underrun <=
wdata[3];
                                                                                                                                ro_filter_ovf_flag <=
wdata[5];
                                                                                                                                end*/
                    11'h00c:
                        rf_i2si_bist_incr <= wdata[7:0];
                    11'h00d:
                        rf_i2si_bist_start_val_a <= wdata[7:0];
                    11'h00e:
                        rf_i2si_bist_start_val_b <= wdata[3:0];
                    11'h010:
                                                                                rf_i2si_bist_upper_limit_a <= wdata[7:0];
                    11'h011:
                        rf_i2si_bist_upper_limit_b <= wdata[3:0];
                    11'h400:
                        rf_filter_coeff0_a <= wdata[7:0];
                    11'h401:
                        rf_filter_coeff0_b <= wdata[7:0];
                    11'h402:
                        rf_filter_coeff1_a <= wdata[7:0];
                    11'h403:
                        rf_filter_coeff1_b <= wdata[7:0];
                    11'h404:
                        rf_filter_coeff2_a <= wdata[7:0];
                    11'h405:
                        rf_filter_coeff2_b <= wdata[7:0];
                    11'h406:
                        rf_filter_coeff3_a <= wdata[7:0];
                    11'h407:
                        rf_filter_coeff3_b <= wdata[7:0];
                    11'h408:
                        rf_filter_coeff4_a <= wdata[7:0];
                    11'h409:
                        rf_filter_coeff4_b <= wdata[7:0];
                    11'h40a:
                        rf_filter_coeff5_a <= wdata[7:0];
                    11'h40b:
                        rf_filter_coeff5_b <= wdata[7:0];
                    11'h40c:
                        rf_filter_coeff6_a <= wdata[7:0];
                    11'h40d:
                        rf_filter_coeff6_b <= wdata[7:0];
                    11'h40e:
                        rf_filter_coeff7_a <= wdata[7:0];
                    11'h40f:
                        rf_filter_coeff7_b <= wdata[7:0];
                    11'h410:
                        rf_filter_coeff8_a <= wdata[7:0];
                    11'h411:
                        rf_filter_coeff8_b <= wdata[7:0];
                    11'h412:
                        rf_filter_coeff9_a <= wdata[7:0];
                    11'h413:
                        rf_filter_coeff9_b <= wdata[7:0];
                    11'h414:
                        rf_filter_coeff10_a <= wdata[7:0];
                    11'h415:
```

```
    rf_filter_coeff10_b <= wdata[7:0];
11'h416:
    rf_filter_coeff11_a <= wdata[7:0];
11'h417:
    rf_filter_coeff11_b <= wdata[7:0];
11'h418:
    rf_filter_coeff12_a <= wdata[7:0];
11'h419:
    rf_filter_coeff12_b <= wdata[7:0];
11'h41a:
    rf_filter_coeff13_a <= wdata[7:0];
11'h41b:
    rf_filter_coeff13_b <= wdata[7:0];
11'h41c:
    rf_filter_coeff14_a <= wdata[7:0];
11'h41d:
    rf_filter_coeff14_b <= wdata[7:0];
11'h41e:
    rf_filter_coeff15_a <= wdata[7:0];
11'h41f:
    rf_filter_coeff15_b <= wdata[7:0];
11'h420:
    rf_filter_coeff16_a <= wdata[7:0];
11'h421:
    rf_filter_coeff16_b <= wdata[7:0];
11'h422:
    rf_filter_coeff17_a <= wdata[7:0];
11'h423:
    rf_filter_coeff17_b <= wdata[7:0];
11'h424:
    rf_filter_coeff18_a <= wdata[7:0];
11'h425:
    rf_filter_coeff18_b <= wdata[7:0];
11'h426:
    rf_filter_coeff19_a <= wdata[7:0];
11'h427:
    rf_filter_coeff19_b <= wdata[7:0];
11'h428:
    rf_filter_coeff20_a <= wdata[7:0];
11'h429:
    rf_filter_coeff20_b <= wdata[7:0];
11'h42a:
    rf_filter_coeff21_a <= wdata[7:0];
11'h42b:
    rf_filter_coeff21_b <= wdata[7:0];
11'h42c:
    rf_filter_coeff22_a <= wdata[7:0];
11'h42d:
    rf_filter_coeff22_b <= wdata[7:0];
11'h42e:
    rf_filter_coeff23_a <= wdata[7:0];
11'h42f:
    rf_filter_coeff23_b <= wdata[7:0];
11'h430:
    rf_filter_coeff24_a <= wdata[7:0];
11'h431:
    rf_filter_coeff24_b <= wdata[7:0];
11'h432:
    rf_filter_coeff25_a <= wdata[7:0];
11'h433:
    rf_filter_coeff25_b <= wdata[7:0];
11'h434:
    rf_filter_coeff26_a <= wdata[7:0];
11'h435:
    rf_filter_coeff26_b <= wdata[7:0];
11'h436:
    rf_filter_coeff27_a <= wdata[7:0];
11'h437:
    rf_filter_coeff27_b <= wdata[7:0];
11'h438:
    rf_filter_coeff28_a <= wdata[7:0];
11'h439:
    rf_filter_coeff28_b <= wdata[7:0];
11'h43a:
    rf_filter_coeff29_a <= wdata[7:0];
11'h43b:
    rf_filter_coeff29_b <= wdata[7:0];
11'h43c:
    rf_filter_coeff30_a <= wdata[7:0];
11'h43d:
    rf_filter_coeff30_b <= wdata[7:0];
11'h43e:
    rf_filter_coeff31_a <= wdata[7:0];
11'h43f:
    rf_filter_coeff31_b <= wdata[7:0];
11'h440:
    rf_filter_coeff32_a <= wdata[7:0];
11'h441:
    rf_filter_coeff32_b <= wdata[7:0];
11'h442:
    rf_filter_coeff33_a <= wdata[7:0];
11'h443:
    rf_filter_coeff33_b <= wdata[7:0];
11'h444:
    rf_filter_coeff34_a <= wdata[7:0];
```

```
11'h445:
    rf_filter_coeff34_b <= wdata[7:0];
11'h446:
    rf_filter_coeff35_a <= wdata[7:0];
11'h447:
    rf_filter_coeff35_b <= wdata[7:0];
11'h448:
    rf_filter_coeff36_a <= wdata[7:0];
11'h449:
    rf_filter_coeff36_b <= wdata[7:0];
11'h44a:
    rf_filter_coeff37_a <= wdata[7:0];
11'h44b:
    rf_filter_coeff37_b <= wdata[7:0];
11'h44c:
    rf_filter_coeff38_a <= wdata[7:0];
11'h44d:
    rf_filter_coeff38_b <= wdata[7:0];
11'h44e:
    rf_filter_coeff39_a <= wdata[7:0];
11'h44f:
    rf_filter_coeff39_b <= wdata[7:0];
11'h450:
    rf_filter_coeff40_a <= wdata[7:0];
11'h451:
    rf_filter_coeff40_b <= wdata[7:0];
11'h452:
    rf_filter_coeff41_a <= wdata[7:0];
11'h453:
    rf_filter_coeff41_b <= wdata[7:0];
11'h454:
    rf_filter_coeff42_a <= wdata[7:0];
11'h455:
    rf_filter_coeff42_b <= wdata[7:0];
11'h456:
    rf_filter_coeff43_a <= wdata[7:0];
11'h457:
    rf_filter_coeff43_b <= wdata[7:0];
11'h458:
    rf_filter_coeff44_a <= wdata[7:0];
11'h459:
    rf_filter_coeff44_b <= wdata[7:0];
11'h45a:
    rf_filter_coeff45_a <= wdata[7:0];
11'h45b:
    rf_filter_coeff45_b <= wdata[7:0];
11'h45c:
    rf_filter_coeff46_a <= wdata[7:0];
11'h45d:
    rf_filter_coeff46_b <= wdata[7:0];
11'h45e:
    rf_filter_coeff47_a <= wdata[7:0];
11'h45f:
    rf_filter_coeff47_b <= wdata[7:0];
11'h460:
    rf_filter_coeff48_a <= wdata[7:0];
11'h461:
    rf_filter_coeff48_b <= wdata[7:0];
11'h462:
    rf_filter_coeff49_a <= wdata[7:0];
11'h463:
    rf_filter_coeff49_b <= wdata[7:0];
11'h464:
    rf_filter_coeff50_a <= wdata[7:0];
11'h465:
    rf_filter_coeff50_b <= wdata[7:0];
11'h466:
    rf_filter_coeff51_a <= wdata[7:0];
11'h467:
    rf_filter_coeff51_b <= wdata[7:0];
11'h468:
    rf_filter_coeff52_a <= wdata[7:0];
11'h469:
    rf_filter_coeff52_b <= wdata[7:0];
11'h46a:
    rf_filter_coeff53_a <= wdata[7:0];
11'h46b:
    rf_filter_coeff53_b <= wdata[7:0];
11'h46c:
    rf_filter_coeff54_a <= wdata[7:0];
11'h46d:
    rf_filter_coeff54_b <= wdata[7:0];
11'h46e:
    rf_filter_coeff55_a <= wdata[7:0];
11'h46f:
    rf_filter_coeff55_b <= wdata[7:0];
11'h470:
    rf_filter_coeff56_a <= wdata[7:0];
11'h471:
    rf_filter_coeff56_b <= wdata[7:0];
11'h472:
    rf_filter_coeff57_a <= wdata[7:0];
11'h473:
    rf_filter_coeff57_b <= wdata[7:0];
11'h474:
```

```
    rf_filter_coeff58_a <= wdata[7:0];
11'h475:
    rf_filter_coeff58_b <= wdata[7:0];
11'h476:
    rf_filter_coeff59_a <= wdata[7:0];
11'h477:
    rf_filter_coeff59_b <= wdata[7:0];
11'h478:
    rf_filter_coeff60_a <= wdata[7:0];
11'h479:
    rf_filter_coeff60_b <= wdata[7:0];
11'h47a:
    rf_filter_coeff61_a <= wdata[7:0];
11'h47b:
    rf_filter_coeff61_b <= wdata[7:0];
11'h47c:
    rf_filter_coeff62_a <= wdata[7:0];
11'h47d:
    rf_filter_coeff62_b <= wdata[7:0];
11'h47e:
    rf_filter_coeff63_a <= wdata[7:0];
11'h47f:
    rf_filter_coeff63_b <= wdata[7:0];
11'h480:
    rf_filter_coeff64_a <= wdata[7:0];
11'h481:
    rf_filter_coeff64_b <= wdata[7:0];
11'h482:
    rf_filter_coeff65_a <= wdata[7:0];
11'h483:
    rf_filter_coeff65_b <= wdata[7:0];
11'h484:
    rf_filter_coeff66_a <= wdata[7:0];
11'h485:
    rf_filter_coeff66_b <= wdata[7:0];
11'h486:
    rf_filter_coeff67_a <= wdata[7:0];
11'h487:
    rf_filter_coeff67_b <= wdata[7:0];
11'h488:
    rf_filter_coeff68_a <= wdata[7:0];
11'h489:
    rf_filter_coeff68_b <= wdata[7:0];
11'h48a:
    rf_filter_coeff69_a <= wdata[7:0];
11'h48b:
    rf_filter_coeff69_b <= wdata[7:0];
11'h48c:
    rf_filter_coeff70_a <= wdata[7:0];
11'h48d:
    rf_filter_coeff70_b <= wdata[7:0];
11'h48e:
    rf_filter_coeff71_a <= wdata[7:0];
11'h48f:
    rf_filter_coeff71_b <= wdata[7:0];
11'h490:
    rf_filter_coeff72_a <= wdata[7:0];
11'h491:
    rf_filter_coeff72_b <= wdata[7:0];
11'h492:
    rf_filter_coeff73_a <= wdata[7:0];
11'h493:
    rf_filter_coeff73_b <= wdata[7:0];
11'h494:
    rf_filter_coeff74_a <= wdata[7:0];
11'h495:
    rf_filter_coeff74_b <= wdata[7:0];
11'h496:
    rf_filter_coeff75_a <= wdata[7:0];
11'h497:
    rf_filter_coeff75_b <= wdata[7:0];
11'h498:
    rf_filter_coeff76_a <= wdata[7:0];
11'h499:
    rf_filter_coeff76_b <= wdata[7:0];
11'h49a:
    rf_filter_coeff77_a <= wdata[7:0];
11'h49b:
    rf_filter_coeff77_b <= wdata[7:0];
11'h49c:
    rf_filter_coeff78_a <= wdata[7:0];
11'h49d:
    rf_filter_coeff78_b <= wdata[7:0];
11'h49e:
    rf_filter_coeff79_a <= wdata[7:0];
11'h49f:
    rf_filter_coeff79_b <= wdata[7:0];
11'h4a0:
    rf_filter_coeff80_a <= wdata[7:0];
11'h4a1:
    rf_filter_coeff80_b <= wdata[7:0];
11'h4a2:
    rf_filter_coeff81_a <= wdata[7:0];
11'h4a3:
    rf_filter_coeff81_b <= wdata[7:0];
```

```
11'h4a4:
    rf_filter_coeff82_a <= wdata[7:0];
11'h4a5:
    rf_filter_coeff82_b <= wdata[7:0];
11'h4a6:
    rf_filter_coeff83_a <= wdata[7:0];
11'h4a7:
    rf_filter_coeff83_b <= wdata[7:0];
11'h4a8:
    rf_filter_coeff84_a <= wdata[7:0];
11'h4a9:
    rf_filter_coeff84_b <= wdata[7:0];
11'h4aa:
    rf_filter_coeff85_a <= wdata[7:0];
11'h4ab:
    rf_filter_coeff85_b <= wdata[7:0];
11'h4ac:
    rf_filter_coeff86_a <= wdata[7:0];
11'h4ad:
    rf_filter_coeff86_b <= wdata[7:0];
11'h4ae:
    rf_filter_coeff87_a <= wdata[7:0];
11'h4af:
    rf_filter_coeff87_b <= wdata[7:0];
11'h4b0:
    rf_filter_coeff88_a <= wdata[7:0];
11'h4b1:
    rf_filter_coeff88_b <= wdata[7:0];
11'h4b2:
    rf_filter_coeff89_a <= wdata[7:0];
11'h4b3:
    rf_filter_coeff89_b <= wdata[7:0];
11'h4b4:
    rf_filter_coeff90_a <= wdata[7:0];
11'h4b5:
    rf_filter_coeff90_b <= wdata[7:0];
11'h4b6:
    rf_filter_coeff91_a <= wdata[7:0];
11'h4b7:
    rf_filter_coeff91_b <= wdata[7:0];
11'h4b8:
    rf_filter_coeff92_a <= wdata[7:0];
11'h4b9:
    rf_filter_coeff92_b <= wdata[7:0];
11'h4ba:
    rf_filter_coeff93_a <= wdata[7:0];
11'h4bb:
    rf_filter_coeff93_b <= wdata[7:0];
11'h4bc:
    rf_filter_coeff94_a <= wdata[7:0];
11'h4bd:
    rf_filter_coeff94_b <= wdata[7:0];
11'h4be:
    rf_filter_coeff95_a <= wdata[7:0];
11'h4bf:
    rf_filter_coeff95_b <= wdata[7:0];
11'h4c0:
    rf_filter_coeff96_a <= wdata[7:0];
11'h4c1:
    rf_filter_coeff96_b <= wdata[7:0];
11'h4c2:
    rf_filter_coeff97_a <= wdata[7:0];
11'h4c3:
    rf_filter_coeff97_b <= wdata[7:0];
11'h4c4:
    rf_filter_coeff98_a <= wdata[7:0];
11'h4c5:
    rf_filter_coeff98_b <= wdata[7:0];
11'h4c6:
    rf_filter_coeff99_a <= wdata[7:0];
11'h4c7:
    rf_filter_coeff99_b <= wdata[7:0];
11'h4c8:
    rf_filter_coeff100_a <= wdata[7:0];
11'h4c9:
    rf_filter_coeff100_b <= wdata[7:0];
11'h4ca:
    rf_filter_coeff101_a <= wdata[7:0];
11'h4cb:
    rf_filter_coeff101_b <= wdata[7:0];
11'h4cc:
    rf_filter_coeff102_a <= wdata[7:0];
11'h4cd:
    rf_filter_coeff102_b <= wdata[7:0];
11'h4ce:
    rf_filter_coeff103_a <= wdata[7:0];
11'h4cf:
    rf_filter_coeff103_b <= wdata[7:0];
11'h4d0:
    rf_filter_coeff104_a <= wdata[7:0];
11'h4d1:
    rf_filter_coeff104_b <= wdata[7:0];
11'h4d2:
    rf_filter_coeff105_a <= wdata[7:0];
11'h4d3:
```

```
    rf_filter_coeff105_b <= wdata[7:0];
11'h4d4:
    rf_filter_coeff106_a <= wdata[7:0];
11'h4d5:
    rf_filter_coeff106_b <= wdata[7:0];
11'h4d6:
    rf_filter_coeff107_a <= wdata[7:0];
11'h4d7:
    rf_filter_coeff107_b <= wdata[7:0];
11'h4d8:
    rf_filter_coeff108_a <= wdata[7:0];
11'h4d9:
    rf_filter_coeff108_b <= wdata[7:0];
11'h4da:
    rf_filter_coeff109_a <= wdata[7:0];
11'h4db:
    rf_filter_coeff109_b <= wdata[7:0];
11'h4dc:
    rf_filter_coeff110_a <= wdata[7:0];
11'h4dd:
    rf_filter_coeff110_b <= wdata[7:0];
11'h4de:
    rf_filter_coeff111_a <= wdata[7:0];
11'h4df:
    rf_filter_coeff111_b <= wdata[7:0];
11'h4e0:
    rf_filter_coeff112_a <= wdata[7:0];
11'h4e1:
    rf_filter_coeff112_b <= wdata[7:0];
11'h4e2:
    rf_filter_coeff113_a <= wdata[7:0];
11'h4e3:
    rf_filter_coeff113_b <= wdata[7:0];
11'h4e4:
    rf_filter_coeff114_a <= wdata[7:0];
11'h4e5:
    rf_filter_coeff114_b <= wdata[7:0];
11'h4e6:
    rf_filter_coeff115_a <= wdata[7:0];
11'h4e7:
    rf_filter_coeff115_b <= wdata[7:0];
11'h4e8:
    rf_filter_coeff116_a <= wdata[7:0];
11'h4e9:
    rf_filter_coeff116_b <= wdata[7:0];
11'h4ea:
    rf_filter_coeff117_a <= wdata[7:0];
11'h4eb:
    rf_filter_coeff117_b <= wdata[7:0];
11'h4ec:
    rf_filter_coeff118_a <= wdata[7:0];
11'h4ed:
    rf_filter_coeff118_b <= wdata[7:0];
11'h4ee:
    rf_filter_coeff119_a <= wdata[7:0];
11'h4ef:
    rf_filter_coeff119_b <= wdata[7:0];
11'h4f0:
    rf_filter_coeff120_a <= wdata[7:0];
11'h4f1:
    rf_filter_coeff120_b <= wdata[7:0];
11'h4f2:
    rf_filter_coeff121_a <= wdata[7:0];
11'h4f3:
    rf_filter_coeff121_b <= wdata[7:0];
11'h4f4:
    rf_filter_coeff122_a <= wdata[7:0];
11'h4f5:
    rf_filter_coeff122_b <= wdata[7:0];
11'h4f6:
    rf_filter_coeff123_a <= wdata[7:0];
11'h4f7:
    rf_filter_coeff123_b <= wdata[7:0];
11'h4f8:
    rf_filter_coeff124_a <= wdata[7:0];
11'h4f9:
    rf_filter_coeff124_b <= wdata[7:0];
11'h4fa:
    rf_filter_coeff125_a <= wdata[7:0];
11'h4fb:
    rf_filter_coeff125_b <= wdata[7:0];
11'h4fc:
    rf_filter_coeff126_a <= wdata[7:0];
11'h4fd:
    rf_filter_coeff126_b <= wdata[7:0];
11'h4fe:
    rf_filter_coeff127_a <= wdata[7:0];
11'h4ff:
    rf_filter_coeff127_b <= wdata[7:0];
11'h500:
    rf_filter_coeff128_a <= wdata[7:0];
11'h501:
    rf_filter_coeff128_b <= wdata[7:0];
11'h502:
    rf_filter_coeff129_a <= wdata[7:0];
```

```
11'h503:
    rf_filter_coeff129_b <= wdata[7:0];
11'h504:
    rf_filter_coeff130_a <= wdata[7:0];
11'h505:
    rf_filter_coeff130_b <= wdata[7:0];
11'h506:
    rf_filter_coeff131_a <= wdata[7:0];
11'h507:
    rf_filter_coeff131_b <= wdata[7:0];
11'h508:
    rf_filter_coeff132_a <= wdata[7:0];
11'h509:
    rf_filter_coeff132_b <= wdata[7:0];
11'h50a:
    rf_filter_coeff133_a <= wdata[7:0];
11'h50b:
    rf_filter_coeff133_b <= wdata[7:0];
11'h50c:
    rf_filter_coeff134_a <= wdata[7:0];
11'h50d:
    rf_filter_coeff134_b <= wdata[7:0];
11'h50e:
    rf_filter_coeff135_a <= wdata[7:0];
11'h50f:
    rf_filter_coeff135_b <= wdata[7:0];
11'h510:
    rf_filter_coeff136_a <= wdata[7:0];
11'h511:
    rf_filter_coeff136_b <= wdata[7:0];
11'h512:
    rf_filter_coeff137_a <= wdata[7:0];
11'h513:
    rf_filter_coeff137_b <= wdata[7:0];
11'h514:
    rf_filter_coeff138_a <= wdata[7:0];
11'h515:
    rf_filter_coeff138_b <= wdata[7:0];
11'h516:
    rf_filter_coeff139_a <= wdata[7:0];
11'h517:
    rf_filter_coeff139_b <= wdata[7:0];
11'h518:
    rf_filter_coeff140_a <= wdata[7:0];
11'h519:
    rf_filter_coeff140_b <= wdata[7:0];
11'h51a:
    rf_filter_coeff141_a <= wdata[7:0];
11'h51b:
    rf_filter_coeff141_b <= wdata[7:0];
11'h51c:
    rf_filter_coeff142_a <= wdata[7:0];
11'h51d:
    rf_filter_coeff142_b <= wdata[7:0];
11'h51e:
    rf_filter_coeff143_a <= wdata[7:0];
11'h51f:
    rf_filter_coeff143_b <= wdata[7:0];
11'h520:
    rf_filter_coeff144_a <= wdata[7:0];
11'h521:
    rf_filter_coeff144_b <= wdata[7:0];
11'h522:
    rf_filter_coeff145_a <= wdata[7:0];
11'h523:
    rf_filter_coeff145_b <= wdata[7:0];
11'h524:
    rf_filter_coeff146_a <= wdata[7:0];
11'h525:
    rf_filter_coeff146_b <= wdata[7:0];
11'h526:
    rf_filter_coeff147_a <= wdata[7:0];
11'h527:
    rf_filter_coeff147_b <= wdata[7:0];
11'h528:
    rf_filter_coeff148_a <= wdata[7:0];
11'h529:
    rf_filter_coeff148_b <= wdata[7:0];
11'h52a:
    rf_filter_coeff149_a <= wdata[7:0];
11'h52b:
    rf_filter_coeff149_b <= wdata[7:0];
11'h52c:
    rf_filter_coeff150_a <= wdata[7:0];
11'h52d:
    rf_filter_coeff150_b <= wdata[7:0];
11'h52e:
    rf_filter_coeff151_a <= wdata[7:0];
11'h52f:
    rf_filter_coeff151_b <= wdata[7:0];
11'h530:
    rf_filter_coeff152_a <= wdata[7:0];
11'h531:
    rf_filter_coeff152_b <= wdata[7:0];
11'h532:
```

```
    rf_filter_coeff153_a <= wdata[7:0];
11'h533:
    rf_filter_coeff153_b <= wdata[7:0];
11'h534:
    rf_filter_coeff154_a <= wdata[7:0];
11'h535:
    rf_filter_coeff154_b <= wdata[7:0];
11'h536:
    rf_filter_coeff155_a <= wdata[7:0];
11'h537:
    rf_filter_coeff155_b <= wdata[7:0];
11'h538:
    rf_filter_coeff156_a <= wdata[7:0];
11'h539:
    rf_filter_coeff156_b <= wdata[7:0];
11'h53a:
    rf_filter_coeff157_a <= wdata[7:0];
11'h53b:
    rf_filter_coeff157_b <= wdata[7:0];
11'h53c:
    rf_filter_coeff158_a <= wdata[7:0];
11'h53d:
    rf_filter_coeff158_b <= wdata[7:0];
11'h53e:
    rf_filter_coeff159_a <= wdata[7:0];
11'h53f:
    rf_filter_coeff159_b <= wdata[7:0];
11'h540:
    rf_filter_coeff160_a <= wdata[7:0];
11'h541:
    rf_filter_coeff160_b <= wdata[7:0];
11'h542:
    rf_filter_coeff161_a <= wdata[7:0];
11'h543:
    rf_filter_coeff161_b <= wdata[7:0];
11'h544:
    rf_filter_coeff162_a <= wdata[7:0];
11'h545:
    rf_filter_coeff162_b <= wdata[7:0];
11'h546:
    rf_filter_coeff163_a <= wdata[7:0];
11'h547:
    rf_filter_coeff163_b <= wdata[7:0];
11'h548:
    rf_filter_coeff164_a <= wdata[7:0];
11'h549:
    rf_filter_coeff164_b <= wdata[7:0];
11'h54a:
    rf_filter_coeff165_a <= wdata[7:0];
11'h54b:
    rf_filter_coeff165_b <= wdata[7:0];
11'h54c:
    rf_filter_coeff166_a <= wdata[7:0];
11'h54d:
    rf_filter_coeff166_b <= wdata[7:0];
11'h54e:
    rf_filter_coeff167_a <= wdata[7:0];
11'h54f:
    rf_filter_coeff167_b <= wdata[7:0];
11'h550:
    rf_filter_coeff168_a <= wdata[7:0];
11'h551:
    rf_filter_coeff168_b <= wdata[7:0];
11'h552:
    rf_filter_coeff169_a <= wdata[7:0];
11'h553:
    rf_filter_coeff169_b <= wdata[7:0];
11'h554:
    rf_filter_coeff170_a <= wdata[7:0];
11'h555:
    rf_filter_coeff170_b <= wdata[7:0];
11'h556:
    rf_filter_coeff171_a <= wdata[7:0];
11'h557:
    rf_filter_coeff171_b <= wdata[7:0];
11'h558:
    rf_filter_coeff172_a <= wdata[7:0];
11'h559:
    rf_filter_coeff172_b <= wdata[7:0];
11'h55a:
    rf_filter_coeff173_a <= wdata[7:0];
11'h55b:
    rf_filter_coeff173_b <= wdata[7:0];
11'h55c:
    rf_filter_coeff174_a <= wdata[7:0];
11'h55d:
    rf_filter_coeff174_b <= wdata[7:0];
11'h55e:
    rf_filter_coeff175_a <= wdata[7:0];
11'h55f:
    rf_filter_coeff175_b <= wdata[7:0];
11'h560:
    rf_filter_coeff176_a <= wdata[7:0];
11'h561:
    rf_filter_coeff176_b <= wdata[7:0];
```

```
11'h562:
    rf_filter_coeff177_a <= wdata[7:0];
11'h563:
    rf_filter_coeff177_b <= wdata[7:0];
11'h564:
    rf_filter_coeff178_a <= wdata[7:0];
11'h565:
    rf_filter_coeff178_b <= wdata[7:0];
11'h566:
    rf_filter_coeff179_a <= wdata[7:0];
11'h567:
    rf_filter_coeff179_b <= wdata[7:0];
11'h568:
    rf_filter_coeff180_a <= wdata[7:0];
11'h569:
    rf_filter_coeff180_b <= wdata[7:0];
11'h56a:
    rf_filter_coeff181_a <= wdata[7:0];
11'h56b:
    rf_filter_coeff181_b <= wdata[7:0];
11'h56c:
    rf_filter_coeff182_a <= wdata[7:0];
11'h56d:
    rf_filter_coeff182_b <= wdata[7:0];
11'h56e:
    rf_filter_coeff183_a <= wdata[7:0];
11'h56f:
    rf_filter_coeff183_b <= wdata[7:0];
11'h570:
    rf_filter_coeff184_a <= wdata[7:0];
11'h571:
    rf_filter_coeff184_b <= wdata[7:0];
11'h572:
    rf_filter_coeff185_a <= wdata[7:0];
11'h573:
    rf_filter_coeff185_b <= wdata[7:0];
11'h574:
    rf_filter_coeff186_a <= wdata[7:0];
11'h575:
    rf_filter_coeff186_b <= wdata[7:0];
11'h576:
    rf_filter_coeff187_a <= wdata[7:0];
11'h577:
    rf_filter_coeff187_b <= wdata[7:0];
11'h578:
    rf_filter_coeff188_a <= wdata[7:0];
11'h579:
    rf_filter_coeff188_b <= wdata[7:0];
11'h57a:
    rf_filter_coeff189_a <= wdata[7:0];
11'h57b:
    rf_filter_coeff189_b <= wdata[7:0];
11'h57c:
    rf_filter_coeff190_a <= wdata[7:0];
11'h57d:
    rf_filter_coeff190_b <= wdata[7:0];
11'h57e:
    rf_filter_coeff191_a <= wdata[7:0];
11'h57f:
    rf_filter_coeff191_b <= wdata[7:0];
11'h580:
    rf_filter_coeff192_a <= wdata[7:0];
11'h581:
    rf_filter_coeff192_b <= wdata[7:0];
11'h582:
    rf_filter_coeff193_a <= wdata[7:0];
11'h583:
    rf_filter_coeff193_b <= wdata[7:0];
11'h584:
    rf_filter_coeff194_a <= wdata[7:0];
11'h585:
    rf_filter_coeff194_b <= wdata[7:0];
11'h586:
    rf_filter_coeff195_a <= wdata[7:0];
11'h587:
    rf_filter_coeff195_b <= wdata[7:0];
11'h588:
    rf_filter_coeff196_a <= wdata[7:0];
11'h589:
    rf_filter_coeff196_b <= wdata[7:0];
11'h58a:
    rf_filter_coeff197_a <= wdata[7:0];
11'h58b:
    rf_filter_coeff197_b <= wdata[7:0];
11'h58c:
    rf_filter_coeff198_a <= wdata[7:0];
11'h58d:
    rf_filter_coeff198_b <= wdata[7:0];
11'h58e:
    rf_filter_coeff199_a <= wdata[7:0];
11'h58f:
    rf_filter_coeff199_b <= wdata[7:0];
11'h590:
    rf_filter_coeff200_a <= wdata[7:0];
11'h591:
```

```
    rf_filter_coeff200_b <= wdata[7:0];
11'h592:
    rf_filter_coeff201_a <= wdata[7:0];
11'h593:
    rf_filter_coeff201_b <= wdata[7:0];
11'h594:
    rf_filter_coeff202_a <= wdata[7:0];
11'h595:
    rf_filter_coeff202_b <= wdata[7:0];
11'h596:
    rf_filter_coeff203_a <= wdata[7:0];
11'h597:
    rf_filter_coeff203_b <= wdata[7:0];
11'h598:
    rf_filter_coeff204_a <= wdata[7:0];
11'h599:
    rf_filter_coeff204_b <= wdata[7:0];
11'h59a:
    rf_filter_coeff205_a <= wdata[7:0];
11'h59b:
    rf_filter_coeff205_b <= wdata[7:0];
11'h59c:
    rf_filter_coeff206_a <= wdata[7:0];
11'h59d:
    rf_filter_coeff206_b <= wdata[7:0];
11'h59e:
    rf_filter_coeff207_a <= wdata[7:0];
11'h59f:
    rf_filter_coeff207_b <= wdata[7:0];
11'h5a0:
    rf_filter_coeff208_a <= wdata[7:0];
11'h5a1:
    rf_filter_coeff208_b <= wdata[7:0];
11'h5a2:
    rf_filter_coeff209_a <= wdata[7:0];
11'h5a3:
    rf_filter_coeff209_b <= wdata[7:0];
11'h5a4:
    rf_filter_coeff210_a <= wdata[7:0];
11'h5a5:
    rf_filter_coeff210_b <= wdata[7:0];
11'h5a6:
    rf_filter_coeff211_a <= wdata[7:0];
11'h5a7:
    rf_filter_coeff211_b <= wdata[7:0];
11'h5a8:
    rf_filter_coeff212_a <= wdata[7:0];
11'h5a9:
    rf_filter_coeff212_b <= wdata[7:0];
11'h5aa:
    rf_filter_coeff213_a <= wdata[7:0];
11'h5ab:
    rf_filter_coeff213_b <= wdata[7:0];
11'h5ac:
    rf_filter_coeff214_a <= wdata[7:0];
11'h5ad:
    rf_filter_coeff214_b <= wdata[7:0];
11'h5ae:
    rf_filter_coeff215_a <= wdata[7:0];
11'h5af:
    rf_filter_coeff215_b <= wdata[7:0];
11'h5b0:
    rf_filter_coeff216_a <= wdata[7:0];
11'h5b1:
    rf_filter_coeff216_b <= wdata[7:0];
11'h5b2:
    rf_filter_coeff217_a <= wdata[7:0];
11'h5b3:
    rf_filter_coeff217_b <= wdata[7:0];
11'h5b4:
    rf_filter_coeff218_a <= wdata[7:0];
11'h5b5:
    rf_filter_coeff218_b <= wdata[7:0];
11'h5b6:
    rf_filter_coeff219_a <= wdata[7:0];
11'h5b7:
    rf_filter_coeff219_b <= wdata[7:0];
11'h5b8:
    rf_filter_coeff220_a <= wdata[7:0];
11'h5b9:
    rf_filter_coeff220_b <= wdata[7:0];
11'h5ba:
    rf_filter_coeff221_a <= wdata[7:0];
11'h5bb:
    rf_filter_coeff221_b <= wdata[7:0];
11'h5bc:
    rf_filter_coeff222_a <= wdata[7:0];
11'h5bd:
    rf_filter_coeff222_b <= wdata[7:0];
11'h5be:
    rf_filter_coeff223_a <= wdata[7:0];
11'h5bf:
    rf_filter_coeff223_b <= wdata[7:0];
11'h5c0:
    rf_filter_coeff224_a <= wdata[7:0];
```

```
11'h5c1:
    rf_filter_coeff224_b <= wdata[7:0];
11'h5c2:
    rf_filter_coeff225_a <= wdata[7:0];
11'h5c3:
    rf_filter_coeff225_b <= wdata[7:0];
11'h5c4:
    rf_filter_coeff226_a <= wdata[7:0];
11'h5c5:
    rf_filter_coeff226_b <= wdata[7:0];
11'h5c6:
    rf_filter_coeff227_a <= wdata[7:0];
11'h5c7:
    rf_filter_coeff227_b <= wdata[7:0];
11'h5c8:
    rf_filter_coeff228_a <= wdata[7:0];
11'h5c9:
    rf_filter_coeff228_b <= wdata[7:0];
11'h5ca:
    rf_filter_coeff229_a <= wdata[7:0];
11'h5cb:
    rf_filter_coeff229_b <= wdata[7:0];
11'h5cc:
    rf_filter_coeff230_a <= wdata[7:0];
11'h5cd:
    rf_filter_coeff230_b <= wdata[7:0];
11'h5ce:
    rf_filter_coeff231_a <= wdata[7:0];
11'h5cf:
    rf_filter_coeff231_b <= wdata[7:0];
11'h5d0:
    rf_filter_coeff232_a <= wdata[7:0];
11'h5d1:
    rf_filter_coeff232_b <= wdata[7:0];
11'h5d2:
    rf_filter_coeff233_a <= wdata[7:0];
11'h5d3:
    rf_filter_coeff233_b <= wdata[7:0];
11'h5d4:
    rf_filter_coeff234_a <= wdata[7:0];
11'h5d5:
    rf_filter_coeff234_b <= wdata[7:0];
11'h5d6:
    rf_filter_coeff235_a <= wdata[7:0];
11'h5d7:
    rf_filter_coeff235_b <= wdata[7:0];
11'h5d8:
    rf_filter_coeff236_a <= wdata[7:0];
11'h5d9:
    rf_filter_coeff236_b <= wdata[7:0];
11'h5da:
    rf_filter_coeff237_a <= wdata[7:0];
11'h5db:
    rf_filter_coeff237_b <= wdata[7:0];
11'h5dc:
    rf_filter_coeff238_a <= wdata[7:0];
11'h5dd:
    rf_filter_coeff238_b <= wdata[7:0];
11'h5de:
    rf_filter_coeff239_a <= wdata[7:0];
11'h5df:
    rf_filter_coeff239_b <= wdata[7:0];
11'h5e0:
    rf_filter_coeff240_a <= wdata[7:0];
11'h5e1:
    rf_filter_coeff240_b <= wdata[7:0];
11'h5e2:
    rf_filter_coeff241_a <= wdata[7:0];
11'h5e3:
    rf_filter_coeff241_b <= wdata[7:0];
11'h5e4:
    rf_filter_coeff242_a <= wdata[7:0];
11'h5e5:
    rf_filter_coeff242_b <= wdata[7:0];
11'h5e6:
    rf_filter_coeff243_a <= wdata[7:0];
11'h5e7:
    rf_filter_coeff243_b <= wdata[7:0];
11'h5e8:
    rf_filter_coeff244_a <= wdata[7:0];
11'h5e9:
    rf_filter_coeff244_b <= wdata[7:0];
11'h5ea:
    rf_filter_coeff245_a <= wdata[7:0];
11'h5eb:
    rf_filter_coeff245_b <= wdata[7:0];
11'h5ec:
    rf_filter_coeff246_a <= wdata[7:0];
11'h5ed:
    rf_filter_coeff246_b <= wdata[7:0];
11'h5ee:
    rf_filter_coeff247_a <= wdata[7:0];
11'h5ef:
    rf_filter_coeff247_b <= wdata[7:0];
11'h5f0:
```

```
    rf_filter_coeff248_a <= wdata[7:0];
11'h5f1:
    rf_filter_coeff248_b <= wdata[7:0];
11'h5f2:
    rf_filter_coeff249_a <= wdata[7:0];
11'h5f3:
    rf_filter_coeff249_b <= wdata[7:0];
11'h5f4:
    rf_filter_coeff250_a <= wdata[7:0];
11'h5f5:
    rf_filter_coeff250_b <= wdata[7:0];
11'h5f6:
    rf_filter_coeff251_a <= wdata[7:0];
11'h5f7:
    rf_filter_coeff251_b <= wdata[7:0];
11'h5f8:
    rf_filter_coeff252_a <= wdata[7:0];
11'h5f9:
    rf_filter_coeff252_b <= wdata[7:0];
11'h5fa:
    rf_filter_coeff253_a <= wdata[7:0];
11'h5fb:
    rf_filter_coeff253_b <= wdata[7:0];
11'h5fc:
    rf_filter_coeff254_a <= wdata[7:0];
11'h5fd:
    rf_filter_coeff254_b <= wdata[7:0];
11'h5fe:
    rf_filter_coeff255_a <= wdata[7:0];
11'h5ff:
    rf_filter_coeff255_b <= wdata[7:0];
11'h600:
    rf_filter_coeff256_a <= wdata[7:0];
11'h601:
    rf_filter_coeff256_b <= wdata[7:0];
11'h602:
    rf_filter_coeff257_a <= wdata[7:0];
11'h603:
    rf_filter_coeff257_b <= wdata[7:0];
11'h604:
    rf_filter_coeff258_a <= wdata[7:0];
11'h605:
    rf_filter_coeff258_b <= wdata[7:0];
11'h606:
    rf_filter_coeff259_a <= wdata[7:0];
11'h607:
    rf_filter_coeff259_b <= wdata[7:0];
11'h608:
    rf_filter_coeff260_a <= wdata[7:0];
11'h609:
    rf_filter_coeff260_b <= wdata[7:0];
11'h60a:
    rf_filter_coeff261_a <= wdata[7:0];
11'h60b:
    rf_filter_coeff261_b <= wdata[7:0];
11'h60c:
    rf_filter_coeff262_a <= wdata[7:0];
11'h60d:
    rf_filter_coeff262_b <= wdata[7:0];
11'h60e:
    rf_filter_coeff263_a <= wdata[7:0];
11'h60f:
    rf_filter_coeff263_b <= wdata[7:0];
11'h610:
    rf_filter_coeff264_a <= wdata[7:0];
11'h611:
    rf_filter_coeff264_b <= wdata[7:0];
11'h612:
    rf_filter_coeff265_a <= wdata[7:0];
11'h613:
    rf_filter_coeff265_b <= wdata[7:0];
11'h614:
    rf_filter_coeff266_a <= wdata[7:0];
11'h615:
    rf_filter_coeff266_b <= wdata[7:0];
11'h616:
    rf_filter_coeff267_a <= wdata[7:0];
11'h617:
    rf_filter_coeff267_b <= wdata[7:0];
11'h618:
    rf_filter_coeff268_a <= wdata[7:0];
11'h619:
    rf_filter_coeff268_b <= wdata[7:0];
11'h61a:
    rf_filter_coeff269_a <= wdata[7:0];
11'h61b:
    rf_filter_coeff269_b <= wdata[7:0];
11'h61c:
    rf_filter_coeff270_a <= wdata[7:0];
11'h61d:
    rf_filter_coeff270_b <= wdata[7:0];
11'h61e:
    rf_filter_coeff271_a <= wdata[7:0];
11'h61f:
    rf_filter_coeff271_b <= wdata[7:0];
```

```
11'h620:
    rf_filter_coeff272_a <= wdata[7:0];
11'h621:
    rf_filter_coeff272_b <= wdata[7:0];
11'h622:
    rf_filter_coeff273_a <= wdata[7:0];
11'h623:
    rf_filter_coeff273_b <= wdata[7:0];
11'h624:
    rf_filter_coeff274_a <= wdata[7:0];
11'h625:
    rf_filter_coeff274_b <= wdata[7:0];
11'h626:
    rf_filter_coeff275_a <= wdata[7:0];
11'h627:
    rf_filter_coeff275_b <= wdata[7:0];
11'h628:
    rf_filter_coeff276_a <= wdata[7:0];
11'h629:
    rf_filter_coeff276_b <= wdata[7:0];
11'h62a:
    rf_filter_coeff277_a <= wdata[7:0];
11'h62b:
    rf_filter_coeff277_b <= wdata[7:0];
11'h62c:
    rf_filter_coeff278_a <= wdata[7:0];
11'h62d:
    rf_filter_coeff278_b <= wdata[7:0];
11'h62e:
    rf_filter_coeff279_a <= wdata[7:0];
11'h62f:
    rf_filter_coeff279_b <= wdata[7:0];
11'h630:
    rf_filter_coeff280_a <= wdata[7:0];
11'h631:
    rf_filter_coeff280_b <= wdata[7:0];
11'h632:
    rf_filter_coeff281_a <= wdata[7:0];
11'h633:
    rf_filter_coeff281_b <= wdata[7:0];
11'h634:
    rf_filter_coeff282_a <= wdata[7:0];
11'h635:
    rf_filter_coeff282_b <= wdata[7:0];
11'h636:
    rf_filter_coeff283_a <= wdata[7:0];
11'h637:
    rf_filter_coeff283_b <= wdata[7:0];
11'h638:
    rf_filter_coeff284_a <= wdata[7:0];
11'h639:
    rf_filter_coeff284_b <= wdata[7:0];
11'h63a:
    rf_filter_coeff285_a <= wdata[7:0];
11'h63b:
    rf_filter_coeff285_b <= wdata[7:0];
11'h63c:
    rf_filter_coeff286_a <= wdata[7:0];
11'h63d:
    rf_filter_coeff286_b <= wdata[7:0];
11'h63e:
    rf_filter_coeff287_a <= wdata[7:0];
11'h63f:
    rf_filter_coeff287_b <= wdata[7:0];
11'h640:
    rf_filter_coeff288_a <= wdata[7:0];
11'h641:
    rf_filter_coeff288_b <= wdata[7:0];
11'h642:
    rf_filter_coeff289_a <= wdata[7:0];
11'h643:
    rf_filter_coeff289_b <= wdata[7:0];
11'h644:
    rf_filter_coeff290_a <= wdata[7:0];
11'h645:
    rf_filter_coeff290_b <= wdata[7:0];
11'h646:
    rf_filter_coeff291_a <= wdata[7:0];
11'h647:
    rf_filter_coeff291_b <= wdata[7:0];
11'h648:
    rf_filter_coeff292_a <= wdata[7:0];
11'h649:
    rf_filter_coeff292_b <= wdata[7:0];
11'h64a:
    rf_filter_coeff293_a <= wdata[7:0];
11'h64b:
    rf_filter_coeff293_b <= wdata[7:0];
11'h64c:
    rf_filter_coeff294_a <= wdata[7:0];
11'h64d:
    rf_filter_coeff294_b <= wdata[7:0];
11'h64e:
    rf_filter_coeff295_a <= wdata[7:0];
11'h64f:
```

```
    rf_filter_coeff295_b <= wdata[7:0];
11'h650:
    rf_filter_coeff296_a <= wdata[7:0];
11'h651:
    rf_filter_coeff296_b <= wdata[7:0];
11'h652:
    rf_filter_coeff297_a <= wdata[7:0];
11'h653:
    rf_filter_coeff297_b <= wdata[7:0];
11'h654:
    rf_filter_coeff298_a <= wdata[7:0];
11'h655:
    rf_filter_coeff298_b <= wdata[7:0];
11'h656:
    rf_filter_coeff299_a <= wdata[7:0];
11'h657:
    rf_filter_coeff299_b <= wdata[7:0];
11'h658:
    rf_filter_coeff300_a <= wdata[7:0];
11'h659:
    rf_filter_coeff300_b <= wdata[7:0];
11'h65a:
    rf_filter_coeff301_a <= wdata[7:0];
11'h65b:
    rf_filter_coeff301_b <= wdata[7:0];
11'h65c:
    rf_filter_coeff302_a <= wdata[7:0];
11'h65d:
    rf_filter_coeff302_b <= wdata[7:0];
11'h65e:
    rf_filter_coeff303_a <= wdata[7:0];
11'h65f:
    rf_filter_coeff303_b <= wdata[7:0];
11'h660:
    rf_filter_coeff304_a <= wdata[7:0];
11'h661:
    rf_filter_coeff304_b <= wdata[7:0];
11'h662:
    rf_filter_coeff305_a <= wdata[7:0];
11'h663:
    rf_filter_coeff305_b <= wdata[7:0];
11'h664:
    rf_filter_coeff306_a <= wdata[7:0];
11'h665:
    rf_filter_coeff306_b <= wdata[7:0];
11'h666:
    rf_filter_coeff307_a <= wdata[7:0];
11'h667:
    rf_filter_coeff307_b <= wdata[7:0];
11'h668:
    rf_filter_coeff308_a <= wdata[7:0];
11'h669:
    rf_filter_coeff308_b <= wdata[7:0];
11'h66a:
    rf_filter_coeff309_a <= wdata[7:0];
11'h66b:
    rf_filter_coeff309_b <= wdata[7:0];
11'h66c:
    rf_filter_coeff310_a <= wdata[7:0];
11'h66d:
    rf_filter_coeff310_b <= wdata[7:0];
11'h66e:
    rf_filter_coeff311_a <= wdata[7:0];
11'h66f:
    rf_filter_coeff311_b <= wdata[7:0];
11'h670:
    rf_filter_coeff312_a <= wdata[7:0];
11'h671:
    rf_filter_coeff312_b <= wdata[7:0];
11'h672:
    rf_filter_coeff313_a <= wdata[7:0];
11'h673:
    rf_filter_coeff313_b <= wdata[7:0];
11'h674:
    rf_filter_coeff314_a <= wdata[7:0];
11'h675:
    rf_filter_coeff314_b <= wdata[7:0];
11'h676:
    rf_filter_coeff315_a <= wdata[7:0];
11'h677:
    rf_filter_coeff315_b <= wdata[7:0];
11'h678:
    rf_filter_coeff316_a <= wdata[7:0];
11'h679:
    rf_filter_coeff316_b <= wdata[7:0];
11'h67a:
    rf_filter_coeff317_a <= wdata[7:0];
11'h67b:
    rf_filter_coeff317_b <= wdata[7:0];
11'h67c:
    rf_filter_coeff318_a <= wdata[7:0];
11'h67d:
    rf_filter_coeff318_b <= wdata[7:0];
11'h67e:
    rf_filter_coeff319_a <= wdata[7:0];
```

```
11'h67f:
    rf_filter_coeff319_b <= wdata[7:0];
11'h680:
    rf_filter_coeff320_a <= wdata[7:0];
11'h681:
    rf_filter_coeff320_b <= wdata[7:0];
11'h682:
    rf_filter_coeff321_a <= wdata[7:0];
11'h683:
    rf_filter_coeff321_b <= wdata[7:0];
11'h684:
    rf_filter_coeff322_a <= wdata[7:0];
11'h685:
    rf_filter_coeff322_b <= wdata[7:0];
11'h686:
    rf_filter_coeff323_a <= wdata[7:0];
11'h687:
    rf_filter_coeff323_b <= wdata[7:0];
11'h688:
    rf_filter_coeff324_a <= wdata[7:0];
11'h689:
    rf_filter_coeff324_b <= wdata[7:0];
11'h68a:
    rf_filter_coeff325_a <= wdata[7:0];
11'h68b:
    rf_filter_coeff325_b <= wdata[7:0];
11'h68c:
    rf_filter_coeff326_a <= wdata[7:0];
11'h68d:
    rf_filter_coeff326_b <= wdata[7:0];
11'h68e:
    rf_filter_coeff327_a <= wdata[7:0];
11'h68f:
    rf_filter_coeff327_b <= wdata[7:0];
11'h690:
    rf_filter_coeff328_a <= wdata[7:0];
11'h691:
    rf_filter_coeff328_b <= wdata[7:0];
11'h692:
    rf_filter_coeff329_a <= wdata[7:0];
11'h693:
    rf_filter_coeff329_b <= wdata[7:0];
11'h694:
    rf_filter_coeff330_a <= wdata[7:0];
11'h695:
    rf_filter_coeff330_b <= wdata[7:0];
11'h696:
    rf_filter_coeff331_a <= wdata[7:0];
11'h697:
    rf_filter_coeff331_b <= wdata[7:0];
11'h698:
    rf_filter_coeff332_a <= wdata[7:0];
11'h699:
    rf_filter_coeff332_b <= wdata[7:0];
11'h69a:
    rf_filter_coeff333_a <= wdata[7:0];
11'h69b:
    rf_filter_coeff333_b <= wdata[7:0];
11'h69c:
    rf_filter_coeff334_a <= wdata[7:0];
11'h69d:
    rf_filter_coeff334_b <= wdata[7:0];
11'h69e:
    rf_filter_coeff335_a <= wdata[7:0];
11'h69f:
    rf_filter_coeff335_b <= wdata[7:0];
11'h6a0:
    rf_filter_coeff336_a <= wdata[7:0];
11'h6a1:
    rf_filter_coeff336_b <= wdata[7:0];
11'h6a2:
    rf_filter_coeff337_a <= wdata[7:0];
11'h6a3:
    rf_filter_coeff337_b <= wdata[7:0];
11'h6a4:
    rf_filter_coeff338_a <= wdata[7:0];
11'h6a5:
    rf_filter_coeff338_b <= wdata[7:0];
11'h6a6:
    rf_filter_coeff339_a <= wdata[7:0];
11'h6a7:
    rf_filter_coeff339_b <= wdata[7:0];
11'h6a8:
    rf_filter_coeff340_a <= wdata[7:0];
11'h6a9:
    rf_filter_coeff340_b <= wdata[7:0];
11'h6aa:
    rf_filter_coeff341_a <= wdata[7:0];
11'h6ab:
    rf_filter_coeff341_b <= wdata[7:0];
11'h6ac:
    rf_filter_coeff342_a <= wdata[7:0];
11'h6ad:
    rf_filter_coeff342_b <= wdata[7:0];
11'h6ae:
```

```
    rf_filter_coeff343_a <= wdata[7:0];
11'h6af:
    rf_filter_coeff343_b <= wdata[7:0];
11'h6b0:
    rf_filter_coeff344_a <= wdata[7:0];
11'h6b1:
    rf_filter_coeff344_b <= wdata[7:0];
11'h6b2:
    rf_filter_coeff345_a <= wdata[7:0];
11'h6b3:
    rf_filter_coeff345_b <= wdata[7:0];
11'h6b4:
    rf_filter_coeff346_a <= wdata[7:0];
11'h6b5:
    rf_filter_coeff346_b <= wdata[7:0];
11'h6b6:
    rf_filter_coeff347_a <= wdata[7:0];
11'h6b7:
    rf_filter_coeff347_b <= wdata[7:0];
11'h6b8:
    rf_filter_coeff348_a <= wdata[7:0];
11'h6b9:
    rf_filter_coeff348_b <= wdata[7:0];
11'h6ba:
    rf_filter_coeff349_a <= wdata[7:0];
11'h6bb:
    rf_filter_coeff349_b <= wdata[7:0];
11'h6bc:
    rf_filter_coeff350_a <= wdata[7:0];
11'h6bd:
    rf_filter_coeff350_b <= wdata[7:0];
11'h6be:
    rf_filter_coeff351_a <= wdata[7:0];
11'h6bf:
    rf_filter_coeff351_b <= wdata[7:0];
11'h6c0:
    rf_filter_coeff352_a <= wdata[7:0];
11'h6c1:
    rf_filter_coeff352_b <= wdata[7:0];
11'h6c2:
    rf_filter_coeff353_a <= wdata[7:0];
11'h6c3:
    rf_filter_coeff353_b <= wdata[7:0];
11'h6c4:
    rf_filter_coeff354_a <= wdata[7:0];
11'h6c5:
    rf_filter_coeff354_b <= wdata[7:0];
11'h6c6:
    rf_filter_coeff355_a <= wdata[7:0];
11'h6c7:
    rf_filter_coeff355_b <= wdata[7:0];
11'h6c8:
    rf_filter_coeff356_a <= wdata[7:0];
11'h6c9:
    rf_filter_coeff356_b <= wdata[7:0];
11'h6ca:
    rf_filter_coeff357_a <= wdata[7:0];
11'h6cb:
    rf_filter_coeff357_b <= wdata[7:0];
11'h6cc:
    rf_filter_coeff358_a <= wdata[7:0];
11'h6cd:
    rf_filter_coeff358_b <= wdata[7:0];
11'h6ce:
    rf_filter_coeff359_a <= wdata[7:0];
11'h6cf:
    rf_filter_coeff359_b <= wdata[7:0];
11'h6d0:
    rf_filter_coeff360_a <= wdata[7:0];
11'h6d1:
    rf_filter_coeff360_b <= wdata[7:0];
11'h6d2:
    rf_filter_coeff361_a <= wdata[7:0];
11'h6d3:
    rf_filter_coeff361_b <= wdata[7:0];
11'h6d4:
    rf_filter_coeff362_a <= wdata[7:0];
11'h6d5:
    rf_filter_coeff362_b <= wdata[7:0];
11'h6d6:
    rf_filter_coeff363_a <= wdata[7:0];
11'h6d7:
    rf_filter_coeff363_b <= wdata[7:0];
11'h6d8:
    rf_filter_coeff364_a <= wdata[7:0];
11'h6d9:
    rf_filter_coeff364_b <= wdata[7:0];
11'h6da:
    rf_filter_coeff365_a <= wdata[7:0];
11'h6db:
    rf_filter_coeff365_b <= wdata[7:0];
11'h6dc:
    rf_filter_coeff366_a <= wdata[7:0];
11'h6dd:
    rf_filter_coeff366_b <= wdata[7:0];
```

```
11'h6de:
    rf_filter_coeff367_a <= wdata[7:0];
11'h6df:
    rf_filter_coeff367_b <= wdata[7:0];
11'h6e0:
    rf_filter_coeff368_a <= wdata[7:0];
11'h6e1:
    rf_filter_coeff368_b <= wdata[7:0];
11'h6e2:
    rf_filter_coeff369_a <= wdata[7:0];
11'h6e3:
    rf_filter_coeff369_b <= wdata[7:0];
11'h6e4:
    rf_filter_coeff370_a <= wdata[7:0];
11'h6e5:
    rf_filter_coeff370_b <= wdata[7:0];
11'h6e6:
    rf_filter_coeff371_a <= wdata[7:0];
11'h6e7:
    rf_filter_coeff371_b <= wdata[7:0];
11'h6e8:
    rf_filter_coeff372_a <= wdata[7:0];
11'h6e9:
    rf_filter_coeff372_b <= wdata[7:0];
11'h6ea:
    rf_filter_coeff373_a <= wdata[7:0];
11'h6eb:
    rf_filter_coeff373_b <= wdata[7:0];
11'h6ec:
    rf_filter_coeff374_a <= wdata[7:0];
11'h6ed:
    rf_filter_coeff374_b <= wdata[7:0];
11'h6ee:
    rf_filter_coeff375_a <= wdata[7:0];
11'h6ef:
    rf_filter_coeff375_b <= wdata[7:0];
11'h6f0:
    rf_filter_coeff376_a <= wdata[7:0];
11'h6f1:
    rf_filter_coeff376_b <= wdata[7:0];
11'h6f2:
    rf_filter_coeff377_a <= wdata[7:0];
11'h6f3:
    rf_filter_coeff377_b <= wdata[7:0];
11'h6f4:
    rf_filter_coeff378_a <= wdata[7:0];
11'h6f5:
    rf_filter_coeff378_b <= wdata[7:0];
11'h6f6:
    rf_filter_coeff379_a <= wdata[7:0];
11'h6f7:
    rf_filter_coeff379_b <= wdata[7:0];
11'h6f8:
    rf_filter_coeff380_a <= wdata[7:0];
11'h6f9:
    rf_filter_coeff380_b <= wdata[7:0];
11'h6fa:
    rf_filter_coeff381_a <= wdata[7:0];
11'h6fb:
    rf_filter_coeff381_b <= wdata[7:0];
11'h6fc:
    rf_filter_coeff382_a <= wdata[7:0];
11'h6fd:
    rf_filter_coeff382_b <= wdata[7:0];
11'h6fe:
    rf_filter_coeff383_a <= wdata[7:0];
11'h6ff:
    rf_filter_coeff383_b <= wdata[7:0];
11'h700:
    rf_filter_coeff384_a <= wdata[7:0];
11'h701:
    rf_filter_coeff384_b <= wdata[7:0];
11'h702:
    rf_filter_coeff385_a <= wdata[7:0];
11'h703:
    rf_filter_coeff385_b <= wdata[7:0];
11'h704:
    rf_filter_coeff386_a <= wdata[7:0];
11'h705:
    rf_filter_coeff386_b <= wdata[7:0];
11'h706:
    rf_filter_coeff387_a <= wdata[7:0];
11'h707:
    rf_filter_coeff387_b <= wdata[7:0];
11'h708:
    rf_filter_coeff388_a <= wdata[7:0];
11'h709:
    rf_filter_coeff388_b <= wdata[7:0];
11'h70a:
    rf_filter_coeff389_a <= wdata[7:0];
11'h70b:
    rf_filter_coeff389_b <= wdata[7:0];
11'h70c:
    rf_filter_coeff390_a <= wdata[7:0];
11'h70d:
```

```
    rf_filter_coeff390_b <= wdata[7:0];
11'h70e:
    rf_filter_coeff391_a <= wdata[7:0];
11'h70f:
    rf_filter_coeff391_b <= wdata[7:0];
11'h710:
    rf_filter_coeff392_a <= wdata[7:0];
11'h711:
    rf_filter_coeff392_b <= wdata[7:0];
11'h712:
    rf_filter_coeff393_a <= wdata[7:0];
11'h713:
    rf_filter_coeff393_b <= wdata[7:0];
11'h714:
    rf_filter_coeff394_a <= wdata[7:0];
11'h715:
    rf_filter_coeff394_b <= wdata[7:0];
11'h716:
    rf_filter_coeff395_a <= wdata[7:0];
11'h717:
    rf_filter_coeff395_b <= wdata[7:0];
11'h718:
    rf_filter_coeff396_a <= wdata[7:0];
11'h719:
    rf_filter_coeff396_b <= wdata[7:0];
11'h71a:
    rf_filter_coeff397_a <= wdata[7:0];
11'h71b:
    rf_filter_coeff397_b <= wdata[7:0];
11'h71c:
    rf_filter_coeff398_a <= wdata[7:0];
11'h71d:
    rf_filter_coeff398_b <= wdata[7:0];
11'h71e:
    rf_filter_coeff399_a <= wdata[7:0];
11'h71f:
    rf_filter_coeff399_b <= wdata[7:0];
11'h720:
    rf_filter_coeff400_a <= wdata[7:0];
11'h721:
    rf_filter_coeff400_b <= wdata[7:0];
11'h722:
    rf_filter_coeff401_a <= wdata[7:0];
11'h723:
    rf_filter_coeff401_b <= wdata[7:0];
11'h724:
    rf_filter_coeff402_a <= wdata[7:0];
11'h725:
    rf_filter_coeff402_b <= wdata[7:0];
11'h726:
    rf_filter_coeff403_a <= wdata[7:0];
11'h727:
    rf_filter_coeff403_b <= wdata[7:0];
11'h728:
    rf_filter_coeff404_a <= wdata[7:0];
11'h729:
    rf_filter_coeff404_b <= wdata[7:0];
11'h72a:
    rf_filter_coeff405_a <= wdata[7:0];
11'h72b:
    rf_filter_coeff405_b <= wdata[7:0];
11'h72c:
    rf_filter_coeff406_a <= wdata[7:0];
11'h72d:
    rf_filter_coeff406_b <= wdata[7:0];
11'h72e:
    rf_filter_coeff407_a <= wdata[7:0];
11'h72f:
    rf_filter_coeff407_b <= wdata[7:0];
11'h730:
    rf_filter_coeff408_a <= wdata[7:0];
11'h731:
    rf_filter_coeff408_b <= wdata[7:0];
11'h732:
    rf_filter_coeff409_a <= wdata[7:0];
11'h733:
    rf_filter_coeff409_b <= wdata[7:0];
11'h734:
    rf_filter_coeff410_a <= wdata[7:0];
11'h735:
    rf_filter_coeff410_b <= wdata[7:0];
11'h736:
    rf_filter_coeff411_a <= wdata[7:0];
11'h737:
    rf_filter_coeff411_b <= wdata[7:0];
11'h738:
    rf_filter_coeff412_a <= wdata[7:0];
11'h739:
    rf_filter_coeff412_b <= wdata[7:0];
11'h73a:
    rf_filter_coeff413_a <= wdata[7:0];
11'h73b:
    rf_filter_coeff413_b <= wdata[7:0];
11'h73c:
    rf_filter_coeff414_a <= wdata[7:0];
```

```
11'h73d:
    rf_filter_coeff414_b <= wdata[7:0];
11'h73e:
    rf_filter_coeff415_a <= wdata[7:0];
11'h73f:
    rf_filter_coeff415_b <= wdata[7:0];
11'h740:
    rf_filter_coeff416_a <= wdata[7:0];
11'h741:
    rf_filter_coeff416_b <= wdata[7:0];
11'h742:
    rf_filter_coeff417_a <= wdata[7:0];
11'h743:
    rf_filter_coeff417_b <= wdata[7:0];
11'h744:
    rf_filter_coeff418_a <= wdata[7:0];
11'h745:
    rf_filter_coeff418_b <= wdata[7:0];
11'h746:
    rf_filter_coeff419_a <= wdata[7:0];
11'h747:
    rf_filter_coeff419_b <= wdata[7:0];
11'h748:
    rf_filter_coeff420_a <= wdata[7:0];
11'h749:
    rf_filter_coeff420_b <= wdata[7:0];
11'h74a:
    rf_filter_coeff421_a <= wdata[7:0];
11'h74b:
    rf_filter_coeff421_b <= wdata[7:0];
11'h74c:
    rf_filter_coeff422_a <= wdata[7:0];
11'h74d:
    rf_filter_coeff422_b <= wdata[7:0];
11'h74e:
    rf_filter_coeff423_a <= wdata[7:0];
11'h74f:
    rf_filter_coeff423_b <= wdata[7:0];
11'h750:
    rf_filter_coeff424_a <= wdata[7:0];
11'h751:
    rf_filter_coeff424_b <= wdata[7:0];
11'h752:
    rf_filter_coeff425_a <= wdata[7:0];
11'h753:
    rf_filter_coeff425_b <= wdata[7:0];
11'h754:
    rf_filter_coeff426_a <= wdata[7:0];
11'h755:
    rf_filter_coeff426_b <= wdata[7:0];
11'h756:
    rf_filter_coeff427_a <= wdata[7:0];
11'h757:
    rf_filter_coeff427_b <= wdata[7:0];
11'h758:
    rf_filter_coeff428_a <= wdata[7:0];
11'h759:
    rf_filter_coeff428_b <= wdata[7:0];
11'h75a:
    rf_filter_coeff429_a <= wdata[7:0];
11'h75b:
    rf_filter_coeff429_b <= wdata[7:0];
11'h75c:
    rf_filter_coeff430_a <= wdata[7:0];
11'h75d:
    rf_filter_coeff430_b <= wdata[7:0];
11'h75e:
    rf_filter_coeff431_a <= wdata[7:0];
11'h75f:
    rf_filter_coeff431_b <= wdata[7:0];
11'h760:
    rf_filter_coeff432_a <= wdata[7:0];
11'h761:
    rf_filter_coeff432_b <= wdata[7:0];
11'h762:
    rf_filter_coeff433_a <= wdata[7:0];
11'h763:
    rf_filter_coeff433_b <= wdata[7:0];
11'h764:
    rf_filter_coeff434_a <= wdata[7:0];
11'h765:
    rf_filter_coeff434_b <= wdata[7:0];
11'h766:
    rf_filter_coeff435_a <= wdata[7:0];
11'h767:
    rf_filter_coeff435_b <= wdata[7:0];
11'h768:
    rf_filter_coeff436_a <= wdata[7:0];
11'h769:
    rf_filter_coeff436_b <= wdata[7:0];
11'h76a:
    rf_filter_coeff437_a <= wdata[7:0];
11'h76b:
    rf_filter_coeff437_b <= wdata[7:0];
11'h76c:
```

```
    rf_filter_coeff438_a <= wdata[7:0];
11'h76d:
    rf_filter_coeff438_b <= wdata[7:0];
11'h76e:
    rf_filter_coeff439_a <= wdata[7:0];
11'h76f:
    rf_filter_coeff439_b <= wdata[7:0];
11'h770:
    rf_filter_coeff440_a <= wdata[7:0];
11'h771:
    rf_filter_coeff440_b <= wdata[7:0];
11'h772:
    rf_filter_coeff441_a <= wdata[7:0];
11'h773:
    rf_filter_coeff441_b <= wdata[7:0];
11'h774:
    rf_filter_coeff442_a <= wdata[7:0];
11'h775:
    rf_filter_coeff442_b <= wdata[7:0];
11'h776:
    rf_filter_coeff443_a <= wdata[7:0];
11'h777:
    rf_filter_coeff443_b <= wdata[7:0];
11'h778:
    rf_filter_coeff444_a <= wdata[7:0];
11'h779:
    rf_filter_coeff444_b <= wdata[7:0];
11'h77a:
    rf_filter_coeff445_a <= wdata[7:0];
11'h77b:
    rf_filter_coeff445_b <= wdata[7:0];
11'h77c:
    rf_filter_coeff446_a <= wdata[7:0];
11'h77d:
    rf_filter_coeff446_b <= wdata[7:0];
11'h77e:
    rf_filter_coeff447_a <= wdata[7:0];
11'h77f:
    rf_filter_coeff447_b <= wdata[7:0];
11'h780:
    rf_filter_coeff448_a <= wdata[7:0];
11'h781:
    rf_filter_coeff448_b <= wdata[7:0];
11'h782:
    rf_filter_coeff449_a <= wdata[7:0];
11'h783:
    rf_filter_coeff449_b <= wdata[7:0];
11'h784:
    rf_filter_coeff450_a <= wdata[7:0];
11'h785:
    rf_filter_coeff450_b <= wdata[7:0];
11'h786:
    rf_filter_coeff451_a <= wdata[7:0];
11'h787:
    rf_filter_coeff451_b <= wdata[7:0];
11'h788:
    rf_filter_coeff452_a <= wdata[7:0];
11'h789:
    rf_filter_coeff452_b <= wdata[7:0];
11'h78a:
    rf_filter_coeff453_a <= wdata[7:0];
11'h78b:
    rf_filter_coeff453_b <= wdata[7:0];
11'h78c:
    rf_filter_coeff454_a <= wdata[7:0];
11'h78d:
    rf_filter_coeff454_b <= wdata[7:0];
11'h78e:
    rf_filter_coeff455_a <= wdata[7:0];
11'h78f:
    rf_filter_coeff455_b <= wdata[7:0];
11'h790:
    rf_filter_coeff456_a <= wdata[7:0];
11'h791:
    rf_filter_coeff456_b <= wdata[7:0];
11'h792:
    rf_filter_coeff457_a <= wdata[7:0];
11'h793:
    rf_filter_coeff457_b <= wdata[7:0];
11'h794:
    rf_filter_coeff458_a <= wdata[7:0];
11'h795:
    rf_filter_coeff458_b <= wdata[7:0];
11'h796:
    rf_filter_coeff459_a <= wdata[7:0];
11'h797:
    rf_filter_coeff459_b <= wdata[7:0];
11'h798:
    rf_filter_coeff460_a <= wdata[7:0];
11'h799:
    rf_filter_coeff460_b <= wdata[7:0];
11'h79a:
    rf_filter_coeff461_a <= wdata[7:0];
11'h79b:
    rf_filter_coeff461_b <= wdata[7:0];
```

```
11'h79c:
    rf_filter_coeff462_a <= wdata[7:0];
11'h79d:
    rf_filter_coeff462_b <= wdata[7:0];
11'h79e:
    rf_filter_coeff463_a <= wdata[7:0];
11'h79f:
    rf_filter_coeff463_b <= wdata[7:0];
11'h7a0:
    rf_filter_coeff464_a <= wdata[7:0];
11'h7a1:
    rf_filter_coeff464_b <= wdata[7:0];
11'h7a2:
    rf_filter_coeff465_a <= wdata[7:0];
11'h7a3:
    rf_filter_coeff465_b <= wdata[7:0];
11'h7a4:
    rf_filter_coeff466_a <= wdata[7:0];
11'h7a5:
    rf_filter_coeff466_b <= wdata[7:0];
11'h7a6:
    rf_filter_coeff467_a <= wdata[7:0];
11'h7a7:
    rf_filter_coeff467_b <= wdata[7:0];
11'h7a8:
    rf_filter_coeff468_a <= wdata[7:0];
11'h7a9:
    rf_filter_coeff468_b <= wdata[7:0];
11'h7aa:
    rf_filter_coeff469_a <= wdata[7:0];
11'h7ab:
    rf_filter_coeff469_b <= wdata[7:0];
11'h7ac:
    rf_filter_coeff470_a <= wdata[7:0];
11'h7ad:
    rf_filter_coeff470_b <= wdata[7:0];
11'h7ae:
    rf_filter_coeff471_a <= wdata[7:0];
11'h7af:
    rf_filter_coeff471_b <= wdata[7:0];
11'h7b0:
    rf_filter_coeff472_a <= wdata[7:0];
11'h7b1:
    rf_filter_coeff472_b <= wdata[7:0];
11'h7b2:
    rf_filter_coeff473_a <= wdata[7:0];
11'h7b3:
    rf_filter_coeff473_b <= wdata[7:0];
11'h7b4:
    rf_filter_coeff474_a <= wdata[7:0];
11'h7b5:
    rf_filter_coeff474_b <= wdata[7:0];
11'h7b6:
    rf_filter_coeff475_a <= wdata[7:0];
11'h7b7:
    rf_filter_coeff475_b <= wdata[7:0];
11'h7b8:
    rf_filter_coeff476_a <= wdata[7:0];
11'h7b9:
    rf_filter_coeff476_b <= wdata[7:0];
11'h7ba:
    rf_filter_coeff477_a <= wdata[7:0];
11'h7bb:
    rf_filter_coeff477_b <= wdata[7:0];
11'h7bc:
    rf_filter_coeff478_a <= wdata[7:0];
11'h7bd:
    rf_filter_coeff478_b <= wdata[7:0];
11'h7be:
    rf_filter_coeff479_a <= wdata[7:0];
11'h7bf:
    rf_filter_coeff479_b <= wdata[7:0];
11'h7c0:
    rf_filter_coeff480_a <= wdata[7:0];
11'h7c1:
    rf_filter_coeff480_b <= wdata[7:0];
11'h7c2:
    rf_filter_coeff481_a <= wdata[7:0];
11'h7c3:
    rf_filter_coeff481_b <= wdata[7:0];
11'h7c4:
    rf_filter_coeff482_a <= wdata[7:0];
11'h7c5:
    rf_filter_coeff482_b <= wdata[7:0];
11'h7c6:
    rf_filter_coeff483_a <= wdata[7:0];
11'h7c7:
    rf_filter_coeff483_b <= wdata[7:0];
11'h7c8:
    rf_filter_coeff484_a <= wdata[7:0];
11'h7c9:
    rf_filter_coeff484_b <= wdata[7:0];
11'h7ca:
    rf_filter_coeff485_a <= wdata[7:0];
11'h7cb:
```

```
    rf_filter_coeff485_b <= wdata[7:0];
11'h7cc:
    rf_filter_coeff486_a <= wdata[7:0];
11'h7cd:
    rf_filter_coeff486_b <= wdata[7:0];
11'h7ce:
    rf_filter_coeff487_a <= wdata[7:0];
11'h7cf:
    rf_filter_coeff487_b <= wdata[7:0];
11'h7d0:
    rf_filter_coeff488_a <= wdata[7:0];
11'h7d1:
    rf_filter_coeff488_b <= wdata[7:0];
11'h7d2:
    rf_filter_coeff489_a <= wdata[7:0];
11'h7d3:
    rf_filter_coeff489_b <= wdata[7:0];
11'h7d4:
    rf_filter_coeff490_a <= wdata[7:0];
11'h7d5:
    rf_filter_coeff490_b <= wdata[7:0];
11'h7d6:
    rf_filter_coeff491_a <= wdata[7:0];
11'h7d7:
    rf_filter_coeff491_b <= wdata[7:0];
11'h7d8:
    rf_filter_coeff492_a <= wdata[7:0];
11'h7d9:
    rf_filter_coeff492_b <= wdata[7:0];
11'h7da:
    rf_filter_coeff493_a <= wdata[7:0];
11'h7db:
    rf_filter_coeff493_b <= wdata[7:0];
11'h7dc:
    rf_filter_coeff494_a <= wdata[7:0];
11'h7dd:
    rf_filter_coeff494_b <= wdata[7:0];
11'h7de:
    rf_filter_coeff495_a <= wdata[7:0];
11'h7df:
    rf_filter_coeff495_b <= wdata[7:0];
11'h7e0:
    rf_filter_coeff496_a <= wdata[7:0];
11'h7e1:
    rf_filter_coeff496_b <= wdata[7:0];
11'h7e2:
    rf_filter_coeff497_a <= wdata[7:0];
11'h7e3:
    rf_filter_coeff497_b <= wdata[7:0];
11'h7e4:
    rf_filter_coeff498_a <= wdata[7:0];
11'h7e5:
    rf_filter_coeff498_b <= wdata[7:0];
11'h7e6:
    rf_filter_coeff499_a <= wdata[7:0];
11'h7e7:
    rf_filter_coeff499_b <= wdata[7:0];
11'h7e8:
    rf_filter_coeff500_a <= wdata[7:0];
11'h7e9:
    rf_filter_coeff500_b <= wdata[7:0];
11'h7ea:
    rf_filter_coeff501_a <= wdata[7:0];
11'h7eb:
    rf_filter_coeff501_b <= wdata[7:0];
11'h7ec:
    rf_filter_coeff502_a <= wdata[7:0];
11'h7ed:
    rf_filter_coeff502_b <= wdata[7:0];
11'h7ee:
    rf_filter_coeff503_a <= wdata[7:0];
11'h7ef:
    rf_filter_coeff503_b <= wdata[7:0];
11'h7f0:
    rf_filter_coeff504_a <= wdata[7:0];
11'h7f1:
    rf_filter_coeff504_b <= wdata[7:0];
11'h7f2:
    rf_filter_coeff505_a <= wdata[7:0];
11'h7f3:
    rf_filter_coeff505_b <= wdata[7:0];
11'h7f4:
    rf_filter_coeff506_a <= wdata[7:0];
11'h7f5:
    rf_filter_coeff506_b <= wdata[7:0];
11'h7f6:
    rf_filter_coeff507_a <= wdata[7:0];
11'h7f7:
    rf_filter_coeff507_b <= wdata[7:0];
11'h7f8:
    rf_filter_coeff508_a <= wdata[7:0];
11'h7f9:
    rf_filter_coeff508_b <= wdata[7:0];
11'h7fa:
    rf_filter_coeff509_a <= wdata[7:0];
```

```
                    11'h7fb:
                        rf_filter_coeff509_b <= wdata[7:0];
                    11'h7fc:
                        rf_filter_coeff510_a <= wdata[7:0];
                    11'h7fd:
                        rf_filter_coeff510_b <= wdata[7:0];
                    11'h7fe:
                        rf_filter_coeff511_a <= wdata[7:0];
                    11'h7ff:
                        rf_filter_coeff511_b <= wdata[7:0];
                endcase
            end

    always @ (posedge clk)
            begin
                                                if (~rst_n)
                                                            begin
                                                            rdata <= 8'b00000000;
                                                            end

                                                            rxfc <= wxfc & ~w_enable; // should not assert rxfc for every
transaction, only for an actual read

                // Given the address, the signals are assigned to their correlated bits of data
                                                            if(wxfc & ~w_enable);
                                                            begin
                                                            case(addr)
                    11'h004: begin
                        rdata[0] <= rf_i2si_bist_en;
                        rdata[3:1] <= rf_filter_shift;
                        rdata[4] <= rf_filter_clip_en;
                        rdata[5] <= rf_i2si_en;
                        end
                    11'h008: begin
                        rdata[0] <= trig_fifo_overrun;
                        rdata[1] <= ro_fifo_overrun;
                        rdata[2] <= trig_fifo_underrun;
                        rdata[3] <= ro_fifo_underrun;
                                                                                                                rdata[4] <=
trig_filter_ovf_flag_clear;
                                                                                                                rdata[5] <=
ro_filter_ovf_flag;
                        end
                    11'h00c:
                        rdata[7:0] <= rf_i2si_bist_incr;
                    11'h00d:
                        rdata[7:0] <= rf_i2si_bist_start_val_a;
                    11'h00e:
                        rdata[3:0] <= rf_i2si_bist_start_val_b;
                                                                        11'h010:
                        rdata[7:0] <= rf_i2si_bist_upper_limit_a;
                    11'h011:
                        rdata[3:0] <= rf_i2si_bist_upper_limit_b;
                    11'h400:
                        rdata[7:0] <= rf_filter_coeff0_a;
                    11'h401:
                        rdata[7:0] <= rf_filter_coeff0_b;
                    11'h402:
                        rdata[7:0] <= rf_filter_coeff1_a;
                    11'h403:
                        rdata[7:0] <= rf_filter_coeff1_b;
                    11'h404:
                        rdata[7:0] <= rf_filter_coeff2_a;
                    11'h405:
                        rdata[7:0] <= rf_filter_coeff2_b;
                    11'h406:
                        rdata[7:0] <= rf_filter_coeff3_a;
                    11'h407:
                        rdata[7:0] <= rf_filter_coeff3_b;
                    11'h408:
                        rdata[7:0] <= rf_filter_coeff4_a;
                    11'h409:
                        rdata[7:0] <= rf_filter_coeff4_b;
                    11'h40a:
                        rdata[7:0] <= rf_filter_coeff5_a;
                    11'h40b:
                        rdata[7:0] <= rf_filter_coeff5_b;
                    11'h40c:
                        rdata[7:0] <= rf_filter_coeff6_a;
                    11'h40d:
                        rdata[7:0] <= rf_filter_coeff6_b;
                    11'h40e:
                        rdata[7:0] <= rf_filter_coeff7_a;
                    11'h40f:
                        rdata[7:0] <= rf_filter_coeff7_b;
                    11'h410:
                        rdata[7:0] <= rf_filter_coeff8_a;
                    11'h411:
                        rdata[7:0] <= rf_filter_coeff8_b;
                    11'h412:
                        rdata[7:0] <= rf_filter_coeff9_a;
                    11'h413:
                        rdata[7:0] <= rf_filter_coeff9_b;
                    11'h414:
                        rdata[7:0] <= rf_filter_coeff10_a;
```

```
11'h415:
    rdata[7:0] <= rf_filter_coeff10_b;
11'h416:
    rdata[7:0] <= rf_filter_coeff11_a;
11'h417:
    rdata[7:0] <= rf_filter_coeff11_b;
11'h418:
    rdata[7:0] <= rf_filter_coeff12_a;
11'h419:
    rdata[7:0] <= rf_filter_coeff12_b;
11'h41a:
    rdata[7:0] <= rf_filter_coeff13_a;
11'h41b:
    rdata[7:0] <= rf_filter_coeff13_b;
11'h41c:
    rdata[7:0] <= rf_filter_coeff14_a;
11'h41d:
    rdata[7:0] <= rf_filter_coeff14_b;
11'h41e:
    rdata[7:0] <= rf_filter_coeff15_a;
11'h41f:
    rdata[7:0] <= rf_filter_coeff15_b;
11'h420:
    rdata[7:0] <= rf_filter_coeff16_a;
11'h421:
    rdata[7:0] <= rf_filter_coeff16_b;
11'h422:
    rdata[7:0] <= rf_filter_coeff17_a;
11'h423:
    rdata[7:0] <= rf_filter_coeff17_b;
11'h424:
    rdata[7:0] <= rf_filter_coeff18_a;
11'h425:
    rdata[7:0] <= rf_filter_coeff18_b;
11'h426:
    rdata[7:0] <= rf_filter_coeff19_a;
11'h427:
    rdata[7:0] <= rf_filter_coeff19_b;
11'h428:
    rdata[7:0] <= rf_filter_coeff20_a;
11'h429:
    rdata[7:0] <= rf_filter_coeff20_b;
11'h42a:
    rdata[7:0] <= rf_filter_coeff21_a;
11'h42b:
    rdata[7:0] <= rf_filter_coeff21_b;
11'h42c:
    rdata[7:0] <= rf_filter_coeff22_a;
11'h42d:
    rdata[7:0] <= rf_filter_coeff22_b;
11'h42e:
    rdata[7:0] <= rf_filter_coeff23_a;
11'h42f:
    rdata[7:0] <= rf_filter_coeff23_b;
11'h430:
    rdata[7:0] <= rf_filter_coeff24_a;
11'h431:
    rdata[7:0] <= rf_filter_coeff24_b;
11'h432:
    rdata[7:0] <= rf_filter_coeff25_a;
11'h433:
    rdata[7:0] <= rf_filter_coeff25_b;
11'h434:
    rdata[7:0] <= rf_filter_coeff26_a;
11'h435:
    rdata[7:0] <= rf_filter_coeff26_b;
11'h436:
    rdata[7:0] <= rf_filter_coeff27_a;
11'h437:
    rdata[7:0] <= rf_filter_coeff27_b;
11'h438:
    rdata[7:0] <= rf_filter_coeff28_a;
11'h439:
    rdata[7:0] <= rf_filter_coeff28_b;
11'h43a:
    rdata[7:0] <= rf_filter_coeff29_a;
11'h43b:
    rdata[7:0] <= rf_filter_coeff29_b;
11'h43c:
    rdata[7:0] <= rf_filter_coeff30_a;
11'h43d:
    rdata[7:0] <= rf_filter_coeff30_b;
11'h43e:
    rdata[7:0] <= rf_filter_coeff31_a;
11'h43f:
    rdata[7:0] <= rf_filter_coeff31_b;
11'h440:
    rdata[7:0] <= rf_filter_coeff32_a;
11'h441:
    rdata[7:0] <= rf_filter_coeff32_b;
11'h442:
    rdata[7:0] <= rf_filter_coeff33_a;
11'h443:
    rdata[7:0] <= rf_filter_coeff33_b;
11'h444:
```

```
    rdata[7:0] <= rf_filter_coeff34_a;
11'h445:
    rdata[7:0] <= rf_filter_coeff34_b;
11'h446:
    rdata[7:0] <= rf_filter_coeff35_a;
11'h447:
    rdata[7:0] <= rf_filter_coeff35_b;
11'h448:
    rdata[7:0] <= rf_filter_coeff36_a;
11'h449:
    rdata[7:0] <= rf_filter_coeff36_b;
11'h44a:
    rdata[7:0] <= rf_filter_coeff37_a;
11'h44b:
    rdata[7:0] <= rf_filter_coeff37_b;
11'h44c:
    rdata[7:0] <= rf_filter_coeff38_a;
11'h44d:
    rdata[7:0] <= rf_filter_coeff38_b;
11'h44e:
    rdata[7:0] <= rf_filter_coeff39_a;
11'h44f:
    rdata[7:0] <= rf_filter_coeff39_b;
11'h450:
    rdata[7:0] <= rf_filter_coeff40_a;
11'h451:
    rdata[7:0] <= rf_filter_coeff40_b;
11'h452:
    rdata[7:0] <= rf_filter_coeff41_a;
11'h453:
    rdata[7:0] <= rf_filter_coeff41_b;
11'h454:
    rdata[7:0] <= rf_filter_coeff42_a;
11'h455:
    rdata[7:0] <= rf_filter_coeff42_b;
11'h456:
    rdata[7:0] <= rf_filter_coeff43_a;
11'h457:
    rdata[7:0] <= rf_filter_coeff43_b;
11'h458:
    rdata[7:0] <= rf_filter_coeff44_a;
11'h459:
    rdata[7:0] <= rf_filter_coeff44_b;
11'h45a:
    rdata[7:0] <= rf_filter_coeff45_a;
11'h45b:
    rdata[7:0] <= rf_filter_coeff45_b;
11'h45c:
    rdata[7:0] <= rf_filter_coeff46_a;
11'h45d:
    rdata[7:0] <= rf_filter_coeff46_b;
11'h45e:
    rdata[7:0] <= rf_filter_coeff47_a;
11'h45f:
    rdata[7:0] <= rf_filter_coeff47_b;
11'h460:
    rdata[7:0] <= rf_filter_coeff48_a;
11'h461:
    rdata[7:0] <= rf_filter_coeff48_b;
11'h462:
    rdata[7:0] <= rf_filter_coeff49_a;
11'h463:
    rdata[7:0] <= rf_filter_coeff49_b;
11'h464:
    rdata[7:0] <= rf_filter_coeff50_a;
11'h465:
    rdata[7:0] <= rf_filter_coeff50_b;
11'h466:
    rdata[7:0] <= rf_filter_coeff51_a;
11'h467:
    rdata[7:0] <= rf_filter_coeff51_b;
11'h468:
    rdata[7:0] <= rf_filter_coeff52_a;
11'h469:
    rdata[7:0] <= rf_filter_coeff52_b;
11'h46a:
    rdata[7:0] <= rf_filter_coeff53_a;
11'h46b:
    rdata[7:0] <= rf_filter_coeff53_b;
11'h46c:
    rdata[7:0] <= rf_filter_coeff54_a;
11'h46d:
    rdata[7:0] <= rf_filter_coeff54_b;
11'h46e:
    rdata[7:0] <= rf_filter_coeff55_a;
11'h46f:
    rdata[7:0] <= rf_filter_coeff55_b;
11'h470:
    rdata[7:0] <= rf_filter_coeff56_a;
11'h471:
    rdata[7:0] <= rf_filter_coeff56_b;
11'h472:
    rdata[7:0] <= rf_filter_coeff57_a;
11'h473:
    rdata[7:0] <= rf_filter_coeff57_b;
```

```
11'h474:
    rdata[7:0] <= rf_filter_coeff58_a;
11'h475:
    rdata[7:0] <= rf_filter_coeff58_b;
11'h476:
    rdata[7:0] <= rf_filter_coeff59_a;
11'h477:
    rdata[7:0] <= rf_filter_coeff59_b;
11'h478:
    rdata[7:0] <= rf_filter_coeff60_a;
11'h479:
    rdata[7:0] <= rf_filter_coeff60_b;
11'h47a:
    rdata[7:0] <= rf_filter_coeff61_a;
11'h47b:
    rdata[7:0] <= rf_filter_coeff61_b;
11'h47c:
    rdata[7:0] <= rf_filter_coeff62_a;
11'h47d:
    rdata[7:0] <= rf_filter_coeff62_b;
11'h47e:
    rdata[7:0] <= rf_filter_coeff63_a;
11'h47f:
    rdata[7:0] <= rf_filter_coeff63_b;
11'h480:
    rdata[7:0] <= rf_filter_coeff64_a;
11'h481:
    rdata[7:0] <= rf_filter_coeff64_b;
11'h482:
    rdata[7:0] <= rf_filter_coeff65_a;
11'h483:
    rdata[7:0] <= rf_filter_coeff65_b;
11'h484:
    rdata[7:0] <= rf_filter_coeff66_a;
11'h485:
    rdata[7:0] <= rf_filter_coeff66_b;
11'h486:
    rdata[7:0] <= rf_filter_coeff67_a;
11'h487:
    rdata[7:0] <= rf_filter_coeff67_b;
11'h488:
    rdata[7:0] <= rf_filter_coeff68_a;
11'h489:
    rdata[7:0] <= rf_filter_coeff68_b;
11'h48a:
    rdata[7:0] <= rf_filter_coeff69_a;
11'h48b:
    rdata[7:0] <= rf_filter_coeff69_b;
11'h48c:
    rdata[7:0] <= rf_filter_coeff70_a;
11'h48d:
    rdata[7:0] <= rf_filter_coeff70_b;
11'h48e:
    rdata[7:0] <= rf_filter_coeff71_a;
11'h48f:
    rdata[7:0] <= rf_filter_coeff71_b;
11'h490:
    rdata[7:0] <= rf_filter_coeff72_a;
11'h491:
    rdata[7:0] <= rf_filter_coeff72_b;
11'h492:
    rdata[7:0] <= rf_filter_coeff73_a;
11'h493:
    rdata[7:0] <= rf_filter_coeff73_b;
11'h494:
    rdata[7:0] <= rf_filter_coeff74_a;
11'h495:
    rdata[7:0] <= rf_filter_coeff74_b;
11'h496:
    rdata[7:0] <= rf_filter_coeff75_a;
11'h497:
    rdata[7:0] <= rf_filter_coeff75_b;
11'h498:
    rdata[7:0] <= rf_filter_coeff76_a;
11'h499:
    rdata[7:0] <= rf_filter_coeff76_b;
11'h49a:
    rdata[7:0] <= rf_filter_coeff77_a;
11'h49b:
    rdata[7:0] <= rf_filter_coeff77_b;
11'h49c:
    rdata[7:0] <= rf_filter_coeff78_a;
11'h49d:
    rdata[7:0] <= rf_filter_coeff78_b;
11'h49e:
    rdata[7:0] <= rf_filter_coeff79_a;
11'h49f:
    rdata[7:0] <= rf_filter_coeff79_b;
11'h4a0:
    rdata[7:0] <= rf_filter_coeff80_a;
11'h4a1:
    rdata[7:0] <= rf_filter_coeff80_b;
11'h4a2:
    rdata[7:0] <= rf_filter_coeff81_a;
11'h4a3:
```

```
    rdata[7:0] <= rf_filter_coeff81_b;
11'h4a4:
    rdata[7:0] <= rf_filter_coeff82_a;
11'h4a5:
    rdata[7:0] <= rf_filter_coeff82_b;
11'h4a6:
    rdata[7:0] <= rf_filter_coeff83_a;
11'h4a7:
    rdata[7:0] <= rf_filter_coeff83_b;
11'h4a8:
    rdata[7:0] <= rf_filter_coeff84_a;
11'h4a9:
    rdata[7:0] <= rf_filter_coeff84_b;
11'h4aa:
    rdata[7:0] <= rf_filter_coeff85_a;
11'h4ab:
    rdata[7:0] <= rf_filter_coeff85_b;
11'h4ac:
    rdata[7:0] <= rf_filter_coeff86_a;
11'h4ad:
    rdata[7:0] <= rf_filter_coeff86_b;
11'h4ae:
    rdata[7:0] <= rf_filter_coeff87_a;
11'h4af:
    rdata[7:0] <= rf_filter_coeff87_b;
11'h4b0:
    rdata[7:0] <= rf_filter_coeff88_a;
11'h4b1:
    rdata[7:0] <= rf_filter_coeff88_b;
11'h4b2:
    rdata[7:0] <= rf_filter_coeff89_a;
11'h4b3:
    rdata[7:0] <= rf_filter_coeff89_b;
11'h4b4:
    rdata[7:0] <= rf_filter_coeff90_a;
11'h4b5:
    rdata[7:0] <= rf_filter_coeff90_b;
11'h4b6:
    rdata[7:0] <= rf_filter_coeff91_a;
11'h4b7:
    rdata[7:0] <= rf_filter_coeff91_b;
11'h4b8:
    rdata[7:0] <= rf_filter_coeff92_a;
11'h4b9:
    rdata[7:0] <= rf_filter_coeff92_b;
11'h4ba:
    rdata[7:0] <= rf_filter_coeff93_a;
11'h4bb:
    rdata[7:0] <= rf_filter_coeff93_b;
11'h4bc:
    rdata[7:0] <= rf_filter_coeff94_a;
11'h4bd:
    rdata[7:0] <= rf_filter_coeff94_b;
11'h4be:
    rdata[7:0] <= rf_filter_coeff95_a;
11'h4bf:
    rdata[7:0] <= rf_filter_coeff95_b;
11'h4c0:
    rdata[7:0] <= rf_filter_coeff96_a;
11'h4c1:
    rdata[7:0] <= rf_filter_coeff96_b;
11'h4c2:
    rdata[7:0] <= rf_filter_coeff97_a;
11'h4c3:
    rdata[7:0] <= rf_filter_coeff97_b;
11'h4c4:
    rdata[7:0] <= rf_filter_coeff98_a;
11'h4c5:
    rdata[7:0] <= rf_filter_coeff98_b;
11'h4c6:
    rdata[7:0] <= rf_filter_coeff99_a;
11'h4c7:
    rdata[7:0] <= rf_filter_coeff99_b;
11'h4c8:
    rdata[7:0] <= rf_filter_coeff100_a;
11'h4c9:
    rdata[7:0] <= rf_filter_coeff100_b;
11'h4ca:
    rdata[7:0] <= rf_filter_coeff101_a;
11'h4cb:
    rdata[7:0] <= rf_filter_coeff101_b;
11'h4cc:
    rdata[7:0] <= rf_filter_coeff102_a;
11'h4cd:
    rdata[7:0] <= rf_filter_coeff102_b;
11'h4ce:
    rdata[7:0] <= rf_filter_coeff103_a;
11'h4cf:
    rdata[7:0] <= rf_filter_coeff103_b;
11'h4d0:
    rdata[7:0] <= rf_filter_coeff104_a;
11'h4d1:
    rdata[7:0] <= rf_filter_coeff104_b;
11'h4d2:
    rdata[7:0] <= rf_filter_coeff105_a;
```

```
11'h4d3:
    rdata[7:0] <= rf_filter_coeff105_b;
11'h4d4:
    rdata[7:0] <= rf_filter_coeff106_a;
11'h4d5:
    rdata[7:0] <= rf_filter_coeff106_b;
11'h4d6:
    rdata[7:0] <= rf_filter_coeff107_a;
11'h4d7:
    rdata[7:0] <= rf_filter_coeff107_b;
11'h4d8:
    rdata[7:0] <= rf_filter_coeff108_a;
11'h4d9:
    rdata[7:0] <= rf_filter_coeff108_b;
11'h4da:
    rdata[7:0] <= rf_filter_coeff109_a;
11'h4db:
    rdata[7:0] <= rf_filter_coeff109_b;
11'h4dc:
    rdata[7:0] <= rf_filter_coeff110_a;
11'h4dd:
    rdata[7:0] <= rf_filter_coeff110_b;
11'h4de:
    rdata[7:0] <= rf_filter_coeff111_a;
11'h4df:
    rdata[7:0] <= rf_filter_coeff111_b;
11'h4e0:
    rdata[7:0] <= rf_filter_coeff112_a;
11'h4e1:
    rdata[7:0] <= rf_filter_coeff112_b;
11'h4e2:
    rdata[7:0] <= rf_filter_coeff113_a;
11'h4e3:
    rdata[7:0] <= rf_filter_coeff113_b;
11'h4e4:
    rdata[7:0] <= rf_filter_coeff114_a;
11'h4e5:
    rdata[7:0] <= rf_filter_coeff114_b;
11'h4e6:
    rdata[7:0] <= rf_filter_coeff115_a;
11'h4e7:
    rdata[7:0] <= rf_filter_coeff115_b;
11'h4e8:
    rdata[7:0] <= rf_filter_coeff116_a;
11'h4e9:
    rdata[7:0] <= rf_filter_coeff116_b;
11'h4ea:
    rdata[7:0] <= rf_filter_coeff117_a;
11'h4eb:
    rdata[7:0] <= rf_filter_coeff117_b;
11'h4ec:
    rdata[7:0] <= rf_filter_coeff118_a;
11'h4ed:
    rdata[7:0] <= rf_filter_coeff118_b;
11'h4ee:
    rdata[7:0] <= rf_filter_coeff119_a;
11'h4ef:
    rdata[7:0] <= rf_filter_coeff119_b;
11'h4f0:
    rdata[7:0] <= rf_filter_coeff120_a;
11'h4f1:
    rdata[7:0] <= rf_filter_coeff120_b;
11'h4f2:
    rdata[7:0] <= rf_filter_coeff121_a;
11'h4f3:
    rdata[7:0] <= rf_filter_coeff121_b;
11'h4f4:
    rdata[7:0] <= rf_filter_coeff122_a;
11'h4f5:
    rdata[7:0] <= rf_filter_coeff122_b;
11'h4f6:
    rdata[7:0] <= rf_filter_coeff123_a;
11'h4f7:
    rdata[7:0] <= rf_filter_coeff123_b;
11'h4f8:
    rdata[7:0] <= rf_filter_coeff124_a;
11'h4f9:
    rdata[7:0] <= rf_filter_coeff124_b;
11'h4fa:
    rdata[7:0] <= rf_filter_coeff125_a;
11'h4fb:
    rdata[7:0] <= rf_filter_coeff125_b;
11'h4fc:
    rdata[7:0] <= rf_filter_coeff126_a;
11'h4fd:
    rdata[7:0] <= rf_filter_coeff126_b;
11'h4fe:
    rdata[7:0] <= rf_filter_coeff127_a;
11'h4ff:
    rdata[7:0] <= rf_filter_coeff127_b;
11'h500:
    rdata[7:0] <= rf_filter_coeff128_a;
11'h501:
    rdata[7:0] <= rf_filter_coeff128_b;
11'h502:
```

```
    rdata[7:0] <= rf_filter_coeff129_a;
11'h503:
    rdata[7:0] <= rf_filter_coeff129_b;
11'h504:
    rdata[7:0] <= rf_filter_coeff130_a;
11'h505:
    rdata[7:0] <= rf_filter_coeff130_b;
11'h506:
    rdata[7:0] <= rf_filter_coeff131_a;
11'h507:
    rdata[7:0] <= rf_filter_coeff131_b;
11'h508:
    rdata[7:0] <= rf_filter_coeff132_a;
11'h509:
    rdata[7:0] <= rf_filter_coeff132_b;
11'h50a:
    rdata[7:0] <= rf_filter_coeff133_a;
11'h50b:
    rdata[7:0] <= rf_filter_coeff133_b;
11'h50c:
    rdata[7:0] <= rf_filter_coeff134_a;
11'h50d:
    rdata[7:0] <= rf_filter_coeff134_b;
11'h50e:
    rdata[7:0] <= rf_filter_coeff135_a;
11'h50f:
    rdata[7:0] <= rf_filter_coeff135_b;
11'h510:
    rdata[7:0] <= rf_filter_coeff136_a;
11'h511:
    rdata[7:0] <= rf_filter_coeff136_b;
11'h512:
    rdata[7:0] <= rf_filter_coeff137_a;
11'h513:
    rdata[7:0] <= rf_filter_coeff137_b;
11'h514:
    rdata[7:0] <= rf_filter_coeff138_a;
11'h515:
    rdata[7:0] <= rf_filter_coeff138_b;
11'h516:
    rdata[7:0] <= rf_filter_coeff139_a;
11'h517:
    rdata[7:0] <= rf_filter_coeff139_b;
11'h518:
    rdata[7:0] <= rf_filter_coeff140_a;
11'h519:
    rdata[7:0] <= rf_filter_coeff140_b;
11'h51a:
    rdata[7:0] <= rf_filter_coeff141_a;
11'h51b:
    rdata[7:0] <= rf_filter_coeff141_b;
11'h51c:
    rdata[7:0] <= rf_filter_coeff142_a;
11'h51d:
    rdata[7:0] <= rf_filter_coeff142_b;
11'h51e:
    rdata[7:0] <= rf_filter_coeff143_a;
11'h51f:
    rdata[7:0] <= rf_filter_coeff143_b;
11'h520:
    rdata[7:0] <= rf_filter_coeff144_a;
11'h521:
    rdata[7:0] <= rf_filter_coeff144_b;
11'h522:
    rdata[7:0] <= rf_filter_coeff145_a;
11'h523:
    rdata[7:0] <= rf_filter_coeff145_b;
11'h524:
    rdata[7:0] <= rf_filter_coeff146_a;
11'h525:
    rdata[7:0] <= rf_filter_coeff146_b;
11'h526:
    rdata[7:0] <= rf_filter_coeff147_a;
11'h527:
    rdata[7:0] <= rf_filter_coeff147_b;
11'h528:
    rdata[7:0] <= rf_filter_coeff148_a;
11'h529:
    rdata[7:0] <= rf_filter_coeff148_b;
11'h52a:
    rdata[7:0] <= rf_filter_coeff149_a;
11'h52b:
    rdata[7:0] <= rf_filter_coeff149_b;
11'h52c:
    rdata[7:0] <= rf_filter_coeff150_a;
11'h52d:
    rdata[7:0] <= rf_filter_coeff150_b;
11'h52e:
    rdata[7:0] <= rf_filter_coeff151_a;
11'h52f:
    rdata[7:0] <= rf_filter_coeff151_b;
11'h530:
    rdata[7:0] <= rf_filter_coeff152_a;
11'h531:
    rdata[7:0] <= rf_filter_coeff152_b;
```

```
11'h532:
    rdata[7:0] <= rf_filter_coeff153_a;
11'h533:
    rdata[7:0] <= rf_filter_coeff153_b;
11'h534:
    rdata[7:0] <= rf_filter_coeff154_a;
11'h535:
    rdata[7:0] <= rf_filter_coeff154_b;
11'h536:
    rdata[7:0] <= rf_filter_coeff155_a;
11'h537:
    rdata[7:0] <= rf_filter_coeff155_b;
11'h538:
    rdata[7:0] <= rf_filter_coeff156_a;
11'h539:
    rdata[7:0] <= rf_filter_coeff156_b;
11'h53a:
    rdata[7:0] <= rf_filter_coeff157_a;
11'h53b:
    rdata[7:0] <= rf_filter_coeff157_b;
11'h53c:
    rdata[7:0] <= rf_filter_coeff158_a;
11'h53d:
    rdata[7:0] <= rf_filter_coeff158_b;
11'h53e:
    rdata[7:0] <= rf_filter_coeff159_a;
11'h53f:
    rdata[7:0] <= rf_filter_coeff159_b;
11'h540:
    rdata[7:0] <= rf_filter_coeff160_a;
11'h541:
    rdata[7:0] <= rf_filter_coeff160_b;
11'h542:
    rdata[7:0] <= rf_filter_coeff161_a;
11'h543:
    rdata[7:0] <= rf_filter_coeff161_b;
11'h544:
    rdata[7:0] <= rf_filter_coeff162_a;
11'h545:
    rdata[7:0] <= rf_filter_coeff162_b;
11'h546:
    rdata[7:0] <= rf_filter_coeff163_a;
11'h547:
    rdata[7:0] <= rf_filter_coeff163_b;
11'h548:
    rdata[7:0] <= rf_filter_coeff164_a;
11'h549:
    rdata[7:0] <= rf_filter_coeff164_b;
11'h54a:
    rdata[7:0] <= rf_filter_coeff165_a;
11'h54b:
    rdata[7:0] <= rf_filter_coeff165_b;
11'h54c:
    rdata[7:0] <= rf_filter_coeff166_a;
11'h54d:
    rdata[7:0] <= rf_filter_coeff166_b;
11'h54e:
    rdata[7:0] <= rf_filter_coeff167_a;
11'h54f:
    rdata[7:0] <= rf_filter_coeff167_b;
11'h550:
    rdata[7:0] <= rf_filter_coeff168_a;
11'h551:
    rdata[7:0] <= rf_filter_coeff168_b;
11'h552:
    rdata[7:0] <= rf_filter_coeff169_a;
11'h553:
    rdata[7:0] <= rf_filter_coeff169_b;
11'h554:
    rdata[7:0] <= rf_filter_coeff170_a;
11'h555:
    rdata[7:0] <= rf_filter_coeff170_b;
11'h556:
    rdata[7:0] <= rf_filter_coeff171_a;
11'h557:
    rdata[7:0] <= rf_filter_coeff171_b;
11'h558:
    rdata[7:0] <= rf_filter_coeff172_a;
11'h559:
    rdata[7:0] <= rf_filter_coeff172_b;
11'h55a:
    rdata[7:0] <= rf_filter_coeff173_a;
11'h55b:
    rdata[7:0] <= rf_filter_coeff173_b;
11'h55c:
    rdata[7:0] <= rf_filter_coeff174_a;
11'h55d:
    rdata[7:0] <= rf_filter_coeff174_b;
11'h55e:
    rdata[7:0] <= rf_filter_coeff175_a;
11'h55f:
    rdata[7:0] <= rf_filter_coeff175_b;
11'h560:
    rdata[7:0] <= rf_filter_coeff176_a;
11'h561:
```

```
    rdata[7:0] <= rf_filter_coeff176_b;
11'h562:
    rdata[7:0] <= rf_filter_coeff177_a;
11'h563:
    rdata[7:0] <= rf_filter_coeff177_b;
11'h564:
    rdata[7:0] <= rf_filter_coeff178_a;
11'h565:
    rdata[7:0] <= rf_filter_coeff178_b;
11'h566:
    rdata[7:0] <= rf_filter_coeff179_a;
11'h567:
    rdata[7:0] <= rf_filter_coeff179_b;
11'h568:
    rdata[7:0] <= rf_filter_coeff180_a;
11'h569:
    rdata[7:0] <= rf_filter_coeff180_b;
11'h56a:
    rdata[7:0] <= rf_filter_coeff181_a;
11'h56b:
    rdata[7:0] <= rf_filter_coeff181_b;
11'h56c:
    rdata[7:0] <= rf_filter_coeff182_a;
11'h56d:
    rdata[7:0] <= rf_filter_coeff182_b;
11'h56e:
    rdata[7:0] <= rf_filter_coeff183_a;
11'h56f:
    rdata[7:0] <= rf_filter_coeff183_b;
11'h570:
    rdata[7:0] <= rf_filter_coeff184_a;
11'h571:
    rdata[7:0] <= rf_filter_coeff184_b;
11'h572:
    rdata[7:0] <= rf_filter_coeff185_a;
11'h573:
    rdata[7:0] <= rf_filter_coeff185_b;
11'h574:
    rdata[7:0] <= rf_filter_coeff186_a;
11'h575:
    rdata[7:0] <= rf_filter_coeff186_b;
11'h576:
    rdata[7:0] <= rf_filter_coeff187_a;
11'h577:
    rdata[7:0] <= rf_filter_coeff187_b;
11'h578:
    rdata[7:0] <= rf_filter_coeff188_a;
11'h579:
    rdata[7:0] <= rf_filter_coeff188_b;
11'h57a:
    rdata[7:0] <= rf_filter_coeff189_a;
11'h57b:
    rdata[7:0] <= rf_filter_coeff189_b;
11'h57c:
    rdata[7:0] <= rf_filter_coeff190_a;
11'h57d:
    rdata[7:0] <= rf_filter_coeff190_b;
11'h57e:
    rdata[7:0] <= rf_filter_coeff191_a;
11'h57f:
    rdata[7:0] <= rf_filter_coeff191_b;
11'h580:
    rdata[7:0] <= rf_filter_coeff192_a;
11'h581:
    rdata[7:0] <= rf_filter_coeff192_b;
11'h582:
    rdata[7:0] <= rf_filter_coeff193_a;
11'h583:
    rdata[7:0] <= rf_filter_coeff193_b;
11'h584:
    rdata[7:0] <= rf_filter_coeff194_a;
11'h585:
    rdata[7:0] <= rf_filter_coeff194_b;
11'h586:
    rdata[7:0] <= rf_filter_coeff195_a;
11'h587:
    rdata[7:0] <= rf_filter_coeff195_b;
11'h588:
    rdata[7:0] <= rf_filter_coeff196_a;
11'h589:
    rdata[7:0] <= rf_filter_coeff196_b;
11'h58a:
    rdata[7:0] <= rf_filter_coeff197_a;
11'h58b:
    rdata[7:0] <= rf_filter_coeff197_b;
11'h58c:
    rdata[7:0] <= rf_filter_coeff198_a;
11'h58d:
    rdata[7:0] <= rf_filter_coeff198_b;
11'h58e:
    rdata[7:0] <= rf_filter_coeff199_a;
11'h58f:
    rdata[7:0] <= rf_filter_coeff199_b;
11'h590:
    rdata[7:0] <= rf_filter_coeff200_a;
```

```
11'h591:
    rdata[7:0] <= rf_filter_coeff200_b;
11'h592:
    rdata[7:0] <= rf_filter_coeff201_a;
11'h593:
    rdata[7:0] <= rf_filter_coeff201_b;
11'h594:
    rdata[7:0] <= rf_filter_coeff202_a;
11'h595:
    rdata[7:0] <= rf_filter_coeff202_b;
11'h596:
    rdata[7:0] <= rf_filter_coeff203_a;
11'h597:
    rdata[7:0] <= rf_filter_coeff203_b;
11'h598:
    rdata[7:0] <= rf_filter_coeff204_a;
11'h599:
    rdata[7:0] <= rf_filter_coeff204_b;
11'h59a:
    rdata[7:0] <= rf_filter_coeff205_a;
11'h59b:
    rdata[7:0] <= rf_filter_coeff205_b;
11'h59c:
    rdata[7:0] <= rf_filter_coeff206_a;
11'h59d:
    rdata[7:0] <= rf_filter_coeff206_b;
11'h59e:
    rdata[7:0] <= rf_filter_coeff207_a;
11'h59f:
    rdata[7:0] <= rf_filter_coeff207_b;
11'h5a0:
    rdata[7:0] <= rf_filter_coeff208_a;
11'h5a1:
    rdata[7:0] <= rf_filter_coeff208_b;
11'h5a2:
    rdata[7:0] <= rf_filter_coeff209_a;
11'h5a3:
    rdata[7:0] <= rf_filter_coeff209_b;
11'h5a4:
    rdata[7:0] <= rf_filter_coeff210_a;
11'h5a5:
    rdata[7:0] <= rf_filter_coeff210_b;
11'h5a6:
    rdata[7:0] <= rf_filter_coeff211_a;
11'h5a7:
    rdata[7:0] <= rf_filter_coeff211_b;
11'h5a8:
    rdata[7:0] <= rf_filter_coeff212_a;
11'h5a9:
    rdata[7:0] <= rf_filter_coeff212_b;
11'h5aa:
    rdata[7:0] <= rf_filter_coeff213_a;
11'h5ab:
    rdata[7:0] <= rf_filter_coeff213_b;
11'h5ac:
    rdata[7:0] <= rf_filter_coeff214_a;
11'h5ad:
    rdata[7:0] <= rf_filter_coeff214_b;
11'h5ae:
    rdata[7:0] <= rf_filter_coeff215_a;
11'h5af:
    rdata[7:0] <= rf_filter_coeff215_b;
11'h5b0:
    rdata[7:0] <= rf_filter_coeff216_a;
11'h5b1:
    rdata[7:0] <= rf_filter_coeff216_b;
11'h5b2:
    rdata[7:0] <= rf_filter_coeff217_a;
11'h5b3:
    rdata[7:0] <= rf_filter_coeff217_b;
11'h5b4:
    rdata[7:0] <= rf_filter_coeff218_a;
11'h5b5:
    rdata[7:0] <= rf_filter_coeff218_b;
11'h5b6:
    rdata[7:0] <= rf_filter_coeff219_a;
11'h5b7:
    rdata[7:0] <= rf_filter_coeff219_b;
11'h5b8:
    rdata[7:0] <= rf_filter_coeff220_a;
11'h5b9:
    rdata[7:0] <= rf_filter_coeff220_b;
11'h5ba:
    rdata[7:0] <= rf_filter_coeff221_a;
11'h5bb:
    rdata[7:0] <= rf_filter_coeff221_b;
11'h5bc:
    rdata[7:0] <= rf_filter_coeff222_a;
11'h5bd:
    rdata[7:0] <= rf_filter_coeff222_b;
11'h5be:
    rdata[7:0] <= rf_filter_coeff223_a;
11'h5bf:
    rdata[7:0] <= rf_filter_coeff223_b;
11'h5c0:
```

```
    rdata[7:0] <= rf_filter_coeff224_a;
11'h5c1:
    rdata[7:0] <= rf_filter_coeff224_b;
11'h5c2:
    rdata[7:0] <= rf_filter_coeff225_a;
11'h5c3:
    rdata[7:0] <= rf_filter_coeff225_b;
11'h5c4:
    rdata[7:0] <= rf_filter_coeff226_a;
11'h5c5:
    rdata[7:0] <= rf_filter_coeff226_b;
11'h5c6:
    rdata[7:0] <= rf_filter_coeff227_a;
11'h5c7:
    rdata[7:0] <= rf_filter_coeff227_b;
11'h5c8:
    rdata[7:0] <= rf_filter_coeff228_a;
11'h5c9:
    rdata[7:0] <= rf_filter_coeff228_b;
11'h5ca:
    rdata[7:0] <= rf_filter_coeff229_a;
11'h5cb:
    rdata[7:0] <= rf_filter_coeff229_b;
11'h5cc:
    rdata[7:0] <= rf_filter_coeff230_a;
11'h5cd:
    rdata[7:0] <= rf_filter_coeff230_b;
11'h5ce:
    rdata[7:0] <= rf_filter_coeff231_a;
11'h5cf:
    rdata[7:0] <= rf_filter_coeff231_b;
11'h5d0:
    rdata[7:0] <= rf_filter_coeff232_a;
11'h5d1:
    rdata[7:0] <= rf_filter_coeff232_b;
11'h5d2:
    rdata[7:0] <= rf_filter_coeff233_a;
11'h5d3:
    rdata[7:0] <= rf_filter_coeff233_b;
11'h5d4:
    rdata[7:0] <= rf_filter_coeff234_a;
11'h5d5:
    rdata[7:0] <= rf_filter_coeff234_b;
11'h5d6:
    rdata[7:0] <= rf_filter_coeff235_a;
11'h5d7:
    rdata[7:0] <= rf_filter_coeff235_b;
11'h5d8:
    rdata[7:0] <= rf_filter_coeff236_a;
11'h5d9:
    rdata[7:0] <= rf_filter_coeff236_b;
11'h5da:
    rdata[7:0] <= rf_filter_coeff237_a;
11'h5db:
    rdata[7:0] <= rf_filter_coeff237_b;
11'h5dc:
    rdata[7:0] <= rf_filter_coeff238_a;
11'h5dd:
    rdata[7:0] <= rf_filter_coeff238_b;
11'h5de:
    rdata[7:0] <= rf_filter_coeff239_a;
11'h5df:
    rdata[7:0] <= rf_filter_coeff239_b;
11'h5e0:
    rdata[7:0] <= rf_filter_coeff240_a;
11'h5e1:
    rdata[7:0] <= rf_filter_coeff240_b;
11'h5e2:
    rdata[7:0] <= rf_filter_coeff241_a;
11'h5e3:
    rdata[7:0] <= rf_filter_coeff241_b;
11'h5e4:
    rdata[7:0] <= rf_filter_coeff242_a;
11'h5e5:
    rdata[7:0] <= rf_filter_coeff242_b;
11'h5e6:
    rdata[7:0] <= rf_filter_coeff243_a;
11'h5e7:
    rdata[7:0] <= rf_filter_coeff243_b;
11'h5e8:
    rdata[7:0] <= rf_filter_coeff244_a;
11'h5e9:
    rdata[7:0] <= rf_filter_coeff244_b;
11'h5ea:
    rdata[7:0] <= rf_filter_coeff245_a;
11'h5eb:
    rdata[7:0] <= rf_filter_coeff245_b;
11'h5ec:
    rdata[7:0] <= rf_filter_coeff246_a;
11'h5ed:
    rdata[7:0] <= rf_filter_coeff246_b;
11'h5ee:
    rdata[7:0] <= rf_filter_coeff247_a;
11'h5ef:
    rdata[7:0] <= rf_filter_coeff247_b;
```

```
11'h5f0:
    rdata[7:0] <= rf_filter_coeff248_a;
11'h5f1:
    rdata[7:0] <= rf_filter_coeff248_b;
11'h5f2:
    rdata[7:0] <= rf_filter_coeff249_a;
11'h5f3:
    rdata[7:0] <= rf_filter_coeff249_b;
11'h5f4:
    rdata[7:0] <= rf_filter_coeff250_a;
11'h5f5:
    rdata[7:0] <= rf_filter_coeff250_b;
11'h5f6:
    rdata[7:0] <= rf_filter_coeff251_a;
11'h5f7:
    rdata[7:0] <= rf_filter_coeff251_b;
11'h5f8:
    rdata[7:0] <= rf_filter_coeff252_a;
11'h5f9:
    rdata[7:0] <= rf_filter_coeff252_b;
11'h5fa:
    rdata[7:0] <= rf_filter_coeff253_a;
11'h5fb:
    rdata[7:0] <= rf_filter_coeff253_b;
11'h5fc:
    rdata[7:0] <= rf_filter_coeff254_a;
11'h5fd:
    rdata[7:0] <= rf_filter_coeff254_b;
11'h5fe:
    rdata[7:0] <= rf_filter_coeff255_a;
11'h5ff:
    rdata[7:0] <= rf_filter_coeff255_b;
11'h600:
    rdata[7:0] <= rf_filter_coeff256_a;
11'h601:
    rdata[7:0] <= rf_filter_coeff256_b;
11'h602:
    rdata[7:0] <= rf_filter_coeff257_a;
11'h603:
    rdata[7:0] <= rf_filter_coeff257_b;
11'h604:
    rdata[7:0] <= rf_filter_coeff258_a;
11'h605:
    rdata[7:0] <= rf_filter_coeff258_b;
11'h606:
    rdata[7:0] <= rf_filter_coeff259_a;
11'h607:
    rdata[7:0] <= rf_filter_coeff259_b;
11'h608:
    rdata[7:0] <= rf_filter_coeff260_a;
11'h609:
    rdata[7:0] <= rf_filter_coeff260_b;
11'h60a:
    rdata[7:0] <= rf_filter_coeff261_a;
11'h60b:
    rdata[7:0] <= rf_filter_coeff261_b;
11'h60c:
    rdata[7:0] <= rf_filter_coeff262_a;
11'h60d:
    rdata[7:0] <= rf_filter_coeff262_b;
11'h60e:
    rdata[7:0] <= rf_filter_coeff263_a;
11'h60f:
    rdata[7:0] <= rf_filter_coeff263_b;
11'h610:
    rdata[7:0] <= rf_filter_coeff264_a;
11'h611:
    rdata[7:0] <= rf_filter_coeff264_b;
11'h612:
    rdata[7:0] <= rf_filter_coeff265_a;
11'h613:
    rdata[7:0] <= rf_filter_coeff265_b;
11'h614:
    rdata[7:0] <= rf_filter_coeff266_a;
11'h615:
    rdata[7:0] <= rf_filter_coeff266_b;
11'h616:
    rdata[7:0] <= rf_filter_coeff267_a;
11'h617:
    rdata[7:0] <= rf_filter_coeff267_b;
11'h618:
    rdata[7:0] <= rf_filter_coeff268_a;
11'h619:
    rdata[7:0] <= rf_filter_coeff268_b;
11'h61a:
    rdata[7:0] <= rf_filter_coeff269_a;
11'h61b:
    rdata[7:0] <= rf_filter_coeff269_b;
11'h61c:
    rdata[7:0] <= rf_filter_coeff270_a;
11'h61d:
    rdata[7:0] <= rf_filter_coeff270_b;
11'h61e:
    rdata[7:0] <= rf_filter_coeff271_a;
11'h61f:
```

```
    rdata[7:0] <= rf_filter_coeff271_b;
11'h620:
    rdata[7:0] <= rf_filter_coeff272_a;
11'h621:
    rdata[7:0] <= rf_filter_coeff272_b;
11'h622:
    rdata[7:0] <= rf_filter_coeff273_a;
11'h623:
    rdata[7:0] <= rf_filter_coeff273_b;
11'h624:
    rdata[7:0] <= rf_filter_coeff274_a;
11'h625:
    rdata[7:0] <= rf_filter_coeff274_b;
11'h626:
    rdata[7:0] <= rf_filter_coeff275_a;
11'h627:
    rdata[7:0] <= rf_filter_coeff275_b;
11'h628:
    rdata[7:0] <= rf_filter_coeff276_a;
11'h629:
    rdata[7:0] <= rf_filter_coeff276_b;
11'h62a:
    rdata[7:0] <= rf_filter_coeff277_a;
11'h62b:
    rdata[7:0] <= rf_filter_coeff277_b;
11'h62c:
    rdata[7:0] <= rf_filter_coeff278_a;
11'h62d:
    rdata[7:0] <= rf_filter_coeff278_b;
11'h62e:
    rdata[7:0] <= rf_filter_coeff279_a;
11'h62f:
    rdata[7:0] <= rf_filter_coeff279_b;
11'h630:
    rdata[7:0] <= rf_filter_coeff280_a;
11'h631:
    rdata[7:0] <= rf_filter_coeff280_b;
11'h632:
    rdata[7:0] <= rf_filter_coeff281_a;
11'h633:
    rdata[7:0] <= rf_filter_coeff281_b;
11'h634:
    rdata[7:0] <= rf_filter_coeff282_a;
11'h635:
    rdata[7:0] <= rf_filter_coeff282_b;
11'h636:
    rdata[7:0] <= rf_filter_coeff283_a;
11'h637:
    rdata[7:0] <= rf_filter_coeff283_b;
11'h638:
    rdata[7:0] <= rf_filter_coeff284_a;
11'h639:
    rdata[7:0] <= rf_filter_coeff284_b;
11'h63a:
    rdata[7:0] <= rf_filter_coeff285_a;
11'h63b:
    rdata[7:0] <= rf_filter_coeff285_b;
11'h63c:
    rdata[7:0] <= rf_filter_coeff286_a;
11'h63d:
    rdata[7:0] <= rf_filter_coeff286_b;
11'h63e:
    rdata[7:0] <= rf_filter_coeff287_a;
11'h63f:
    rdata[7:0] <= rf_filter_coeff287_b;
11'h640:
    rdata[7:0] <= rf_filter_coeff288_a;
11'h641:
    rdata[7:0] <= rf_filter_coeff288_b;
11'h642:
    rdata[7:0] <= rf_filter_coeff289_a;
11'h643:
    rdata[7:0] <= rf_filter_coeff289_b;
11'h644:
    rdata[7:0] <= rf_filter_coeff290_a;
11'h645:
    rdata[7:0] <= rf_filter_coeff290_b;
11'h646:
    rdata[7:0] <= rf_filter_coeff291_a;
11'h647:
    rdata[7:0] <= rf_filter_coeff291_b;
11'h648:
    rdata[7:0] <= rf_filter_coeff292_a;
11'h649:
    rdata[7:0] <= rf_filter_coeff292_b;
11'h64a:
    rdata[7:0] <= rf_filter_coeff293_a;
11'h64b:
    rdata[7:0] <= rf_filter_coeff293_b;
11'h64c:
    rdata[7:0] <= rf_filter_coeff294_a;
11'h64d:
    rdata[7:0] <= rf_filter_coeff294_b;
11'h64e:
    rdata[7:0] <= rf_filter_coeff295_a;
```

```
11'h64f:
    rdata[7:0] <= rf_filter_coeff295_b;
11'h650:
    rdata[7:0] <= rf_filter_coeff296_a;
11'h651:
    rdata[7:0] <= rf_filter_coeff296_b;
11'h652:
    rdata[7:0] <= rf_filter_coeff297_a;
11'h653:
    rdata[7:0] <= rf_filter_coeff297_b;
11'h654:
    rdata[7:0] <= rf_filter_coeff298_a;
11'h655:
    rdata[7:0] <= rf_filter_coeff298_b;
11'h656:
    rdata[7:0] <= rf_filter_coeff299_a;
11'h657:
    rdata[7:0] <= rf_filter_coeff299_b;
11'h658:
    rdata[7:0] <= rf_filter_coeff300_a;
11'h659:
    rdata[7:0] <= rf_filter_coeff300_b;
11'h65a:
    rdata[7:0] <= rf_filter_coeff301_a;
11'h65b:
    rdata[7:0] <= rf_filter_coeff301_b;
11'h65c:
    rdata[7:0] <= rf_filter_coeff302_a;
11'h65d:
    rdata[7:0] <= rf_filter_coeff302_b;
11'h65e:
    rdata[7:0] <= rf_filter_coeff303_a;
11'h65f:
    rdata[7:0] <= rf_filter_coeff303_b;
11'h660:
    rdata[7:0] <= rf_filter_coeff304_a;
11'h661:
    rdata[7:0] <= rf_filter_coeff304_b;
11'h662:
    rdata[7:0] <= rf_filter_coeff305_a;
11'h663:
    rdata[7:0] <= rf_filter_coeff305_b;
11'h664:
    rdata[7:0] <= rf_filter_coeff306_a;
11'h665:
    rdata[7:0] <= rf_filter_coeff306_b;
11'h666:
    rdata[7:0] <= rf_filter_coeff307_a;
11'h667:
    rdata[7:0] <= rf_filter_coeff307_b;
11'h668:
    rdata[7:0] <= rf_filter_coeff308_a;
11'h669:
    rdata[7:0] <= rf_filter_coeff308_b;
11'h66a:
    rdata[7:0] <= rf_filter_coeff309_a;
11'h66b:
    rdata[7:0] <= rf_filter_coeff309_b;
11'h66c:
    rdata[7:0] <= rf_filter_coeff310_a;
11'h66d:
    rdata[7:0] <= rf_filter_coeff310_b;
11'h66e:
    rdata[7:0] <= rf_filter_coeff311_a;
11'h66f:
    rdata[7:0] <= rf_filter_coeff311_b;
11'h670:
    rdata[7:0] <= rf_filter_coeff312_a;
11'h671:
    rdata[7:0] <= rf_filter_coeff312_b;
11'h672:
    rdata[7:0] <= rf_filter_coeff313_a;
11'h673:
    rdata[7:0] <= rf_filter_coeff313_b;
11'h674:
    rdata[7:0] <= rf_filter_coeff314_a;
11'h675:
    rdata[7:0] <= rf_filter_coeff314_b;
11'h676:
    rdata[7:0] <= rf_filter_coeff315_a;
11'h677:
    rdata[7:0] <= rf_filter_coeff315_b;
11'h678:
    rdata[7:0] <= rf_filter_coeff316_a;
11'h679:
    rdata[7:0] <= rf_filter_coeff316_b;
11'h67a:
    rdata[7:0] <= rf_filter_coeff317_a;
11'h67b:
    rdata[7:0] <= rf_filter_coeff317_b;
11'h67c:
    rdata[7:0] <= rf_filter_coeff318_a;
11'h67d:
    rdata[7:0] <= rf_filter_coeff318_b;
11'h67e:
```

```
    rdata[7:0] <= rf_filter_coeff319_a;
11'h67f:
    rdata[7:0] <= rf_filter_coeff319_b;
11'h680:
    rdata[7:0] <= rf_filter_coeff320_a;
11'h681:
    rdata[7:0] <= rf_filter_coeff320_b;
11'h682:
    rdata[7:0] <= rf_filter_coeff321_a;
11'h683:
    rdata[7:0] <= rf_filter_coeff321_b;
11'h684:
    rdata[7:0] <= rf_filter_coeff322_a;
11'h685:
    rdata[7:0] <= rf_filter_coeff322_b;
11'h686:
    rdata[7:0] <= rf_filter_coeff323_a;
11'h687:
    rdata[7:0] <= rf_filter_coeff323_b;
11'h688:
    rdata[7:0] <= rf_filter_coeff324_a;
11'h689:
    rdata[7:0] <= rf_filter_coeff324_b;
11'h68a:
    rdata[7:0] <= rf_filter_coeff325_a;
11'h68b:
    rdata[7:0] <= rf_filter_coeff325_b;
11'h68c:
    rdata[7:0] <= rf_filter_coeff326_a;
11'h68d:
    rdata[7:0] <= rf_filter_coeff326_b;
11'h68e:
    rdata[7:0] <= rf_filter_coeff327_a;
11'h68f:
    rdata[7:0] <= rf_filter_coeff327_b;
11'h690:
    rdata[7:0] <= rf_filter_coeff328_a;
11'h691:
    rdata[7:0] <= rf_filter_coeff328_b;
11'h692:
    rdata[7:0] <= rf_filter_coeff329_a;
11'h693:
    rdata[7:0] <= rf_filter_coeff329_b;
11'h694:
    rdata[7:0] <= rf_filter_coeff330_a;
11'h695:
    rdata[7:0] <= rf_filter_coeff330_b;
11'h696:
    rdata[7:0] <= rf_filter_coeff331_a;
11'h697:
    rdata[7:0] <= rf_filter_coeff331_b;
11'h698:
    rdata[7:0] <= rf_filter_coeff332_a;
11'h699:
    rdata[7:0] <= rf_filter_coeff332_b;
11'h69a:
    rdata[7:0] <= rf_filter_coeff333_a;
11'h69b:
    rdata[7:0] <= rf_filter_coeff333_b;
11'h69c:
    rdata[7:0] <= rf_filter_coeff334_a;
11'h69d:
    rdata[7:0] <= rf_filter_coeff334_b;
11'h69e:
    rdata[7:0] <= rf_filter_coeff335_a;
11'h69f:
    rdata[7:0] <= rf_filter_coeff335_b;
11'h6a0:
    rdata[7:0] <= rf_filter_coeff336_a;
11'h6a1:
    rdata[7:0] <= rf_filter_coeff336_b;
11'h6a2:
    rdata[7:0] <= rf_filter_coeff337_a;
11'h6a3:
    rdata[7:0] <= rf_filter_coeff337_b;
11'h6a4:
    rdata[7:0] <= rf_filter_coeff338_a;
11'h6a5:
    rdata[7:0] <= rf_filter_coeff338_b;
11'h6a6:
    rdata[7:0] <= rf_filter_coeff339_a;
11'h6a7:
    rdata[7:0] <= rf_filter_coeff339_b;
11'h6a8:
    rdata[7:0] <= rf_filter_coeff340_a;
11'h6a9:
    rdata[7:0] <= rf_filter_coeff340_b;
11'h6aa:
    rdata[7:0] <= rf_filter_coeff341_a;
11'h6ab:
    rdata[7:0] <= rf_filter_coeff341_b;
11'h6ac:
    rdata[7:0] <= rf_filter_coeff342_a;
11'h6ad:
    rdata[7:0] <= rf_filter_coeff342_b;
```

```
11'h6ae:
    rdata[7:0] <= rf_filter_coeff343_a;
11'h6af:
    rdata[7:0] <= rf_filter_coeff343_b;
11'h6b0:
    rdata[7:0] <= rf_filter_coeff344_a;
11'h6b1:
    rdata[7:0] <= rf_filter_coeff344_b;
11'h6b2:
    rdata[7:0] <= rf_filter_coeff345_a;
11'h6b3:
    rdata[7:0] <= rf_filter_coeff345_b;
11'h6b4:
    rdata[7:0] <= rf_filter_coeff346_a;
11'h6b5:
    rdata[7:0] <= rf_filter_coeff346_b;
11'h6b6:
    rdata[7:0] <= rf_filter_coeff347_a;
11'h6b7:
    rdata[7:0] <= rf_filter_coeff347_b;
11'h6b8:
    rdata[7:0] <= rf_filter_coeff348_a;
11'h6b9:
    rdata[7:0] <= rf_filter_coeff348_b;
11'h6ba:
    rdata[7:0] <= rf_filter_coeff349_a;
11'h6bb:
    rdata[7:0] <= rf_filter_coeff349_b;
11'h6bc:
    rdata[7:0] <= rf_filter_coeff350_a;
11'h6bd:
    rdata[7:0] <= rf_filter_coeff350_b;
11'h6be:
    rdata[7:0] <= rf_filter_coeff351_a;
11'h6bf:
    rdata[7:0] <= rf_filter_coeff351_b;
11'h6c0:
    rdata[7:0] <= rf_filter_coeff352_a;
11'h6c1:
    rdata[7:0] <= rf_filter_coeff352_b;
11'h6c2:
    rdata[7:0] <= rf_filter_coeff353_a;
11'h6c3:
    rdata[7:0] <= rf_filter_coeff353_b;
11'h6c4:
    rdata[7:0] <= rf_filter_coeff354_a;
11'h6c5:
    rdata[7:0] <= rf_filter_coeff354_b;
11'h6c6:
    rdata[7:0] <= rf_filter_coeff355_a;
11'h6c7:
    rdata[7:0] <= rf_filter_coeff355_b;
11'h6c8:
    rdata[7:0] <= rf_filter_coeff356_a;
11'h6c9:
    rdata[7:0] <= rf_filter_coeff356_b;
11'h6ca:
    rdata[7:0] <= rf_filter_coeff357_a;
11'h6cb:
    rdata[7:0] <= rf_filter_coeff357_b;
11'h6cc:
    rdata[7:0] <= rf_filter_coeff358_a;
11'h6cd:
    rdata[7:0] <= rf_filter_coeff358_b;
11'h6ce:
    rdata[7:0] <= rf_filter_coeff359_a;
11'h6cf:
    rdata[7:0] <= rf_filter_coeff359_b;
11'h6d0:
    rdata[7:0] <= rf_filter_coeff360_a;
11'h6d1:
    rdata[7:0] <= rf_filter_coeff360_b;
11'h6d2:
    rdata[7:0] <= rf_filter_coeff361_a;
11'h6d3:
    rdata[7:0] <= rf_filter_coeff361_b;
11'h6d4:
    rdata[7:0] <= rf_filter_coeff362_a;
11'h6d5:
    rdata[7:0] <= rf_filter_coeff362_b;
11'h6d6:
    rdata[7:0] <= rf_filter_coeff363_a;
11'h6d7:
    rdata[7:0] <= rf_filter_coeff363_b;
11'h6d8:
    rdata[7:0] <= rf_filter_coeff364_a;
11'h6d9:
    rdata[7:0] <= rf_filter_coeff364_b;
11'h6da:
    rdata[7:0] <= rf_filter_coeff365_a;
11'h6db:
    rdata[7:0] <= rf_filter_coeff365_b;
11'h6dc:
    rdata[7:0] <= rf_filter_coeff366_a;
11'h6dd:
```

```
    rdata[7:0] <= rf_filter_coeff366_b;
11'h6de:
    rdata[7:0] <= rf_filter_coeff367_a;
11'h6df:
    rdata[7:0] <= rf_filter_coeff367_b;
11'h6e0:
    rdata[7:0] <= rf_filter_coeff368_a;
11'h6e1:
    rdata[7:0] <= rf_filter_coeff368_b;
11'h6e2:
    rdata[7:0] <= rf_filter_coeff369_a;
11'h6e3:
    rdata[7:0] <= rf_filter_coeff369_b;
11'h6e4:
    rdata[7:0] <= rf_filter_coeff370_a;
11'h6e5:
    rdata[7:0] <= rf_filter_coeff370_b;
11'h6e6:
    rdata[7:0] <= rf_filter_coeff371_a;
11'h6e7:
    rdata[7:0] <= rf_filter_coeff371_b;
11'h6e8:
    rdata[7:0] <= rf_filter_coeff372_a;
11'h6e9:
    rdata[7:0] <= rf_filter_coeff372_b;
11'h6ea:
    rdata[7:0] <= rf_filter_coeff373_a;
11'h6eb:
    rdata[7:0] <= rf_filter_coeff373_b;
11'h6ec:
    rdata[7:0] <= rf_filter_coeff374_a;
11'h6ed:
    rdata[7:0] <= rf_filter_coeff374_b;
11'h6ee:
    rdata[7:0] <= rf_filter_coeff375_a;
11'h6ef:
    rdata[7:0] <= rf_filter_coeff375_b;
11'h6f0:
    rdata[7:0] <= rf_filter_coeff376_a;
11'h6f1:
    rdata[7:0] <= rf_filter_coeff376_b;
11'h6f2:
    rdata[7:0] <= rf_filter_coeff377_a;
11'h6f3:
    rdata[7:0] <= rf_filter_coeff377_b;
11'h6f4:
    rdata[7:0] <= rf_filter_coeff378_a;
11'h6f5:
    rdata[7:0] <= rf_filter_coeff378_b;
11'h6f6:
    rdata[7:0] <= rf_filter_coeff379_a;
11'h6f7:
    rdata[7:0] <= rf_filter_coeff379_b;
11'h6f8:
    rdata[7:0] <= rf_filter_coeff380_a;
11'h6f9:
    rdata[7:0] <= rf_filter_coeff380_b;
11'h6fa:
    rdata[7:0] <= rf_filter_coeff381_a;
11'h6fb:
    rdata[7:0] <= rf_filter_coeff381_b;
11'h6fc:
    rdata[7:0] <= rf_filter_coeff382_a;
11'h6fd:
    rdata[7:0] <= rf_filter_coeff382_b;
11'h6fe:
    rdata[7:0] <= rf_filter_coeff383_a;
11'h6ff:
    rdata[7:0] <= rf_filter_coeff383_b;
11'h700:
    rdata[7:0] <= rf_filter_coeff384_a;
11'h701:
    rdata[7:0] <= rf_filter_coeff384_b;
11'h702:
    rdata[7:0] <= rf_filter_coeff385_a;
11'h703:
    rdata[7:0] <= rf_filter_coeff385_b;
11'h704:
    rdata[7:0] <= rf_filter_coeff386_a;
11'h705:
    rdata[7:0] <= rf_filter_coeff386_b;
11'h706:
    rdata[7:0] <= rf_filter_coeff387_a;
11'h707:
    rdata[7:0] <= rf_filter_coeff387_b;
11'h708:
    rdata[7:0] <= rf_filter_coeff388_a;
11'h709:
    rdata[7:0] <= rf_filter_coeff388_b;
11'h70a:
    rdata[7:0] <= rf_filter_coeff389_a;
11'h70b:
    rdata[7:0] <= rf_filter_coeff389_b;
11'h70c:
    rdata[7:0] <= rf_filter_coeff390_a;
```

```
11'h70d:
    rdata[7:0] <= rf_filter_coeff390_b;
11'h70e:
    rdata[7:0] <= rf_filter_coeff391_a;
11'h70f:
    rdata[7:0] <= rf_filter_coeff391_b;
11'h710:
    rdata[7:0] <= rf_filter_coeff392_a;
11'h711:
    rdata[7:0] <= rf_filter_coeff392_b;
11'h712:
    rdata[7:0] <= rf_filter_coeff393_a;
11'h713:
    rdata[7:0] <= rf_filter_coeff393_b;
11'h714:
    rdata[7:0] <= rf_filter_coeff394_a;
11'h715:
    rdata[7:0] <= rf_filter_coeff394_b;
11'h716:
    rdata[7:0] <= rf_filter_coeff395_a;
11'h717:
    rdata[7:0] <= rf_filter_coeff395_b;
11'h718:
    rdata[7:0] <= rf_filter_coeff396_a;
11'h719:
    rdata[7:0] <= rf_filter_coeff396_b;
11'h71a:
    rdata[7:0] <= rf_filter_coeff397_a;
11'h71b:
    rdata[7:0] <= rf_filter_coeff397_b;
11'h71c:
    rdata[7:0] <= rf_filter_coeff398_a;
11'h71d:
    rdata[7:0] <= rf_filter_coeff398_b;
11'h71e:
    rdata[7:0] <= rf_filter_coeff399_a;
11'h71f:
    rdata[7:0] <= rf_filter_coeff399_b;
11'h720:
    rdata[7:0] <= rf_filter_coeff400_a;
11'h721:
    rdata[7:0] <= rf_filter_coeff400_b;
11'h722:
    rdata[7:0] <= rf_filter_coeff401_a;
11'h723:
    rdata[7:0] <= rf_filter_coeff401_b;
11'h724:
    rdata[7:0] <= rf_filter_coeff402_a;
11'h725:
    rdata[7:0] <= rf_filter_coeff402_b;
11'h726:
    rdata[7:0] <= rf_filter_coeff403_a;
11'h727:
    rdata[7:0] <= rf_filter_coeff403_b;
11'h728:
    rdata[7:0] <= rf_filter_coeff404_a;
11'h729:
    rdata[7:0] <= rf_filter_coeff404_b;
11'h72a:
    rdata[7:0] <= rf_filter_coeff405_a;
11'h72b:
    rdata[7:0] <= rf_filter_coeff405_b;
11'h72c:
    rdata[7:0] <= rf_filter_coeff406_a;
11'h72d:
    rdata[7:0] <= rf_filter_coeff406_b;
11'h72e:
    rdata[7:0] <= rf_filter_coeff407_a;
11'h72f:
    rdata[7:0] <= rf_filter_coeff407_b;
11'h730:
    rdata[7:0] <= rf_filter_coeff408_a;
11'h731:
    rdata[7:0] <= rf_filter_coeff408_b;
11'h732:
    rdata[7:0] <= rf_filter_coeff409_a;
11'h733:
    rdata[7:0] <= rf_filter_coeff409_b;
11'h734:
    rdata[7:0] <= rf_filter_coeff410_a;
11'h735:
    rdata[7:0] <= rf_filter_coeff410_b;
11'h736:
    rdata[7:0] <= rf_filter_coeff411_a;
11'h737:
    rdata[7:0] <= rf_filter_coeff411_b;
11'h738:
    rdata[7:0] <= rf_filter_coeff412_a;
11'h739:
    rdata[7:0] <= rf_filter_coeff412_b;
11'h73a:
    rdata[7:0] <= rf_filter_coeff413_a;
11'h73b:
    rdata[7:0] <= rf_filter_coeff413_b;
11'h73c:
```

```
    rdata[7:0] <= rf_filter_coeff414_a;
11'h73d:
    rdata[7:0] <= rf_filter_coeff414_b;
11'h73e:
    rdata[7:0] <= rf_filter_coeff415_a;
11'h73f:
    rdata[7:0] <= rf_filter_coeff415_b;
11'h740:
    rdata[7:0] <= rf_filter_coeff416_a;
11'h741:
    rdata[7:0] <= rf_filter_coeff416_b;
11'h742:
    rdata[7:0] <= rf_filter_coeff417_a;
11'h743:
    rdata[7:0] <= rf_filter_coeff417_b;
11'h744:
    rdata[7:0] <= rf_filter_coeff418_a;
11'h745:
    rdata[7:0] <= rf_filter_coeff418_b;
11'h746:
    rdata[7:0] <= rf_filter_coeff419_a;
11'h747:
    rdata[7:0] <= rf_filter_coeff419_b;
11'h748:
    rdata[7:0] <= rf_filter_coeff420_a;
11'h749:
    rdata[7:0] <= rf_filter_coeff420_b;
11'h74a:
    rdata[7:0] <= rf_filter_coeff421_a;
11'h74b:
    rdata[7:0] <= rf_filter_coeff421_b;
11'h74c:
    rdata[7:0] <= rf_filter_coeff422_a;
11'h74d:
    rdata[7:0] <= rf_filter_coeff422_b;
11'h74e:
    rdata[7:0] <= rf_filter_coeff423_a;
11'h74f:
    rdata[7:0] <= rf_filter_coeff423_b;
11'h750:
    rdata[7:0] <= rf_filter_coeff424_a;
11'h751:
    rdata[7:0] <= rf_filter_coeff424_b;
11'h752:
    rdata[7:0] <= rf_filter_coeff425_a;
11'h753:
    rdata[7:0] <= rf_filter_coeff425_b;
11'h754:
    rdata[7:0] <= rf_filter_coeff426_a;
11'h755:
    rdata[7:0] <= rf_filter_coeff426_b;
11'h756:
    rdata[7:0] <= rf_filter_coeff427_a;
11'h757:
    rdata[7:0] <= rf_filter_coeff427_b;
11'h758:
    rdata[7:0] <= rf_filter_coeff428_a;
11'h759:
    rdata[7:0] <= rf_filter_coeff428_b;
11'h75a:
    rdata[7:0] <= rf_filter_coeff429_a;
11'h75b:
    rdata[7:0] <= rf_filter_coeff429_b;
11'h75c:
    rdata[7:0] <= rf_filter_coeff430_a;
11'h75d:
    rdata[7:0] <= rf_filter_coeff430_b;
11'h75e:
    rdata[7:0] <= rf_filter_coeff431_a;
11'h75f:
    rdata[7:0] <= rf_filter_coeff431_b;
11'h760:
    rdata[7:0] <= rf_filter_coeff432_a;
11'h761:
    rdata[7:0] <= rf_filter_coeff432_b;
11'h762:
    rdata[7:0] <= rf_filter_coeff433_a;
11'h763:
    rdata[7:0] <= rf_filter_coeff433_b;
11'h764:
    rdata[7:0] <= rf_filter_coeff434_a;
11'h765:
    rdata[7:0] <= rf_filter_coeff434_b;
11'h766:
    rdata[7:0] <= rf_filter_coeff435_a;
11'h767:
    rdata[7:0] <= rf_filter_coeff435_b;
11'h768:
    rdata[7:0] <= rf_filter_coeff436_a;
11'h769:
    rdata[7:0] <= rf_filter_coeff436_b;
11'h76a:
    rdata[7:0] <= rf_filter_coeff437_a;
11'h76b:
    rdata[7:0] <= rf_filter_coeff437_b;
```

```
11'h76c:
    rdata[7:0] <= rf_filter_coeff438_a;
11'h76d:
    rdata[7:0] <= rf_filter_coeff438_b;
11'h76e:
    rdata[7:0] <= rf_filter_coeff439_a;
11'h76f:
    rdata[7:0] <= rf_filter_coeff439_b;
11'h770:
    rdata[7:0] <= rf_filter_coeff440_a;
11'h771:
    rdata[7:0] <= rf_filter_coeff440_b;
11'h772:
    rdata[7:0] <= rf_filter_coeff441_a;
11'h773:
    rdata[7:0] <= rf_filter_coeff441_b;
11'h774:
    rdata[7:0] <= rf_filter_coeff442_a;
11'h775:
    rdata[7:0] <= rf_filter_coeff442_b;
11'h776:
    rdata[7:0] <= rf_filter_coeff443_a;
11'h777:
    rdata[7:0] <= rf_filter_coeff443_b;
11'h778:
    rdata[7:0] <= rf_filter_coeff444_a;
11'h779:
    rdata[7:0] <= rf_filter_coeff444_b;
11'h77a:
    rdata[7:0] <= rf_filter_coeff445_a;
11'h77b:
    rdata[7:0] <= rf_filter_coeff445_b;
11'h77c:
    rdata[7:0] <= rf_filter_coeff446_a;
11'h77d:
    rdata[7:0] <= rf_filter_coeff446_b;
11'h77e:
    rdata[7:0] <= rf_filter_coeff447_a;
11'h77f:
    rdata[7:0] <= rf_filter_coeff447_b;
11'h780:
    rdata[7:0] <= rf_filter_coeff448_a;
11'h781:
    rdata[7:0] <= rf_filter_coeff448_b;
11'h782:
    rdata[7:0] <= rf_filter_coeff449_a;
11'h783:
    rdata[7:0] <= rf_filter_coeff449_b;
11'h784:
    rdata[7:0] <= rf_filter_coeff450_a;
11'h785:
    rdata[7:0] <= rf_filter_coeff450_b;
11'h786:
    rdata[7:0] <= rf_filter_coeff451_a;
11'h787:
    rdata[7:0] <= rf_filter_coeff451_b;
11'h788:
    rdata[7:0] <= rf_filter_coeff452_a;
11'h789:
    rdata[7:0] <= rf_filter_coeff452_b;
11'h78a:
    rdata[7:0] <= rf_filter_coeff453_a;
11'h78b:
    rdata[7:0] <= rf_filter_coeff453_b;
11'h78c:
    rdata[7:0] <= rf_filter_coeff454_a;
11'h78d:
    rdata[7:0] <= rf_filter_coeff454_b;
11'h78e:
    rdata[7:0] <= rf_filter_coeff455_a;
11'h78f:
    rdata[7:0] <= rf_filter_coeff455_b;
11'h790:
    rdata[7:0] <= rf_filter_coeff456_a;
11'h791:
    rdata[7:0] <= rf_filter_coeff456_b;
11'h792:
    rdata[7:0] <= rf_filter_coeff457_a;
11'h793:
    rdata[7:0] <= rf_filter_coeff457_b;
11'h794:
    rdata[7:0] <= rf_filter_coeff458_a;
11'h795:
    rdata[7:0] <= rf_filter_coeff458_b;
11'h796:
    rdata[7:0] <= rf_filter_coeff459_a;
11'h797:
    rdata[7:0] <= rf_filter_coeff459_b;
11'h798:
    rdata[7:0] <= rf_filter_coeff460_a;
11'h799:
    rdata[7:0] <= rf_filter_coeff460_b;
11'h79a:
    rdata[7:0] <= rf_filter_coeff461_a;
11'h79b:
```

```
    rdata[7:0] <= rf_filter_coeff461_b;
11'h79c:
    rdata[7:0] <= rf_filter_coeff462_a;
11'h79d:
    rdata[7:0] <= rf_filter_coeff462_b;
11'h79e:
    rdata[7:0] <= rf_filter_coeff463_a;
11'h79f:
    rdata[7:0] <= rf_filter_coeff463_b;
11'h7a0:
    rdata[7:0] <= rf_filter_coeff464_a;
11'h7a1:
    rdata[7:0] <= rf_filter_coeff464_b;
11'h7a2:
    rdata[7:0] <= rf_filter_coeff465_a;
11'h7a3:
    rdata[7:0] <= rf_filter_coeff465_b;
11'h7a4:
    rdata[7:0] <= rf_filter_coeff466_a;
11'h7a5:
    rdata[7:0] <= rf_filter_coeff466_b;
11'h7a6:
    rdata[7:0] <= rf_filter_coeff467_a;
11'h7a7:
    rdata[7:0] <= rf_filter_coeff467_b;
11'h7a8:
    rdata[7:0] <= rf_filter_coeff468_a;
11'h7a9:
    rdata[7:0] <= rf_filter_coeff468_b;
11'h7aa:
    rdata[7:0] <= rf_filter_coeff469_a;
11'h7ab:
    rdata[7:0] <= rf_filter_coeff469_b;
11'h7ac:
    rdata[7:0] <= rf_filter_coeff470_a;
11'h7ad:
    rdata[7:0] <= rf_filter_coeff470_b;
11'h7ae:
    rdata[7:0] <= rf_filter_coeff471_a;
11'h7af:
    rdata[7:0] <= rf_filter_coeff471_b;
11'h7b0:
    rdata[7:0] <= rf_filter_coeff472_a;
11'h7b1:
    rdata[7:0] <= rf_filter_coeff472_b;
11'h7b2:
    rdata[7:0] <= rf_filter_coeff473_a;
11'h7b3:
    rdata[7:0] <= rf_filter_coeff473_b;
11'h7b4:
    rdata[7:0] <= rf_filter_coeff474_a;
11'h7b5:
    rdata[7:0] <= rf_filter_coeff474_b;
11'h7b6:
    rdata[7:0] <= rf_filter_coeff475_a;
11'h7b7:
    rdata[7:0] <= rf_filter_coeff475_b;
11'h7b8:
    rdata[7:0] <= rf_filter_coeff476_a;
11'h7b9:
    rdata[7:0] <= rf_filter_coeff476_b;
11'h7ba:
    rdata[7:0] <= rf_filter_coeff477_a;
11'h7bb:
    rdata[7:0] <= rf_filter_coeff477_b;
11'h7bc:
    rdata[7:0] <= rf_filter_coeff478_a;
11'h7bd:
    rdata[7:0] <= rf_filter_coeff478_b;
11'h7be:
    rdata[7:0] <= rf_filter_coeff479_a;
11'h7bf:
    rdata[7:0] <= rf_filter_coeff479_b;
11'h7c0:
    rdata[7:0] <= rf_filter_coeff480_a;
11'h7c1:
    rdata[7:0] <= rf_filter_coeff480_b;
11'h7c2:
    rdata[7:0] <= rf_filter_coeff481_a;
11'h7c3:
    rdata[7:0] <= rf_filter_coeff481_b;
11'h7c4:
    rdata[7:0] <= rf_filter_coeff482_a;
11'h7c5:
    rdata[7:0] <= rf_filter_coeff482_b;
11'h7c6:
    rdata[7:0] <= rf_filter_coeff483_a;
11'h7c7:
    rdata[7:0] <= rf_filter_coeff483_b;
11'h7c8:
    rdata[7:0] <= rf_filter_coeff484_a;
11'h7c9:
    rdata[7:0] <= rf_filter_coeff484_b;
11'h7ca:
    rdata[7:0] <= rf_filter_coeff485_a;
```

```
11'h7cb:
    rdata[7:0] <= rf_filter_coeff485_b;
11'h7cc:
    rdata[7:0] <= rf_filter_coeff486_a;
11'h7cd:
    rdata[7:0] <= rf_filter_coeff486_b;
11'h7ce:
    rdata[7:0] <= rf_filter_coeff487_a;
11'h7cf:
    rdata[7:0] <= rf_filter_coeff487_b;
11'h7d0:
    rdata[7:0] <= rf_filter_coeff488_a;
11'h7d1:
    rdata[7:0] <= rf_filter_coeff488_b;
11'h7d2:
    rdata[7:0] <= rf_filter_coeff489_a;
11'h7d3:
    rdata[7:0] <= rf_filter_coeff489_b;
11'h7d4:
    rdata[7:0] <= rf_filter_coeff490_a;
11'h7d5:
    rdata[7:0] <= rf_filter_coeff490_b;
11'h7d6:
    rdata[7:0] <= rf_filter_coeff491_a;
11'h7d7:
    rdata[7:0] <= rf_filter_coeff491_b;
11'h7d8:
    rdata[7:0] <= rf_filter_coeff492_a;
11'h7d9:
    rdata[7:0] <= rf_filter_coeff492_b;
11'h7da:
    rdata[7:0] <= rf_filter_coeff493_a;
11'h7db:
    rdata[7:0] <= rf_filter_coeff493_b;
11'h7dc:
    rdata[7:0] <= rf_filter_coeff494_a;
11'h7dd:
    rdata[7:0] <= rf_filter_coeff494_b;
11'h7de:
    rdata[7:0] <= rf_filter_coeff495_a;
11'h7df:
    rdata[7:0] <= rf_filter_coeff495_b;
11'h7e0:
    rdata[7:0] <= rf_filter_coeff496_a;
11'h7e1:
    rdata[7:0] <= rf_filter_coeff496_b;
11'h7e2:
    rdata[7:0] <= rf_filter_coeff497_a;
11'h7e3:
    rdata[7:0] <= rf_filter_coeff497_b;
11'h7e4:
    rdata[7:0] <= rf_filter_coeff498_a;
11'h7e5:
    rdata[7:0] <= rf_filter_coeff498_b;
11'h7e6:
    rdata[7:0] <= rf_filter_coeff499_a;
11'h7e7:
    rdata[7:0] <= rf_filter_coeff499_b;
11'h7e8:
    rdata[7:0] <= rf_filter_coeff500_a;
11'h7e9:
    rdata[7:0] <= rf_filter_coeff500_b;
11'h7ea:
    rdata[7:0] <= rf_filter_coeff501_a;
11'h7eb:
    rdata[7:0] <= rf_filter_coeff501_b;
11'h7ec:
    rdata[7:0] <= rf_filter_coeff502_a;
11'h7ed:
    rdata[7:0] <= rf_filter_coeff502_b;
11'h7ee:
    rdata[7:0] <= rf_filter_coeff503_a;
11'h7ef:
    rdata[7:0] <= rf_filter_coeff503_b;
11'h7f0:
    rdata[7:0] <= rf_filter_coeff504_a;
11'h7f1:
    rdata[7:0] <= rf_filter_coeff504_b;
11'h7f2:
    rdata[7:0] <= rf_filter_coeff505_a;
11'h7f3:
    rdata[7:0] <= rf_filter_coeff505_b;
11'h7f4:
    rdata[7:0] <= rf_filter_coeff506_a;
11'h7f5:
    rdata[7:0] <= rf_filter_coeff506_b;
11'h7f6:
    rdata[7:0] <= rf_filter_coeff507_a;
11'h7f7:
    rdata[7:0] <= rf_filter_coeff507_b;
11'h7f8:
    rdata[7:0] <= rf_filter_coeff508_a;
11'h7f9:
    rdata[7:0] <= rf_filter_coeff508_b;
11'h7fa:
```

```
                rdata[7:0] <= rf_filter_coeff509_a;
            11'h7fb:
                rdata[7:0] <= rf_filter_coeff509_b;
            11'h7fc:
                rdata[7:0] <= rf_filter_coeff510_a;
            11'h7fd:
                rdata[7:0] <= rf_filter_coeff510_b;
            11'h7fe:
                rdata[7:0] <= rf_filter_coeff511_a;
            11'h7ff:
                rdata[7:0] <= rf_filter_coeff511_b;
        endcase
                                                        end
                                                end
endmodule
```

trig_generator.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             trig_generator.v
// Create Date:             10/1/2015
// Last Modification:       3/29/2016
// Author:                  Julie Swift
// Description: When triggered, this sub-block generates a signal to clear
//                                                 any of the status bits (such as the overrun and
underrun)
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module trig_generator(clk, rst_n, address, wdata, xfc, trig_i2si_fifo_overrun_clr,
                      trig_i2so_fifo_underrun_clr, trig_filter_ovf_flag_clear);

    // Inputs
    input                       clk;                                // master clock
    input                       rst_n;                              // reset
    input       [10:0]          address;                            // register address
    input       [ 7:0]          wdata;                              // data to be written for a
write op
    input                       xfc;                                // transfer complete
    // Outputs
    output reg                  trig_i2si_fifo_overrun_clr;         // address = 0x008 bit 0
    output reg                  trig_i2so_fifo_underrun_clr;            // address = 0x008 bit 2
    output reg                  trig_filter_ovf_flag_clear;         // address = 0x008 bit 4

    always @ (posedge clk or negedge rst_n)
    begin
        if (~rst_n)
        begin
            trig_i2si_fifo_overrun_clr <= 0;
            trig_i2so_fifo_underrun_clr <= 0;
            trig_filter_ovf_flag_clear <= 0;
        end

        else
        begin
            // initializing trigger bits to zero
            trig_i2si_fifo_overrun_clr <= 0;
            trig_i2so_fifo_underrun_clr <= 0;
            // triggering when file transfer is complete and address being written to is 0x00c
            if     (address == 11'h008 && xfc ==1)
            begin
                // if written to bit 0 of 0x008, trig_i2si_fifo_overrun_clr is triggered
                if (wdata[0])
                    trig_i2si_fifo_overrun_clr <= 1;
                // if written to bit 2 of 0x008, trig_i2so_fifo_underrun_clr is triggered
                if (wdata[2])
                    trig_i2so_fifo_underrun_clr <= 1;
                // if written to bit 4 of 0x008, trig_filter_ovf_flag_clear is triggered
                if (wdata[4])
                    trig_filter_ovf_flag_clear <= 1;
            end
        end
    end

endmodule
```

I2C PSoC Code:

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include <project.h>
#include <device.h>
#include <is2_enable.h>
#include <chip_status.h>
#include <trig_enable.h>
#include <lpf.h>
#include <hpf.h>
#include <bpf.h>
#include <pass.h>

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    //int status = 0;
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    I2C_Start();
    I2C_EnableInt();
    I2C_Enable();
    int m=0;
    int j=0;
    uint8 status;
    int i;


    for(;;)
    {
/*
          CyDelay(500);
    if (j==0)
    {
          status = I2C_MasterWriteBuf(80, lpf, 1026, I2C_MODE_COMPLETE_XFER); //write whole coeff
register template, choose filter type
           if(status != I2C_MSTR_NO_ERROR)
        {
            I2C_MasterSendStop();
              j=1;
            break;
        }
        I2C_MasterSendStop();
        }



      /*
          Write master regiester

CyDelay(500);
    if (j==0)
    {
          status = I2C_MasterSendStart(87, 0);
          for(i=0; i<2; i++)
        {
        status = I2C_MasterWriteByte(chip[i]);
        if(status != I2C_MSTR_NO_ERROR)
        {
            I2C_MasterSendStop();\
              m=1;
```

```
            break;
        }
        }

          */

    /*
    PSOC setup with button functionality

    //Write i2s on trig off
    if(i2s_read() == 1)
    {
    I2C_MasterWriteBuf(87, i2s, 3, I2C_MODE_COMPLETE_XFER);
    while(i2s_read())
    {}
    I2C_MasterReadBuf(87, status, 1, I2C_MODE_COMPLETE_XFER);
    }

    //Write i2s off trig on
    if(trig_read() == 1)
    {
    I2C_MasterWriteBuf(87, trig, 3, I2C_MODE_COMPLETE_XFER);
    while(trig_read())
    {}
    }

    //Write LPF
    if(LPF_read() == 1)
    {
    I2C_MasterWriteBuf(87, LPF, 1026, I2C_MODE_COMPLETE_XFER);
    while(LPF_read())
    {}
    }
    */


    }
}

/* [] END OF FILE */
```

I2S PSoC Code:

```
/* ========================================
 *
 * Copyright The College of New Jersey, 2016
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF TCNJ.
 *
 * ========================================
*/

#include <project.h>
#include <stdio.h>
#include <stdlib.h>

void DmaRxConfiguration(void);
void DmaTxConfiguration(void);

/* DMA Configuration for RxDMA */
#define RxDMA_BYTES_PER_BURST       1
#define RxDMA_REQUEST_PER_BURST     1
#define RxDMA_SRC_BASE              (CYDEV_PERIPH_BASE)
#define RxDMA_DST_BASE              (CYDEV_SRAM_BASE)

/* DMA Configuration for TxDMA */
#define TxDMA_BYTES_PER_BURST       1
#define TxDMA_REQUEST_PER_BURST     1
#define TxDMA_SRC_BASE              (CYDEV_SRAM_BASE)
#define TxDMA_DST_BASE              (CYDEV_PERIPH_BASE)

/* Variable declarations for RxDMA */
uint8_t RxDMA_Chan_0;
uint8_t RxDMA_TD_0[1];
uint8_t RxDMA_Chan_1;
uint8_t RxDMA_TD_1[1];

/* Variable declarations for TxDMA */
uint8_t TxDMA_0_Chan;
uint8_t TxDMA_0_TD[1];
uint8_t TxDMA_1_Chan;
uint8_t TxDMA_1_TD[1];

/* The "N" that MATLAB prints */
#define TRANSFER_COUNT 48

/* Copy data that MATLAB produces in "data.txt" file */
volatile uint16_t signal1[TRANSFER_COUNT] =
          {0x0, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
          0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
          0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0, 0x0, 0x0, 0x0, 0x0,
          0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
          0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0};

volatile uint16_t signal2[TRANSFER_COUNT] =
          {0x0, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
          0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
          0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0, 0x0, 0x0, 0x0, 0x0,
          0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
          0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0};

#define BUFFER_SIZE 1024                                /* number of samples to receive */
volatile uint16_t * ReceivedData;     /* left IN buffer */
volatile uint16_t * dummy;                     /* right IN buffer, ignored */

int main()
{
          uint16_t i = 0;
          ReceivedData = malloc(BUFFER_SIZE*sizeof(uint16_t));
```

```
        dummy = malloc(BUFFER_SIZE*sizeof(uint16_t));

        /* Enable global interrupts */
        CyGlobalIntEnable;

        /* Initialize received data array */
        for (i = 0; i < BUFFER_SIZE; i++)
                ReceivedData[i] = i;

    /* Configure DMAs for each direction */
    DmaRxConfiguration();
    DmaTxConfiguration();

        /* Enable I2S component */
    I2S_EnableTx();     /* Enable Tx direction */
        I2S_1_EnableRx();           /* Enable Rx direction */
    I2S_Start();
    I2S_1_Start();

        CyDelay(3000);              /* Wait for data transmissions to complete */
        /* There's probably a better way to do that than simply waiting... */

        UART_Start();      /* Enabling the UART */

        /* Send the data to computer through the UART component, serial to USB */
        for(i = 0; i < BUFFER_SIZE; i++)
        {
                uint8_t part1 = (ReceivedData[i] >> 8) & 0x00ff;         /* get first byte */
                uint8_t part2 = (ReceivedData[i] >> 0) & 0x00ff;         /* get second byte */
                char part1_string[5];
                char part2_string[5];
                sprintf(part1_string, "%02X", part1);
                sprintf(part2_string, "%02X", part2);
                UART_PutString(part1_string);
        /* print first byte through UART */
                UART_PutString(",
");
                UART_PutString(part2_string);
        /* print second byte through UART */
                UART_PutString("\n\r");
        }

        /* Wait forever */
        for(;;);
}

/* Rx DMA Config */
void DmaRxConfiguration(void)
{
    /* Init DMA, 1 byte bursts, each burst requires a request */
    RxDMA_Chan_0 = RxDMA_0_DmaInitialize(RxDMA_BYTES_PER_BURST, RxDMA_REQUEST_PER_BURST,
                                         HI16(RxDMA_SRC_BASE), HI16(RxDMA_DST_BASE));
        RxDMA_Chan_1 = RxDMA_1_DmaInitialize(RxDMA_BYTES_PER_BURST, RxDMA_REQUEST_PER_BURST,
                                         HI16(RxDMA_SRC_BASE), HI16(RxDMA_DST_BASE));

    RxDMA_TD_0[0] = CyDmaTdAllocate();
        RxDMA_TD_1[0] = CyDmaTdAllocate();

        /* Configure this Td chain, get BUFFER_SIZE samples */
    CyDmaTdSetConfiguration(RxDMA_TD_0[0], 2*BUFFER_SIZE, DMA_DISABLE_TD, TD_INC_DST_ADR |
RxDMA_0__TD_TERMOUT_EN | CY_DMA_TD_SWAP_EN);
        CyDmaTdSetConfiguration(RxDMA_TD_1[0], 2*BUFFER_SIZE, DMA_DISABLE_TD, TD_INC_DST_ADR |
RxDMA_1__TD_TERMOUT_EN | CY_DMA_TD_SWAP_EN);

    /* From the I2S to the memory */
    CyDmaTdSetAddress(RxDMA_TD_0[0], LO16((uint32)I2S_1_RX_CH0_F0_PTR),
LO16((uint32)ReceivedData));
        CyDmaTdSetAddress(RxDMA_TD_1[0], LO16((uint32)I2S_1_RX_CH0_F1_PTR),
LO16((uint32)dummy));

    /* Associate the TD with the channel */
```

```
    CyDmaChSetInitialTd(RxDMA_Chan_0, RxDMA_TD_0[0]);
          CyDmaChSetInitialTd(RxDMA_Chan_1, RxDMA_TD_1[0]);

    /* Enable the channel */
    CyDmaChEnable(RxDMA_Chan_0, 1);
          CyDmaChEnable(RxDMA_Chan_1, 1);
}

/* Tx DMA Config */
void DmaTxConfiguration(void)
{
    /* Init DMA, 1 byte bursts, each burst requires a request */
    TxDMA_0_Chan = TxDMA_0_DmaInitialize(TxDMA_BYTES_PER_BURST, TxDMA_REQUEST_PER_BURST,
                                    HI16(TxDMA_SRC_BASE), HI16(TxDMA_DST_BASE));
          TxDMA_1_Chan = TxDMA_1_DmaInitialize(TxDMA_BYTES_PER_BURST, TxDMA_REQUEST_PER_BURST,
                                    HI16(TxDMA_SRC_BASE), HI16(TxDMA_DST_BASE));

    TxDMA_0_TD[0] = CyDmaTdAllocate();
    TxDMA_1_TD[0] = CyDmaTdAllocate();

          /* Configure this Td chain, send TRANSFER_COUNT samples and then restarts itself */
          CyDmaTdSetConfiguration(TxDMA_0_TD[0], 2*TRANSFER_COUNT, TxDMA_0_TD[0],
TD_INC_SRC_ADR);
    CyDmaTdSetConfiguration(TxDMA_1_TD[0], 2*TRANSFER_COUNT, TxDMA_1_TD[0], TD_INC_SRC_ADR);

          /* From the ADC to the I2S */
    CyDmaTdSetAddress(TxDMA_0_TD[0], LO16((uint32)signal1), LO16((uint32)I2S_TX_CH0_F0_PTR));
    CyDmaTdSetAddress(TxDMA_1_TD[0], LO16((uint32)signal2), LO16((uint32)I2S_TX_CH0_F1_PTR));

          /* Associate the TD with the channel */
    CyDmaChSetInitialTd(TxDMA_0_Chan, TxDMA_0_TD[0]);
    CyDmaChSetInitialTd(TxDMA_1_Chan, TxDMA_1_TD[0]);

          /* Enable the channel */
    CyDmaChEnable(TxDMA_0_Chan, 1);
    CyDmaChEnable(TxDMA_1_Chan, 1);
}

/* END OF FILE */
```

Appendix D: Test Benches

chip.v (top-level module) 347
- chip_startup_test.v 347
- chip_test_i2s_serial_enable.v 350
- chip_test_bist_enable.v 360

Filter Code 362
- filter_tf.v 362
- filter_accumulator_tf.v 393
- filter_mux_tf.v 395
- filter_round_truncate_tf.v 440
- filter_stm_tf.v 441

I2C Code 442
- i2c_reg_test_automated.v 442
- i2c_reg_testbench_write_readback.v 461

I2S Input Code 465
- bist_test.v 466
- fifo_test.v 467
- deserializer_testbench.v 470
- i2s_in_test.v 473
- i2s_in_test2.v 479
- i2s_in_test3.v 484
- i2s_in_test4.v 489
- mux_test.v 494
- synchronizer_testbench.v 496

I2S Output Code 499
- i2s_out_test.v 499
- i2s_out_test2.v 504
- i2s_out_test3.v 509
- serializer_testbench.v 514

Register Code 515
- reg_testbench.v 515
- register_testbench.v 546
- trig_generator_testbench.v 577
- trig_generator_testbench1.v 579

chip_startup_test.v:

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   00:52:12 04/04/2016
// Design Name:   chip
// Module Name:   Z:/Desktop/Main Project/chip_startup_test.v
// Project Name:  SeniorProject
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: chip
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define N 100 // number of test elements

module chip_startup_test;


          // General Inputs
          reg                     clk;
          wire                    rst_n;

    // I2S Inputs
          reg                     i2si_sck;
          wire                    i2si_ws;
          wire                    i2si_sd;

    // I2C Inputs
          reg     [2:0]           i2c_addr_bits;
          reg                     i2c_scl;
          reg                     i2c_sda_in;

          // I2S Outputs
          wire                    i2so_sck;
          wire                    i2so_ws;
          wire                    i2so_sd;

    // I2C Outputs
          wire                    i2c_sda_out;


    // General Internal Variables
    integer                         clk_count;
// clock counter



    // I2S Internal Variables
    reg     [15:0]                  i2s_test_data [`N-1:0] [0:1];
// [Bits Per Word] i2s_test_data [# of entities in test] [Left/Right]
    reg                             i2s_sck_dl;
// serial clock delay
    integer                         i2s_sck_cnt;
// serial clock counter
    integer                         i2s_bit_cnt;
// bit number counter
```

```
    integer                         i2s_lr_cnt;
// left right counter
    integer                         i2s_word_cnt;
// word counter
    parameter                       i2s_cyc_per_half_sck = 33;
// about (100 MHz / 1.536 MHz)/2
    parameter                       i2s_bit_tc =  15;
// number of bits in a word
    integer                         index1;
// counter for instantiating i2s_test_data
    integer                         index2;
// counter for instantiating i2s_test_data

    // Instantiate the Unit Under Test (UUT)
          chip uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .i2si_sck(i2si_sck),
                    .i2si_ws(i2si_ws),
                    .i2si_sd(i2si_sd),
                    .i2so_sck(i2so_sck),
                    .i2so_ws(i2so_ws),
                    .i2so_sd(i2so_sd),
                    .i2c_addr_bits(i2c_addr_bits),
                    .i2c_scl(i2c_scl),
                    .i2c_sda_in(i2c_sda_in),
                    .i2c_sda_out(i2c_sda_out)
          );

          initial begin
                    // Initialize Inputs
                    clk = 0;

                    i2si_sck = 0;

                    i2c_addr_bits = 0;
                    i2c_scl = 0;
                    i2c_sda_in = 0;


                    // Add stimulus here

          end

          // Generates master clock signal
    always
    begin
        clk_count = 0;
        forever
        begin
            #5 clk = ~clk;
// 100 MHz clock rate (100MHz/10^9)/2
            clk_count = clk_count + 1;
        end
    end


    // Defines the counters to determine which bit to input into serial data for I2S Interface
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            i2s_sck_cnt     <= 0;
// counts master clock cycles, causes sck to toggle each time it hits i2s_cyc_per_half_sck
            i2s_bit_cnt     <= 0;
// count number of bits
            i2s_word_cnt    <= 0;
// count the word number
            i2s_lr_cnt      <= 0;
// left=0 and right=1
```

```
            i2si_sck        <= 0;
// serial clock
            i2s_sck_dl      <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (i2s_sck_cnt == i2s_cyc_per_half_sck-1)
// i2s_cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                i2s_sck_cnt <= 0;
// reset serial clock counter
                i2si_sck <= ~i2si_sck;
// toggle serial clock
            end
            else
                i2s_sck_cnt <= i2s_sck_cnt + 1;
// increment serial clock counter

            i2s_sck_dl<=i2si_sck;
// generate 1 cycle delay of i2si_sck
            if(i2si_sck & ~i2s_sck_dl)
// on a positive transition of sck...

            begin
                if (i2s_bit_cnt==i2s_bit_tc)
// i2s_bit_tc = 15
                begin
                    if (i2s_lr_cnt == 1)
// if right
                    begin
                        i2s_word_cnt<=i2s_word_cnt+1;
// words in the testbench array
                        i2s_lr_cnt<=0;
// set to left
                    end
                    else
                        i2s_lr_cnt<=1;
// set to right
                    i2s_bit_cnt<=0;
// reset bit counter
                end
                else
                    i2s_bit_cnt<=i2s_bit_cnt+1;
// increment bit counter
            end
        end
    end


    // Set rst_n high after 10 clock cycles
    assign rst_n = !(clk_count < 20);

    // Set which word channel to read from for I2S Interface
    assign i2si_ws = ((0<=i2s_bit_cnt& i2s_bit_cnt<=16'd14)&i2s_lr_cnt==1) |
((i2s_bit_cnt==16'd15)&(i2s_lr_cnt==0));

    // Set which serial data bit to input
    assign i2si_sd = i2s_test_data [i2s_word_cnt][i2s_lr_cnt][i2s_bit_tc-i2s_bit_cnt];

endmodule
```

chip_test_i2s_serial_enable.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   chip_test_i2s_serial_enable.v
// Create Date:   2/17/2016
// Last Edit:     5/8/16
// Author:        Kevin Cao, Whitley Forman
//
// Description:   Testing pass through of data. Input is a square wave. Input data should match
output data in produced text files
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define N 10000 // number of test elements

module chip_test1;

          // General Inputs
          reg                     clk;
          wire                    rst_n;

    // I2S Inputs
          reg                     i2si_sck;
          wire                    i2si_ws;
          wire                    i2si_sd;

    // I2C Inputs
          reg     [2:0]           i2c_addr_bits;
          reg                     i2c_scl;
          reg                     i2c_sda_in;

          // I2S Outputs
          wire                    i2so_sck;
          wire                    i2so_ws;
          wire                    i2so_sd;

    // I2C Outputs
          wire                    i2c_sda_od;


    // General Internal Variables
    integer                         clk_count;
// clock counter



    // I2S Internal Variables
    reg     [15:0]                  i2s_test_data [`N-1:0] [0:1];
// [Bits Per Word] i2s_test_data [# of entities in test] [Left/Right]
    reg                             i2s_sck_dl;
// serial clock delay
    integer                         i2s_sck_cnt;
// serial clock counter
    integer                         i2s_bit_cnt;
// bit number counter
    integer                         i2s_lr_cnt;
// left right counter
    integer                         i2s_word_cnt;
// word counter
    parameter                       i2s_cyc_per_half_sck = 33;
// about (100 MHz / 1.536 MHz)/2
    parameter                       i2s_bit_tc =  15;
// number of bits in a word
    integer                         index1;
// counter for instantiating i2s_test_data
    integer                         index2 = 0;
// counter for instantiating i2s_test_data
```

```
    // I2C Internal Variables
                    reg [3:0] i2c_bit_count;
                    reg [3:0] i2c_master_state;
                    reg [9:0] i2c_data_byte;
                    reg i2c_slave_acknowledgement;
                    reg [7:0] i2c_test_data [2:0];
                    reg [10:0] i2c_reg_addr_test_data [1:0];
                    reg [7:0] i2c_slave_addr_test_data [1:0];
                    reg [8:0] i2c_count;


    // Data Capture Internal Variables
    reg                             ws_d1;
    reg                             ws_d2;
    wire                            ws_transition;
    reg                             i2so_sck_d1;
    wire                            i2so_sck_transition;
    reg         [31:0]              word;
    integer                         data_out;
    reg                             output_to_txt = 1;
    integer                         data_in;
    integer                         index3;
// counter for outputing to text file of list of inputs

    // Instantiate the Unit Under Test (UUT)
          chip uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .i2si_sck(i2si_sck),
                    .i2si_ws(i2si_ws),
                    .i2si_sd(i2si_sd),
                    .i2so_sck(i2so_sck),
                    .i2so_ws(i2so_ws),
                    .i2so_sd(i2so_sd),
                    .i2c_addr_bits(i2c_addr_bits),
                    .i2c_scl(i2c_scl),
                    .i2c_sda_in(i2c_sda_in),
                    .i2c_sda_od(i2c_sda_od)
          );

          initial begin
                    // Initialize Inputs
                    clk = 0;

                    i2si_sck = 0;


                    i2c_addr_bits = 3'b101;
                    i2c_scl = 0;
                    i2c_sda_in = 1;
                    i2c_count = 0;

        data_in = $fopen("chip_test_i2s_serial_enable_input.txt");
// Open chip_test_i2s_serial_enable_input.txt
        data_out = $fopen("chip_test_i2s_serial_enable_output.txt");
// Open chip_test_i2s_serial_enable_output.txt

        // Instantiate I2S Test Data: Method 1; Irrelevant in testing the BIST
        for(index1 = 0; index1 < `N; index1 = index1 + 1)
        begin
            for(index2 = 0; index2 < 2; index2 = index2 + 1)
            begin
                i2s_test_data [index1] [index2] = $random;
            end
        end


        for(index3 = 0; index3 < `N; index3 = index3 + 1)
        begin
            $fdisplay (data_in, "%h", {i2s_test_data [index3] [0], i2s_test_data [index3] [1]});
        end
```

```
        // Instantiate I2C Test Data
       //   i2c_test_data[0] = 8'b00100000;                        // data for h400
                  // i2c_test_data[1] = 8'b00000000;                       //data for h400
coeff0_a
                  // i2c_test_data[2] = 8'b00010000;                       //data for h401
coeff0_b



      //i2c_reg_addr_test_data [0] = 11'b00000000000;    //11'h004 = rf_i2s_bist_en register +
filter registers
                  //i2c_reg_addr_test_data [1] = 11'b10000000000; //11'h400 = rf_i2s_bist_en
register + filter registers
                  //i2c_slave_addr_test_data [0] = 8'b10101011;            //current slave
address w/ 3'b101 as i2c_addr_bits
                  //i2c_slave_addr_test_data [1] = 8'b10101011;            //current slave
address w/ 3'b101 as i2c_addr_bits

        #1 $fclose(data_in);

         end

    // Generates master clock signal
    always
    begin
        clk_count = 0;
        forever
        begin
            #5 clk = ~clk;
// 100 MHz clock rate (100MHz/10^9)/2
            clk_count = clk_count + 1;
        end
    end


    // Defines the counters to determine which bit to input into serial data for I2S Interface
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            i2s_sck_cnt     <= 0;
// counts master clock cycles, causes sck to toggle each time it hits i2s_cyc_per_half_sck
            i2s_bit_cnt     <= 0;
// count number of bits
            i2s_word_cnt    <= 0;
// count the word number
            i2s_lr_cnt      <= 0;
// left=0 and right=1
            i2si_sck        <= 0;
// serial clock
            i2s_sck_dl      <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (i2s_sck_cnt == i2s_cyc_per_half_sck-1)
// i2s_cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                i2s_sck_cnt <= 0;
// reset serial clock counter
                i2si_sck <= ~i2si_sck;
// toggle serial clock
            end
            else
                i2s_sck_cnt <= i2s_sck_cnt + 1;
// increment serial clock counter
```

```
            i2s_sck_dl<=i2si_sck;
// generate 1 cycle delay of i2si_sck
            if(i2si_sck & ~i2s_sck_dl)
// on a positive transition of sck...

            begin
                if (i2s_bit_cnt==i2s_bit_tc)
// i2s_bit_tc = 15
                begin
                    if (i2s_lr_cnt == 1)
// if right
                    begin
                        i2s_word_cnt<=i2s_word_cnt+1;
// words in the testbench array
                        i2s_lr_cnt<=0;
// set to left
                    end
                    else
                        i2s_lr_cnt<=1;
// set to right
                    i2s_bit_cnt<=0;
// reset bit counter
                end
                else
                    i2s_bit_cnt<=i2s_bit_cnt+1;
// increment bit counter
            end
        end
    end


    // Set rst_n high after 10 clock cycles
    assign rst_n = !(clk_count < 20);

    // Set which word channel to read from for I2S Interface
    assign i2si_ws = ((0<=i2s_bit_cnt& i2s_bit_cnt<=16'd14)&i2s_lr_cnt==1) |
((i2s_bit_cnt==16'd15)&(i2s_lr_cnt==0));

    // Set which serial data bit to input
    assign i2si_sd = i2s_test_data [i2s_word_cnt][i2s_lr_cnt][i2s_bit_tc-i2s_bit_cnt];


//-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
--------------------------------

    // I2C TEST BENCH PORTION



    always #1250 i2c_scl = !i2c_scl;  //create i2c_scl @ 400khz


    always @ (posedge clk)
    begin
        i2c_count = i2c_count + 1;
        if (i2c_count == 250)                           //reset i2c_count on full scl cycle
        begin
        i2c_count <= 0;
        end
    end


    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            i2c_bit_count<=8;
            i2c_master_state<=4'b0000;
            i2c_data_byte <= 0;
            i2c_slave_acknowledgement <= 0;
```

```
end

else if ((i2c_master_state == 4'b0000) & (i2c_count == 191))  //start condition
begin
    i2c_sda_in <= 0;
    i2c_master_state <= 4'b0001;
end

else if ((i2c_master_state == 4'b0001))    //serialize slave address and RW bit
begin
    if((i2c_count == 61) & (i2c_bit_count > 0))
    begin
        i2c_bit_count <= i2c_bit_count - 1;
        if (i2c_bit_count == 0)
        begin
            i2c_sda_in <= 1;
        end
        if (i2c_bit_count > 0)
        begin
            i2c_sda_in <= i2c_slave_addr_test_data [0] [i2c_bit_count - 1];
        end
    end

    else if((i2c_count == 61) & (i2c_bit_count == 0))
    begin
        i2c_bit_count <= 8;
        //i2c_slave_acknowledgement <= i2c_sda_od;
    end

    else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
  //i2c_slave_acknowledgement == 1
    begin
        //i2c_slave_acknowledgement <= 0;
        i2c_master_state <= 4'b0010;
    end
end

else if ((i2c_master_state == 4'b0010))    //serialize reg address part 1
begin
    if((i2c_count == 61) & (i2c_bit_count > 0))
    begin
        i2c_bit_count <= i2c_bit_count - 1;
        if (i2c_bit_count == 0)
        begin
            i2c_sda_in <= 1;
        end
        if (i2c_bit_count > 0)
        begin
            i2c_sda_in <= i2c_reg_addr_test_data [0][i2c_bit_count + 2];
        end
    end

    else if((i2c_count == 61) & (i2c_bit_count == 0))
    begin
        i2c_bit_count <= 8;
                                        i2c_sda_in <= 1;
        //i2c_slave_acknowledgement <= i2c_sda_od;
    end

    else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
    begin
        //i2c_slave_acknowledgement <= 0;
        i2c_master_state <= 4'b0011;
    end
end

else if ((i2c_master_state == 4'b0011))    //serialize reg address part 2
begin
    if((i2c_count == 61) & (i2c_bit_count > 5))
    begin
```

```
                i2c_bit_count <= i2c_bit_count - 1;
                if (i2c_bit_count == 0)
                begin
                    i2c_sda_in <= 1;
                end
                if (i2c_bit_count > 0)
                begin
                    i2c_sda_in <= i2c_reg_addr_test_data [0][i2c_bit_count - 6];
                end
            end

                                        if((i2c_count == 61) & ((i2c_bit_count > 0) &&
(i2c_bit_count < 6))) //Unwanted bits set to 1
            begin
                i2c_bit_count <= i2c_bit_count - 1;
                i2c_sda_in <= 1;
            end

            else if((i2c_count == 61) & (i2c_bit_count == 0))
            begin
                i2c_bit_count <= 8;
                                                i2c_sda_in <= 1;
                //i2c_slave_acknowledgement <= i2c_sda_od;
            end

            else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
            begin
                //i2c_slave_acknowledgement <= 0;
                i2c_master_state <= 4'b0100;
            end
        end

        else if ((i2c_master_state == 4'b0100) & (i2c_data_byte <= 1) & (i2c_slave_addr_test_data
[0] [7] == 1))    //serialize data
        begin
            if((i2c_count == 61) & (i2c_bit_count > 0))
            begin
                i2c_bit_count <= i2c_bit_count - 1;
                if (i2c_bit_count == 0)
                begin
                    i2c_sda_in <= 1;
                end
                if (i2c_bit_count > 0)
                begin
                    i2c_sda_in <= i2c_test_data [i2c_data_byte] [i2c_bit_count - 1];
                end
            end

            else if((i2c_count == 61) & (i2c_bit_count == 0))
            begin
                i2c_bit_count <= 8;
                                                i2c_sda_in <= 1;
                //i2c_slave_acknowledgement <= i2c_sda_od;
            end

            else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
            begin
                //i2c_slave_acknowledgement <= 0;
                i2c_data_byte <= i2c_data_byte + 1;
            end

            else if (i2c_data_byte == 1)
            begin
                //i2c_data_byte <= 0;
                i2c_master_state <= 4'b0101;            //enter stop condition
                //i2c_slave_acknowledgement <= 0;
                i2c_bit_count <= 8;
            end
        end

        else if (i2c_master_state == 4'b0101)           // stop condition
```

```
        begin
            if (i2c_count == 61)
            begin
                i2c_sda_in <= 0;
            end

            if (i2c_count == 191)
            begin
                i2c_sda_in <=1;
                i2c_master_state <= 4'b0110;
            end
        end


          ////////////////Transaction 2 for
data//////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////

          else if ((i2c_master_state == 4'b0110) & (i2c_count == 191))  //start condition
        begin
            i2c_sda_in <= 0;
            i2c_master_state <= 4'b1001;
        end

        else if ((i2c_master_state == 4'b1001))    //serialize slave address and RW bit
        begin
            if((i2c_count == 61) & (i2c_bit_count > 0))
            begin
                i2c_bit_count <= i2c_bit_count - 1;
                if (i2c_bit_count == 0)
                begin
                    i2c_sda_in <= 1;
                end
                if (i2c_bit_count > 0)
                begin
                    i2c_sda_in <= i2c_slave_addr_test_data [1][i2c_bit_count - 1];
                end
            end

            else if((i2c_count == 61) & (i2c_bit_count == 0))
            begin
                i2c_bit_count <= 8;
                //i2c_slave_acknowledgement <= i2c_sda_od;
            end

            else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
          //i2c_slave_acknowledgement == 1
            begin
                //i2c_slave_acknowledgement <= 0;
                i2c_master_state <= 4'b1010;
            end
        end

        else if ((i2c_master_state == 4'b1010))    //serialize reg address part 1
        begin
            if((i2c_count == 61) & (i2c_bit_count > 0))
            begin
                i2c_bit_count <= i2c_bit_count - 1;
                if (i2c_bit_count == 0)
                begin
                    i2c_sda_in <= 1;
                end
                else if (i2c_bit_count > 0)
                begin
                    i2c_sda_in <= i2c_reg_addr_test_data [1][i2c_bit_count + 2];
                end
            end

            else if((i2c_count == 61) & (i2c_bit_count == 0))
            begin
                i2c_bit_count <= 8;
```

```
                                                i2c_sda_in <= 1;
                //i2c_slave_acknowledgement <= i2c_sda_od;
            end

            else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
            begin
                //i2c_slave_acknowledgement <= 0;
                i2c_master_state <= 4'b1011;
            end
        end

        else if ((i2c_master_state == 4'b1011))    //serialize reg address part 2
        begin
            if((i2c_count == 61) & (i2c_bit_count > 5))
            begin
                i2c_bit_count <= i2c_bit_count - 1;
                if (i2c_bit_count == 0)
                begin
                    i2c_sda_in <= 1;
                end
                if (i2c_bit_count > 0)
                begin
                    i2c_sda_in <= i2c_reg_addr_test_data [1][i2c_bit_count - 6];
                end
            end

                                        if((i2c_count == 61) & ((i2c_bit_count > 0) &&
(i2c_bit_count < 6))) //Unwanted bits set to 1
            begin
                i2c_bit_count <= i2c_bit_count - 1;
                i2c_sda_in <= 1;
            end

            else if((i2c_count == 61) & (i2c_bit_count == 0))
            begin
                i2c_bit_count <= 8;
                                                i2c_sda_in <= 1;
                //i2c_slave_acknowledgement <= i2c_sda_od;
            end

            else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
            begin
                //i2c_slave_acknowledgement <= 0;
                i2c_master_state <= 4'b1100;
            end
        end

        else if ((i2c_master_state == 4'b1100) & (i2c_data_byte <= 3) & (i2c_slave_addr_test_data
[1][7] == 1))    //serialize data ///   & (i2c_data_byte <= 1025) &
        begin
            if((i2c_count == 61) & (i2c_bit_count > 0))
            begin
                i2c_bit_count <= i2c_bit_count - 1;
                if (i2c_bit_count == 0)
                begin
                    i2c_sda_in <= 1;
                end
                else if (i2c_bit_count > 0)
                begin
                    i2c_sda_in <= i2c_test_data [i2c_data_byte] [i2c_bit_count - 1];
                end
            end

            else if((i2c_count == 61) & (i2c_bit_count == 0))
            begin
                i2c_bit_count <= 8;
                                                i2c_sda_in <= 1;
                //i2c_slave_acknowledgement <= i2c_sda_od;
            end

            else if((i2c_count == 191) & (i2c_bit_count == 8) & (i2c_sda_od == 1))
```

```
            begin
                //i2c_slave_acknowledgement <= 0;
                i2c_data_byte <= i2c_data_byte + 1;
            end

            else if (i2c_data_byte == 3) ///(i2c_data_byte == 1025) for full coeff input
            begin
                i2c_data_byte <= 0;
                i2c_master_state <= 4'b1101;             //enter stop condition
                //i2c_slave_acknowledgement <= 0;
                i2c_bit_count <= 8;
            end
        end

        else if (i2c_master_state == 4'b1101)            // stop condition
        begin
            if (i2c_count == 61)
            begin
                i2c_sda_in <= 0;
            end

            if (i2c_count == 191)
            begin
                i2c_sda_in <=1;
                i2c_master_state <= 4'b1110;
            end
        end
      end



//-------------------------------------------------------------------------------------------------
-------------------------------------------------------

//Capturing Output Data to print to chip_test_i2s_serial_enable_output.txt

    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            i2so_sck_dl <= 0;
        end
        else
        begin
            i2so_sck_dl <= i2so_sck;
        end
    end

    assign i2so_sck_transition = i2so_sck & ~i2so_sck_dl;

    // Creates a delay of word select signal, used to help in comparison test
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            ws_d1 <= 0;
            ws_d2 <= 0;
        end
        else if(i2so_sck_transition)
        begin
            ws_d1 <= i2so_ws;
// generate 1 cycle delay of i2so_ws
            ws_d2 <= ws_d1;
// generate 2nd cycle delay of i2so_ws
        end
    end

    assign ws_transition = ~ws_d1 & ws_d2;
// level to pulse converter when ws goes from high to low
```

```
    // Creates 32 bit words from the serial data being outputted to be compared with the words
being inputted
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            word <= 32'b0;
        end
        else if(i2so_sck_transition)
        begin
            word[31:1] <= word[30:0];
            word[0] <= i2so_sd;
        end
    end


    // Print output data to chip_test_i2s_serial_enable_output.txt
    always @(posedge clk)
    begin
        if(output_to_txt)
        begin
            if(ws_transition && i2so_sck_transition)
            begin
                if(word === 32'hxxxxxxxx)
                begin
                    output_to_txt = 0;
                    #1 $fclose(data_out);
                end
                else
                begin
                    $fdisplay (data_out, "%h", word);
                end
            end
        end
    end


endmodule
```

chip_test_bist_enable.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   bist_test.v
// Create Date:   10/20/2015
// Last Edit:     3/20/16
// Author:        Zachary Nelson
//
// Description:     Creates 32 bit sawtooth wave
//                  Ensuring:
//                      BIST starts at correct value
//                      Increments by specified amount
//                      Resets to starting value when reaching upper limit value
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module bist_test;

    // Inputs
    reg                             clk;
    wire                            rst_n;
    wire                            sck_transition;
    reg         [11:0]              rf_bist_start_val;
    reg         [7:0]               rf_bist_inc;
    reg         [11:0]              rf_bist_up_limit;

    // Internal Variables
    reg                             i2si_sck;
    reg                             sck_d1;                             // serial clock delay
    reg         [31:0]              count;
    reg         [31:0]              sck_cnt;                            // serial clock counter
    reg         [31:0]              cyc_per_half_sck = 40;              // about (100 MHz / 1.44
MHz)/2


    // Outputs
    wire        [31:0]              i2si_bist_out_data;
    wire                            i2si_bist_out_xfc;

    // Instantiate the Unit Under Test (UUT)
    i2si_bist_gen uut (
        .clk(clk),
        .rst_n(rst_n),
        .sck_transition(sck_transition),
        .rf_bist_start_val(rf_bist_start_val),
        .rf_bist_inc(rf_bist_inc),
        .rf_bist_up_limit(rf_bist_up_limit),
        .i2si_bist_out_data(i2si_bist_out_data),
        .i2si_bist_out_xfc(i2si_bist_out_xfc)
    );

    initial
    begin
        // Initialize Inputs
        clk = 0;
        i2si_sck=0;
        rf_bist_start_val = 12'h001;
        rf_bist_inc = 12'h001;
        rf_bist_up_limit = 12'h019;
    end

    always
    begin
        count = 0;                                              // set clock counter to zero
    forever
        begin
            #5 clk = ~clk;                                      // 100 MHz clock
```

```
            count = count + 1;                                  // increment clock counter
        end
    end


    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt<=0;                                         // counts master clock cycles,
causes sck to toggle each time it hits cyc_per_half_sck
            i2si_sck<=0;                                        // serial clock
            sck_d1<=0;                                          // serial clock delayed by one
clock cycle
        end
        else
            sck_cnt <= sck_cnt + 1;                             // increment serial clock counter
        sck_d1<=i2si_sck;                                       // generate 1 cycle delay of
i2si_sck


        if (sck_cnt == cyc_per_half_sck-1)                  // cyc_per_half_sck ~ (100 MHz/1.44
MHz)/2
        begin
            sck_cnt <= 0;                                   // reset serial clock counter
            i2si_sck <= ~i2si_sck;                          // toggle serial clock
        end
    end


    assign sck_transition = i2si_sck & ~sck_d1;
    assign rst_n = !(count < 20);

endmodule
```

filter_tf.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_tf.v
// Create Date:             11/28/2015
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps


module filter_tf;

          // Inputs
          reg clk;
          reg rst_n;
          reg [31:0] aud_in;
          reg aud_in_rts;
          reg aud_out_rtr;
          reg rf_filter_shift;
          reg rf_filter_clip_en;
          reg [7:0] rf_filter_coeff0_a;
          reg [7:0] rf_filter_coeff0_b;
          reg [7:0] rf_filter_coeff1_a;
          reg [7:0] rf_filter_coeff1_b;
          reg [7:0] rf_filter_coeff2_a;
          reg [7:0] rf_filter_coeff2_b;
          reg [7:0] rf_filter_coeff3_a;
          reg [7:0] rf_filter_coeff3_b;
          reg [7:0] rf_filter_coeff4_a;
          reg [7:0] rf_filter_coeff4_b;
          reg [7:0] rf_filter_coeff5_a;
          reg [7:0] rf_filter_coeff5_b;
          reg [7:0] rf_filter_coeff6_a;
          reg [7:0] rf_filter_coeff6_b;
          reg [7:0] rf_filter_coeff7_a;
          reg [7:0] rf_filter_coeff7_b;
          reg [7:0] rf_filter_coeff8_a;
          reg [7:0] rf_filter_coeff8_b;
          reg [7:0] rf_filter_coeff9_a;
          reg [7:0] rf_filter_coeff9_b;
          reg [7:0] rf_filter_coeff10_a;
          reg [7:0] rf_filter_coeff10_b;
          reg [7:0] rf_filter_coeff11_a;
          reg [7:0] rf_filter_coeff11_b;
          reg [7:0] rf_filter_coeff12_a;
          reg [7:0] rf_filter_coeff12_b;
          reg [7:0] rf_filter_coeff13_a;
          reg [7:0] rf_filter_coeff13_b;
          reg [7:0] rf_filter_coeff14_a;
          reg [7:0] rf_filter_coeff14_b;
          reg [7:0] rf_filter_coeff15_a;
          reg [7:0] rf_filter_coeff15_b;
          reg [7:0] rf_filter_coeff16_a;
          reg [7:0] rf_filter_coeff16_b;
          reg [7:0] rf_filter_coeff17_a;
          reg [7:0] rf_filter_coeff17_b;
          reg [7:0] rf_filter_coeff18_a;
          reg [7:0] rf_filter_coeff18_b;
          reg [7:0] rf_filter_coeff19_a;
          reg [7:0] rf_filter_coeff19_b;
          reg [7:0] rf_filter_coeff20_a;
          reg [7:0] rf_filter_coeff20_b;
          reg [7:0] rf_filter_coeff21_a;
          reg [7:0] rf_filter_coeff21_b;
          reg [7:0] rf_filter_coeff22_a;
          reg [7:0] rf_filter_coeff22_b;
          reg [7:0] rf_filter_coeff23_a;
          reg [7:0] rf_filter_coeff23_b;
```

```
reg [7:0] rf_filter_coeff24_a;
reg [7:0] rf_filter_coeff24_b;
reg [7:0] rf_filter_coeff25_a;
reg [7:0] rf_filter_coeff25_b;
reg [7:0] rf_filter_coeff26_a;
reg [7:0] rf_filter_coeff26_b;
reg [7:0] rf_filter_coeff27_a;
reg [7:0] rf_filter_coeff27_b;
reg [7:0] rf_filter_coeff28_a;
reg [7:0] rf_filter_coeff28_b;
reg [7:0] rf_filter_coeff29_a;
reg [7:0] rf_filter_coeff29_b;
reg [7:0] rf_filter_coeff30_a;
reg [7:0] rf_filter_coeff30_b;
reg [7:0] rf_filter_coeff31_a;
reg [7:0] rf_filter_coeff31_b;
reg [7:0] rf_filter_coeff32_a;
reg [7:0] rf_filter_coeff32_b;
reg [7:0] rf_filter_coeff33_a;
reg [7:0] rf_filter_coeff33_b;
reg [7:0] rf_filter_coeff34_a;
reg [7:0] rf_filter_coeff34_b;
reg [7:0] rf_filter_coeff35_a;
reg [7:0] rf_filter_coeff35_b;
reg [7:0] rf_filter_coeff36_a;
reg [7:0] rf_filter_coeff36_b;
reg [7:0] rf_filter_coeff37_a;
reg [7:0] rf_filter_coeff37_b;
reg [7:0] rf_filter_coeff38_a;
reg [7:0] rf_filter_coeff38_b;
reg [7:0] rf_filter_coeff39_a;
reg [7:0] rf_filter_coeff39_b;
reg [7:0] rf_filter_coeff40_a;
reg [7:0] rf_filter_coeff40_b;
reg [7:0] rf_filter_coeff41_a;
reg [7:0] rf_filter_coeff41_b;
reg [7:0] rf_filter_coeff42_a;
reg [7:0] rf_filter_coeff42_b;
reg [7:0] rf_filter_coeff43_a;
reg [7:0] rf_filter_coeff43_b;
reg [7:0] rf_filter_coeff44_a;
reg [7:0] rf_filter_coeff44_b;
reg [7:0] rf_filter_coeff45_a;
reg [7:0] rf_filter_coeff45_b;
reg [7:0] rf_filter_coeff46_a;
reg [7:0] rf_filter_coeff46_b;
reg [7:0] rf_filter_coeff47_a;
reg [7:0] rf_filter_coeff47_b;
reg [7:0] rf_filter_coeff48_a;
reg [7:0] rf_filter_coeff48_b;
reg [7:0] rf_filter_coeff49_a;
reg [7:0] rf_filter_coeff49_b;
reg [7:0] rf_filter_coeff50_a;
reg [7:0] rf_filter_coeff50_b;
reg [7:0] rf_filter_coeff51_a;
reg [7:0] rf_filter_coeff51_b;
reg [7:0] rf_filter_coeff52_a;
reg [7:0] rf_filter_coeff52_b;
reg [7:0] rf_filter_coeff53_a;
reg [7:0] rf_filter_coeff53_b;
reg [7:0] rf_filter_coeff54_a;
reg [7:0] rf_filter_coeff54_b;
reg [7:0] rf_filter_coeff55_a;
reg [7:0] rf_filter_coeff55_b;
reg [7:0] rf_filter_coeff56_a;
reg [7:0] rf_filter_coeff56_b;
reg [7:0] rf_filter_coeff57_a;
reg [7:0] rf_filter_coeff57_b;
reg [7:0] rf_filter_coeff58_a;
reg [7:0] rf_filter_coeff58_b;
reg [7:0] rf_filter_coeff59_a;
```

```
reg [7:0] rf_filter_coeff59_b;
reg [7:0] rf_filter_coeff60_a;
reg [7:0] rf_filter_coeff60_b;
reg [7:0] rf_filter_coeff61_a;
reg [7:0] rf_filter_coeff61_b;
reg [7:0] rf_filter_coeff62_a;
reg [7:0] rf_filter_coeff62_b;
reg [7:0] rf_filter_coeff63_a;
reg [7:0] rf_filter_coeff63_b;
reg [7:0] rf_filter_coeff64_a;
reg [7:0] rf_filter_coeff64_b;
reg [7:0] rf_filter_coeff65_a;
reg [7:0] rf_filter_coeff65_b;
reg [7:0] rf_filter_coeff66_a;
reg [7:0] rf_filter_coeff66_b;
reg [7:0] rf_filter_coeff67_a;
reg [7:0] rf_filter_coeff67_b;
reg [7:0] rf_filter_coeff68_a;
reg [7:0] rf_filter_coeff68_b;
reg [7:0] rf_filter_coeff69_a;
reg [7:0] rf_filter_coeff69_b;
reg [7:0] rf_filter_coeff70_a;
reg [7:0] rf_filter_coeff70_b;
reg [7:0] rf_filter_coeff71_a;
reg [7:0] rf_filter_coeff71_b;
reg [7:0] rf_filter_coeff72_a;
reg [7:0] rf_filter_coeff72_b;
reg [7:0] rf_filter_coeff73_a;
reg [7:0] rf_filter_coeff73_b;
reg [7:0] rf_filter_coeff74_a;
reg [7:0] rf_filter_coeff74_b;
reg [7:0] rf_filter_coeff75_a;
reg [7:0] rf_filter_coeff75_b;
reg [7:0] rf_filter_coeff76_a;
reg [7:0] rf_filter_coeff76_b;
reg [7:0] rf_filter_coeff77_a;
reg [7:0] rf_filter_coeff77_b;
reg [7:0] rf_filter_coeff78_a;
reg [7:0] rf_filter_coeff78_b;
reg [7:0] rf_filter_coeff79_a;
reg [7:0] rf_filter_coeff79_b;
reg [7:0] rf_filter_coeff80_a;
reg [7:0] rf_filter_coeff80_b;
reg [7:0] rf_filter_coeff81_a;
reg [7:0] rf_filter_coeff81_b;
reg [7:0] rf_filter_coeff82_a;
reg [7:0] rf_filter_coeff82_b;
reg [7:0] rf_filter_coeff83_a;
reg [7:0] rf_filter_coeff83_b;
reg [7:0] rf_filter_coeff84_a;
reg [7:0] rf_filter_coeff84_b;
reg [7:0] rf_filter_coeff85_a;
reg [7:0] rf_filter_coeff85_b;
reg [7:0] rf_filter_coeff86_a;
reg [7:0] rf_filter_coeff86_b;
reg [7:0] rf_filter_coeff87_a;
reg [7:0] rf_filter_coeff87_b;
reg [7:0] rf_filter_coeff88_a;
reg [7:0] rf_filter_coeff88_b;
reg [7:0] rf_filter_coeff89_a;
reg [7:0] rf_filter_coeff89_b;
reg [7:0] rf_filter_coeff90_a;
reg [7:0] rf_filter_coeff90_b;
reg [7:0] rf_filter_coeff91_a;
reg [7:0] rf_filter_coeff91_b;
reg [7:0] rf_filter_coeff92_a;
reg [7:0] rf_filter_coeff92_b;
reg [7:0] rf_filter_coeff93_a;
reg [7:0] rf_filter_coeff93_b;
reg [7:0] rf_filter_coeff94_a;
reg [7:0] rf_filter_coeff94_b;
```

```
reg [7:0] rf_filter_coeff95_a;
reg [7:0] rf_filter_coeff95_b;
reg [7:0] rf_filter_coeff96_a;
reg [7:0] rf_filter_coeff96_b;
reg [7:0] rf_filter_coeff97_a;
reg [7:0] rf_filter_coeff97_b;
reg [7:0] rf_filter_coeff98_a;
reg [7:0] rf_filter_coeff98_b;
reg [7:0] rf_filter_coeff99_a;
reg [7:0] rf_filter_coeff99_b;
reg [7:0] rf_filter_coeff100_a;
reg [7:0] rf_filter_coeff100_b;
reg [7:0] rf_filter_coeff101_a;
reg [7:0] rf_filter_coeff101_b;
reg [7:0] rf_filter_coeff102_a;
reg [7:0] rf_filter_coeff102_b;
reg [7:0] rf_filter_coeff103_a;
reg [7:0] rf_filter_coeff103_b;
reg [7:0] rf_filter_coeff104_a;
reg [7:0] rf_filter_coeff104_b;
reg [7:0] rf_filter_coeff105_a;
reg [7:0] rf_filter_coeff105_b;
reg [7:0] rf_filter_coeff106_a;
reg [7:0] rf_filter_coeff106_b;
reg [7:0] rf_filter_coeff107_a;
reg [7:0] rf_filter_coeff107_b;
reg [7:0] rf_filter_coeff108_a;
reg [7:0] rf_filter_coeff108_b;
reg [7:0] rf_filter_coeff109_a;
reg [7:0] rf_filter_coeff109_b;
reg [7:0] rf_filter_coeff110_a;
reg [7:0] rf_filter_coeff110_b;
reg [7:0] rf_filter_coeff111_a;
reg [7:0] rf_filter_coeff111_b;
reg [7:0] rf_filter_coeff112_a;
reg [7:0] rf_filter_coeff112_b;
reg [7:0] rf_filter_coeff113_a;
reg [7:0] rf_filter_coeff113_b;
reg [7:0] rf_filter_coeff114_a;
reg [7:0] rf_filter_coeff114_b;
reg [7:0] rf_filter_coeff115_a;
reg [7:0] rf_filter_coeff115_b;
reg [7:0] rf_filter_coeff116_a;
reg [7:0] rf_filter_coeff116_b;
reg [7:0] rf_filter_coeff117_a;
reg [7:0] rf_filter_coeff117_b;
reg [7:0] rf_filter_coeff118_a;
reg [7:0] rf_filter_coeff118_b;
reg [7:0] rf_filter_coeff119_a;
reg [7:0] rf_filter_coeff119_b;
reg [7:0] rf_filter_coeff120_a;
reg [7:0] rf_filter_coeff120_b;
reg [7:0] rf_filter_coeff121_a;
reg [7:0] rf_filter_coeff121_b;
reg [7:0] rf_filter_coeff122_a;
reg [7:0] rf_filter_coeff122_b;
reg [7:0] rf_filter_coeff123_a;
reg [7:0] rf_filter_coeff123_b;
reg [7:0] rf_filter_coeff124_a;
reg [7:0] rf_filter_coeff124_b;
reg [7:0] rf_filter_coeff125_a;
reg [7:0] rf_filter_coeff125_b;
reg [7:0] rf_filter_coeff126_a;
reg [7:0] rf_filter_coeff126_b;
reg [7:0] rf_filter_coeff127_a;
reg [7:0] rf_filter_coeff127_b;
reg [7:0] rf_filter_coeff128_a;
reg [7:0] rf_filter_coeff128_b;
reg [7:0] rf_filter_coeff129_a;
reg [7:0] rf_filter_coeff129_b;
reg [7:0] rf_filter_coeff130_a;
```

```
reg [7:0] rf_filter_coeff130_b;
reg [7:0] rf_filter_coeff131_a;
reg [7:0] rf_filter_coeff131_b;
reg [7:0] rf_filter_coeff132_a;
reg [7:0] rf_filter_coeff132_b;
reg [7:0] rf_filter_coeff133_a;
reg [7:0] rf_filter_coeff133_b;
reg [7:0] rf_filter_coeff134_a;
reg [7:0] rf_filter_coeff134_b;
reg [7:0] rf_filter_coeff135_a;
reg [7:0] rf_filter_coeff135_b;
reg [7:0] rf_filter_coeff136_a;
reg [7:0] rf_filter_coeff136_b;
reg [7:0] rf_filter_coeff137_a;
reg [7:0] rf_filter_coeff137_b;
reg [7:0] rf_filter_coeff138_a;
reg [7:0] rf_filter_coeff138_b;
reg [7:0] rf_filter_coeff139_a;
reg [7:0] rf_filter_coeff139_b;
reg [7:0] rf_filter_coeff140_a;
reg [7:0] rf_filter_coeff140_b;
reg [7:0] rf_filter_coeff141_a;
reg [7:0] rf_filter_coeff141_b;
reg [7:0] rf_filter_coeff142_a;
reg [7:0] rf_filter_coeff142_b;
reg [7:0] rf_filter_coeff143_a;
reg [7:0] rf_filter_coeff143_b;
reg [7:0] rf_filter_coeff144_a;
reg [7:0] rf_filter_coeff144_b;
reg [7:0] rf_filter_coeff145_a;
reg [7:0] rf_filter_coeff145_b;
reg [7:0] rf_filter_coeff146_a;
reg [7:0] rf_filter_coeff146_b;
reg [7:0] rf_filter_coeff147_a;
reg [7:0] rf_filter_coeff147_b;
reg [7:0] rf_filter_coeff148_a;
reg [7:0] rf_filter_coeff148_b;
reg [7:0] rf_filter_coeff149_a;
reg [7:0] rf_filter_coeff149_b;
reg [7:0] rf_filter_coeff150_a;
reg [7:0] rf_filter_coeff150_b;
reg [7:0] rf_filter_coeff151_a;
reg [7:0] rf_filter_coeff151_b;
reg [7:0] rf_filter_coeff152_a;
reg [7:0] rf_filter_coeff152_b;
reg [7:0] rf_filter_coeff153_a;
reg [7:0] rf_filter_coeff153_b;
reg [7:0] rf_filter_coeff154_a;
reg [7:0] rf_filter_coeff154_b;
reg [7:0] rf_filter_coeff155_a;
reg [7:0] rf_filter_coeff155_b;
reg [7:0] rf_filter_coeff156_a;
reg [7:0] rf_filter_coeff156_b;
reg [7:0] rf_filter_coeff157_a;
reg [7:0] rf_filter_coeff157_b;
reg [7:0] rf_filter_coeff158_a;
reg [7:0] rf_filter_coeff158_b;
reg [7:0] rf_filter_coeff159_a;
reg [7:0] rf_filter_coeff159_b;
reg [7:0] rf_filter_coeff160_a;
reg [7:0] rf_filter_coeff160_b;
reg [7:0] rf_filter_coeff161_a;
reg [7:0] rf_filter_coeff161_b;
reg [7:0] rf_filter_coeff162_a;
reg [7:0] rf_filter_coeff162_b;
reg [7:0] rf_filter_coeff163_a;
reg [7:0] rf_filter_coeff163_b;
reg [7:0] rf_filter_coeff164_a;
reg [7:0] rf_filter_coeff164_b;
reg [7:0] rf_filter_coeff165_a;
reg [7:0] rf_filter_coeff165_b;
```

```
reg [7:0] rf_filter_coeff166_a;
reg [7:0] rf_filter_coeff166_b;
reg [7:0] rf_filter_coeff167_a;
reg [7:0] rf_filter_coeff167_b;
reg [7:0] rf_filter_coeff168_a;
reg [7:0] rf_filter_coeff168_b;
reg [7:0] rf_filter_coeff169_a;
reg [7:0] rf_filter_coeff169_b;
reg [7:0] rf_filter_coeff170_a;
reg [7:0] rf_filter_coeff170_b;
reg [7:0] rf_filter_coeff171_a;
reg [7:0] rf_filter_coeff171_b;
reg [7:0] rf_filter_coeff172_a;
reg [7:0] rf_filter_coeff172_b;
reg [7:0] rf_filter_coeff173_a;
reg [7:0] rf_filter_coeff173_b;
reg [7:0] rf_filter_coeff174_a;
reg [7:0] rf_filter_coeff174_b;
reg [7:0] rf_filter_coeff175_a;
reg [7:0] rf_filter_coeff175_b;
reg [7:0] rf_filter_coeff176_a;
reg [7:0] rf_filter_coeff176_b;
reg [7:0] rf_filter_coeff177_a;
reg [7:0] rf_filter_coeff177_b;
reg [7:0] rf_filter_coeff178_a;
reg [7:0] rf_filter_coeff178_b;
reg [7:0] rf_filter_coeff179_a;
reg [7:0] rf_filter_coeff179_b;
reg [7:0] rf_filter_coeff180_a;
reg [7:0] rf_filter_coeff180_b;
reg [7:0] rf_filter_coeff181_a;
reg [7:0] rf_filter_coeff181_b;
reg [7:0] rf_filter_coeff182_a;
reg [7:0] rf_filter_coeff182_b;
reg [7:0] rf_filter_coeff183_a;
reg [7:0] rf_filter_coeff183_b;
reg [7:0] rf_filter_coeff184_a;
reg [7:0] rf_filter_coeff184_b;
reg [7:0] rf_filter_coeff185_a;
reg [7:0] rf_filter_coeff185_b;
reg [7:0] rf_filter_coeff186_a;
reg [7:0] rf_filter_coeff186_b;
reg [7:0] rf_filter_coeff187_a;
reg [7:0] rf_filter_coeff187_b;
reg [7:0] rf_filter_coeff188_a;
reg [7:0] rf_filter_coeff188_b;
reg [7:0] rf_filter_coeff189_a;
reg [7:0] rf_filter_coeff189_b;
reg [7:0] rf_filter_coeff190_a;
reg [7:0] rf_filter_coeff190_b;
reg [7:0] rf_filter_coeff191_a;
reg [7:0] rf_filter_coeff191_b;
reg [7:0] rf_filter_coeff192_a;
reg [7:0] rf_filter_coeff192_b;
reg [7:0] rf_filter_coeff193_a;
reg [7:0] rf_filter_coeff193_b;
reg [7:0] rf_filter_coeff194_a;
reg [7:0] rf_filter_coeff194_b;
reg [7:0] rf_filter_coeff195_a;
reg [7:0] rf_filter_coeff195_b;
reg [7:0] rf_filter_coeff196_a;
reg [7:0] rf_filter_coeff196_b;
reg [7:0] rf_filter_coeff197_a;
reg [7:0] rf_filter_coeff197_b;
reg [7:0] rf_filter_coeff198_a;
reg [7:0] rf_filter_coeff198_b;
reg [7:0] rf_filter_coeff199_a;
reg [7:0] rf_filter_coeff199_b;
reg [7:0] rf_filter_coeff200_a;
reg [7:0] rf_filter_coeff200_b;
reg [7:0] rf_filter_coeff201_a;
```

```
reg [7:0] rf_filter_coeff201_b;
reg [7:0] rf_filter_coeff202_a;
reg [7:0] rf_filter_coeff202_b;
reg [7:0] rf_filter_coeff203_a;
reg [7:0] rf_filter_coeff203_b;
reg [7:0] rf_filter_coeff204_a;
reg [7:0] rf_filter_coeff204_b;
reg [7:0] rf_filter_coeff205_a;
reg [7:0] rf_filter_coeff205_b;
reg [7:0] rf_filter_coeff206_a;
reg [7:0] rf_filter_coeff206_b;
reg [7:0] rf_filter_coeff207_a;
reg [7:0] rf_filter_coeff207_b;
reg [7:0] rf_filter_coeff208_a;
reg [7:0] rf_filter_coeff208_b;
reg [7:0] rf_filter_coeff209_a;
reg [7:0] rf_filter_coeff209_b;
reg [7:0] rf_filter_coeff210_a;
reg [7:0] rf_filter_coeff210_b;
reg [7:0] rf_filter_coeff211_a;
reg [7:0] rf_filter_coeff211_b;
reg [7:0] rf_filter_coeff212_a;
reg [7:0] rf_filter_coeff212_b;
reg [7:0] rf_filter_coeff213_a;
reg [7:0] rf_filter_coeff213_b;
reg [7:0] rf_filter_coeff214_a;
reg [7:0] rf_filter_coeff214_b;
reg [7:0] rf_filter_coeff215_a;
reg [7:0] rf_filter_coeff215_b;
reg [7:0] rf_filter_coeff216_a;
reg [7:0] rf_filter_coeff216_b;
reg [7:0] rf_filter_coeff217_a;
reg [7:0] rf_filter_coeff217_b;
reg [7:0] rf_filter_coeff218_a;
reg [7:0] rf_filter_coeff218_b;
reg [7:0] rf_filter_coeff219_a;
reg [7:0] rf_filter_coeff219_b;
reg [7:0] rf_filter_coeff220_a;
reg [7:0] rf_filter_coeff220_b;
reg [7:0] rf_filter_coeff221_a;
reg [7:0] rf_filter_coeff221_b;
reg [7:0] rf_filter_coeff222_a;
reg [7:0] rf_filter_coeff222_b;
reg [7:0] rf_filter_coeff223_a;
reg [7:0] rf_filter_coeff223_b;
reg [7:0] rf_filter_coeff224_a;
reg [7:0] rf_filter_coeff224_b;
reg [7:0] rf_filter_coeff225_a;
reg [7:0] rf_filter_coeff225_b;
reg [7:0] rf_filter_coeff226_a;
reg [7:0] rf_filter_coeff226_b;
reg [7:0] rf_filter_coeff227_a;
reg [7:0] rf_filter_coeff227_b;
reg [7:0] rf_filter_coeff228_a;
reg [7:0] rf_filter_coeff228_b;
reg [7:0] rf_filter_coeff229_a;
reg [7:0] rf_filter_coeff229_b;
reg [7:0] rf_filter_coeff230_a;
reg [7:0] rf_filter_coeff230_b;
reg [7:0] rf_filter_coeff231_a;
reg [7:0] rf_filter_coeff231_b;
reg [7:0] rf_filter_coeff232_a;
reg [7:0] rf_filter_coeff232_b;
reg [7:0] rf_filter_coeff233_a;
reg [7:0] rf_filter_coeff233_b;
reg [7:0] rf_filter_coeff234_a;
reg [7:0] rf_filter_coeff234_b;
reg [7:0] rf_filter_coeff235_a;
reg [7:0] rf_filter_coeff235_b;
reg [7:0] rf_filter_coeff236_a;
reg [7:0] rf_filter_coeff236_b;
```

```
reg [7:0] rf_filter_coeff237_a;
reg [7:0] rf_filter_coeff237_b;
reg [7:0] rf_filter_coeff238_a;
reg [7:0] rf_filter_coeff238_b;
reg [7:0] rf_filter_coeff239_a;
reg [7:0] rf_filter_coeff239_b;
reg [7:0] rf_filter_coeff240_a;
reg [7:0] rf_filter_coeff240_b;
reg [7:0] rf_filter_coeff241_a;
reg [7:0] rf_filter_coeff241_b;
reg [7:0] rf_filter_coeff242_a;
reg [7:0] rf_filter_coeff242_b;
reg [7:0] rf_filter_coeff243_a;
reg [7:0] rf_filter_coeff243_b;
reg [7:0] rf_filter_coeff244_a;
reg [7:0] rf_filter_coeff244_b;
reg [7:0] rf_filter_coeff245_a;
reg [7:0] rf_filter_coeff245_b;
reg [7:0] rf_filter_coeff246_a;
reg [7:0] rf_filter_coeff246_b;
reg [7:0] rf_filter_coeff247_a;
reg [7:0] rf_filter_coeff247_b;
reg [7:0] rf_filter_coeff248_a;
reg [7:0] rf_filter_coeff248_b;
reg [7:0] rf_filter_coeff249_a;
reg [7:0] rf_filter_coeff249_b;
reg [7:0] rf_filter_coeff250_a;
reg [7:0] rf_filter_coeff250_b;
reg [7:0] rf_filter_coeff251_a;
reg [7:0] rf_filter_coeff251_b;
reg [7:0] rf_filter_coeff252_a;
reg [7:0] rf_filter_coeff252_b;
reg [7:0] rf_filter_coeff253_a;
reg [7:0] rf_filter_coeff253_b;
reg [7:0] rf_filter_coeff254_a;
reg [7:0] rf_filter_coeff254_b;
reg [7:0] rf_filter_coeff255_a;
reg [7:0] rf_filter_coeff255_b;
reg [7:0] rf_filter_coeff256_a;
reg [7:0] rf_filter_coeff256_b;
reg [7:0] rf_filter_coeff257_a;
reg [7:0] rf_filter_coeff257_b;
reg [7:0] rf_filter_coeff258_a;
reg [7:0] rf_filter_coeff258_b;
reg [7:0] rf_filter_coeff259_a;
reg [7:0] rf_filter_coeff259_b;
reg [7:0] rf_filter_coeff260_a;
reg [7:0] rf_filter_coeff260_b;
reg [7:0] rf_filter_coeff261_a;
reg [7:0] rf_filter_coeff261_b;
reg [7:0] rf_filter_coeff262_a;
reg [7:0] rf_filter_coeff262_b;
reg [7:0] rf_filter_coeff263_a;
reg [7:0] rf_filter_coeff263_b;
reg [7:0] rf_filter_coeff264_a;
reg [7:0] rf_filter_coeff264_b;
reg [7:0] rf_filter_coeff265_a;
reg [7:0] rf_filter_coeff265_b;
reg [7:0] rf_filter_coeff266_a;
reg [7:0] rf_filter_coeff266_b;
reg [7:0] rf_filter_coeff267_a;
reg [7:0] rf_filter_coeff267_b;
reg [7:0] rf_filter_coeff268_a;
reg [7:0] rf_filter_coeff268_b;
reg [7:0] rf_filter_coeff269_a;
reg [7:0] rf_filter_coeff269_b;
reg [7:0] rf_filter_coeff270_a;
reg [7:0] rf_filter_coeff270_b;
reg [7:0] rf_filter_coeff271_a;
reg [7:0] rf_filter_coeff271_b;
reg [7:0] rf_filter_coeff272_a;
```

```
reg [7:0] rf_filter_coeff272_b;
reg [7:0] rf_filter_coeff273_a;
reg [7:0] rf_filter_coeff273_b;
reg [7:0] rf_filter_coeff274_a;
reg [7:0] rf_filter_coeff274_b;
reg [7:0] rf_filter_coeff275_a;
reg [7:0] rf_filter_coeff275_b;
reg [7:0] rf_filter_coeff276_a;
reg [7:0] rf_filter_coeff276_b;
reg [7:0] rf_filter_coeff277_a;
reg [7:0] rf_filter_coeff277_b;
reg [7:0] rf_filter_coeff278_a;
reg [7:0] rf_filter_coeff278_b;
reg [7:0] rf_filter_coeff279_a;
reg [7:0] rf_filter_coeff279_b;
reg [7:0] rf_filter_coeff280_a;
reg [7:0] rf_filter_coeff280_b;
reg [7:0] rf_filter_coeff281_a;
reg [7:0] rf_filter_coeff281_b;
reg [7:0] rf_filter_coeff282_a;
reg [7:0] rf_filter_coeff282_b;
reg [7:0] rf_filter_coeff283_a;
reg [7:0] rf_filter_coeff283_b;
reg [7:0] rf_filter_coeff284_a;
reg [7:0] rf_filter_coeff284_b;
reg [7:0] rf_filter_coeff285_a;
reg [7:0] rf_filter_coeff285_b;
reg [7:0] rf_filter_coeff286_a;
reg [7:0] rf_filter_coeff286_b;
reg [7:0] rf_filter_coeff287_a;
reg [7:0] rf_filter_coeff287_b;
reg [7:0] rf_filter_coeff288_a;
reg [7:0] rf_filter_coeff288_b;
reg [7:0] rf_filter_coeff289_a;
reg [7:0] rf_filter_coeff289_b;
reg [7:0] rf_filter_coeff290_a;
reg [7:0] rf_filter_coeff290_b;
reg [7:0] rf_filter_coeff291_a;
reg [7:0] rf_filter_coeff291_b;
reg [7:0] rf_filter_coeff292_a;
reg [7:0] rf_filter_coeff292_b;
reg [7:0] rf_filter_coeff293_a;
reg [7:0] rf_filter_coeff293_b;
reg [7:0] rf_filter_coeff294_a;
reg [7:0] rf_filter_coeff294_b;
reg [7:0] rf_filter_coeff295_a;
reg [7:0] rf_filter_coeff295_b;
reg [7:0] rf_filter_coeff296_a;
reg [7:0] rf_filter_coeff296_b;
reg [7:0] rf_filter_coeff297_a;
reg [7:0] rf_filter_coeff297_b;
reg [7:0] rf_filter_coeff298_a;
reg [7:0] rf_filter_coeff298_b;
reg [7:0] rf_filter_coeff299_a;
reg [7:0] rf_filter_coeff299_b;
reg [7:0] rf_filter_coeff300_a;
reg [7:0] rf_filter_coeff300_b;
reg [7:0] rf_filter_coeff301_a;
reg [7:0] rf_filter_coeff301_b;
reg [7:0] rf_filter_coeff302_a;
reg [7:0] rf_filter_coeff302_b;
reg [7:0] rf_filter_coeff303_a;
reg [7:0] rf_filter_coeff303_b;
reg [7:0] rf_filter_coeff304_a;
reg [7:0] rf_filter_coeff304_b;
reg [7:0] rf_filter_coeff305_a;
reg [7:0] rf_filter_coeff305_b;
reg [7:0] rf_filter_coeff306_a;
reg [7:0] rf_filter_coeff306_b;
reg [7:0] rf_filter_coeff307_a;
reg [7:0] rf_filter_coeff307_b;
```

```
reg [7:0] rf_filter_coeff308_a;
reg [7:0] rf_filter_coeff308_b;
reg [7:0] rf_filter_coeff309_a;
reg [7:0] rf_filter_coeff309_b;
reg [7:0] rf_filter_coeff310_a;
reg [7:0] rf_filter_coeff310_b;
reg [7:0] rf_filter_coeff311_a;
reg [7:0] rf_filter_coeff311_b;
reg [7:0] rf_filter_coeff312_a;
reg [7:0] rf_filter_coeff312_b;
reg [7:0] rf_filter_coeff313_a;
reg [7:0] rf_filter_coeff313_b;
reg [7:0] rf_filter_coeff314_a;
reg [7:0] rf_filter_coeff314_b;
reg [7:0] rf_filter_coeff315_a;
reg [7:0] rf_filter_coeff315_b;
reg [7:0] rf_filter_coeff316_a;
reg [7:0] rf_filter_coeff316_b;
reg [7:0] rf_filter_coeff317_a;
reg [7:0] rf_filter_coeff317_b;
reg [7:0] rf_filter_coeff318_a;
reg [7:0] rf_filter_coeff318_b;
reg [7:0] rf_filter_coeff319_a;
reg [7:0] rf_filter_coeff319_b;
reg [7:0] rf_filter_coeff320_a;
reg [7:0] rf_filter_coeff320_b;
reg [7:0] rf_filter_coeff321_a;
reg [7:0] rf_filter_coeff321_b;
reg [7:0] rf_filter_coeff322_a;
reg [7:0] rf_filter_coeff322_b;
reg [7:0] rf_filter_coeff323_a;
reg [7:0] rf_filter_coeff323_b;
reg [7:0] rf_filter_coeff324_a;
reg [7:0] rf_filter_coeff324_b;
reg [7:0] rf_filter_coeff325_a;
reg [7:0] rf_filter_coeff325_b;
reg [7:0] rf_filter_coeff326_a;
reg [7:0] rf_filter_coeff326_b;
reg [7:0] rf_filter_coeff327_a;
reg [7:0] rf_filter_coeff327_b;
reg [7:0] rf_filter_coeff328_a;
reg [7:0] rf_filter_coeff328_b;
reg [7:0] rf_filter_coeff329_a;
reg [7:0] rf_filter_coeff329_b;
reg [7:0] rf_filter_coeff330_a;
reg [7:0] rf_filter_coeff330_b;
reg [7:0] rf_filter_coeff331_a;
reg [7:0] rf_filter_coeff331_b;
reg [7:0] rf_filter_coeff332_a;
reg [7:0] rf_filter_coeff332_b;
reg [7:0] rf_filter_coeff333_a;
reg [7:0] rf_filter_coeff333_b;
reg [7:0] rf_filter_coeff334_a;
reg [7:0] rf_filter_coeff334_b;
reg [7:0] rf_filter_coeff335_a;
reg [7:0] rf_filter_coeff335_b;
reg [7:0] rf_filter_coeff336_a;
reg [7:0] rf_filter_coeff336_b;
reg [7:0] rf_filter_coeff337_a;
reg [7:0] rf_filter_coeff337_b;
reg [7:0] rf_filter_coeff338_a;
reg [7:0] rf_filter_coeff338_b;
reg [7:0] rf_filter_coeff339_a;
reg [7:0] rf_filter_coeff339_b;
reg [7:0] rf_filter_coeff340_a;
reg [7:0] rf_filter_coeff340_b;
reg [7:0] rf_filter_coeff341_a;
reg [7:0] rf_filter_coeff341_b;
reg [7:0] rf_filter_coeff342_a;
reg [7:0] rf_filter_coeff342_b;
reg [7:0] rf_filter_coeff343_a;
```

```
reg [7:0] rf_filter_coeff343_b;
reg [7:0] rf_filter_coeff344_a;
reg [7:0] rf_filter_coeff344_b;
reg [7:0] rf_filter_coeff345_a;
reg [7:0] rf_filter_coeff345_b;
reg [7:0] rf_filter_coeff346_a;
reg [7:0] rf_filter_coeff346_b;
reg [7:0] rf_filter_coeff347_a;
reg [7:0] rf_filter_coeff347_b;
reg [7:0] rf_filter_coeff348_a;
reg [7:0] rf_filter_coeff348_b;
reg [7:0] rf_filter_coeff349_a;
reg [7:0] rf_filter_coeff349_b;
reg [7:0] rf_filter_coeff350_a;
reg [7:0] rf_filter_coeff350_b;
reg [7:0] rf_filter_coeff351_a;
reg [7:0] rf_filter_coeff351_b;
reg [7:0] rf_filter_coeff352_a;
reg [7:0] rf_filter_coeff352_b;
reg [7:0] rf_filter_coeff353_a;
reg [7:0] rf_filter_coeff353_b;
reg [7:0] rf_filter_coeff354_a;
reg [7:0] rf_filter_coeff354_b;
reg [7:0] rf_filter_coeff355_a;
reg [7:0] rf_filter_coeff355_b;
reg [7:0] rf_filter_coeff356_a;
reg [7:0] rf_filter_coeff356_b;
reg [7:0] rf_filter_coeff357_a;
reg [7:0] rf_filter_coeff357_b;
reg [7:0] rf_filter_coeff358_a;
reg [7:0] rf_filter_coeff358_b;
reg [7:0] rf_filter_coeff359_a;
reg [7:0] rf_filter_coeff359_b;
reg [7:0] rf_filter_coeff360_a;
reg [7:0] rf_filter_coeff360_b;
reg [7:0] rf_filter_coeff361_a;
reg [7:0] rf_filter_coeff361_b;
reg [7:0] rf_filter_coeff362_a;
reg [7:0] rf_filter_coeff362_b;
reg [7:0] rf_filter_coeff363_a;
reg [7:0] rf_filter_coeff363_b;
reg [7:0] rf_filter_coeff364_a;
reg [7:0] rf_filter_coeff364_b;
reg [7:0] rf_filter_coeff365_a;
reg [7:0] rf_filter_coeff365_b;
reg [7:0] rf_filter_coeff366_a;
reg [7:0] rf_filter_coeff366_b;
reg [7:0] rf_filter_coeff367_a;
reg [7:0] rf_filter_coeff367_b;
reg [7:0] rf_filter_coeff368_a;
reg [7:0] rf_filter_coeff368_b;
reg [7:0] rf_filter_coeff369_a;
reg [7:0] rf_filter_coeff369_b;
reg [7:0] rf_filter_coeff370_a;
reg [7:0] rf_filter_coeff370_b;
reg [7:0] rf_filter_coeff371_a;
reg [7:0] rf_filter_coeff371_b;
reg [7:0] rf_filter_coeff372_a;
reg [7:0] rf_filter_coeff372_b;
reg [7:0] rf_filter_coeff373_a;
reg [7:0] rf_filter_coeff373_b;
reg [7:0] rf_filter_coeff374_a;
reg [7:0] rf_filter_coeff374_b;
reg [7:0] rf_filter_coeff375_a;
reg [7:0] rf_filter_coeff375_b;
reg [7:0] rf_filter_coeff376_a;
reg [7:0] rf_filter_coeff376_b;
reg [7:0] rf_filter_coeff377_a;
reg [7:0] rf_filter_coeff377_b;
reg [7:0] rf_filter_coeff378_a;
reg [7:0] rf_filter_coeff378_b;
```

```
reg [7:0] rf_filter_coeff379_a;
reg [7:0] rf_filter_coeff379_b;
reg [7:0] rf_filter_coeff380_a;
reg [7:0] rf_filter_coeff380_b;
reg [7:0] rf_filter_coeff381_a;
reg [7:0] rf_filter_coeff381_b;
reg [7:0] rf_filter_coeff382_a;
reg [7:0] rf_filter_coeff382_b;
reg [7:0] rf_filter_coeff383_a;
reg [7:0] rf_filter_coeff383_b;
reg [7:0] rf_filter_coeff384_a;
reg [7:0] rf_filter_coeff384_b;
reg [7:0] rf_filter_coeff385_a;
reg [7:0] rf_filter_coeff385_b;
reg [7:0] rf_filter_coeff386_a;
reg [7:0] rf_filter_coeff386_b;
reg [7:0] rf_filter_coeff387_a;
reg [7:0] rf_filter_coeff387_b;
reg [7:0] rf_filter_coeff388_a;
reg [7:0] rf_filter_coeff388_b;
reg [7:0] rf_filter_coeff389_a;
reg [7:0] rf_filter_coeff389_b;
reg [7:0] rf_filter_coeff390_a;
reg [7:0] rf_filter_coeff390_b;
reg [7:0] rf_filter_coeff391_a;
reg [7:0] rf_filter_coeff391_b;
reg [7:0] rf_filter_coeff392_a;
reg [7:0] rf_filter_coeff392_b;
reg [7:0] rf_filter_coeff393_a;
reg [7:0] rf_filter_coeff393_b;
reg [7:0] rf_filter_coeff394_a;
reg [7:0] rf_filter_coeff394_b;
reg [7:0] rf_filter_coeff395_a;
reg [7:0] rf_filter_coeff395_b;
reg [7:0] rf_filter_coeff396_a;
reg [7:0] rf_filter_coeff396_b;
reg [7:0] rf_filter_coeff397_a;
reg [7:0] rf_filter_coeff397_b;
reg [7:0] rf_filter_coeff398_a;
reg [7:0] rf_filter_coeff398_b;
reg [7:0] rf_filter_coeff399_a;
reg [7:0] rf_filter_coeff399_b;
reg [7:0] rf_filter_coeff400_a;
reg [7:0] rf_filter_coeff400_b;
reg [7:0] rf_filter_coeff401_a;
reg [7:0] rf_filter_coeff401_b;
reg [7:0] rf_filter_coeff402_a;
reg [7:0] rf_filter_coeff402_b;
reg [7:0] rf_filter_coeff403_a;
reg [7:0] rf_filter_coeff403_b;
reg [7:0] rf_filter_coeff404_a;
reg [7:0] rf_filter_coeff404_b;
reg [7:0] rf_filter_coeff405_a;
reg [7:0] rf_filter_coeff405_b;
reg [7:0] rf_filter_coeff406_a;
reg [7:0] rf_filter_coeff406_b;
reg [7:0] rf_filter_coeff407_a;
reg [7:0] rf_filter_coeff407_b;
reg [7:0] rf_filter_coeff408_a;
reg [7:0] rf_filter_coeff408_b;
reg [7:0] rf_filter_coeff409_a;
reg [7:0] rf_filter_coeff409_b;
reg [7:0] rf_filter_coeff410_a;
reg [7:0] rf_filter_coeff410_b;
reg [7:0] rf_filter_coeff411_a;
reg [7:0] rf_filter_coeff411_b;
reg [7:0] rf_filter_coeff412_a;
reg [7:0] rf_filter_coeff412_b;
reg [7:0] rf_filter_coeff413_a;
reg [7:0] rf_filter_coeff413_b;
reg [7:0] rf_filter_coeff414_a;
```

```
reg [7:0] rf_filter_coeff414_b;
reg [7:0] rf_filter_coeff415_a;
reg [7:0] rf_filter_coeff415_b;
reg [7:0] rf_filter_coeff416_a;
reg [7:0] rf_filter_coeff416_b;
reg [7:0] rf_filter_coeff417_a;
reg [7:0] rf_filter_coeff417_b;
reg [7:0] rf_filter_coeff418_a;
reg [7:0] rf_filter_coeff418_b;
reg [7:0] rf_filter_coeff419_a;
reg [7:0] rf_filter_coeff419_b;
reg [7:0] rf_filter_coeff420_a;
reg [7:0] rf_filter_coeff420_b;
reg [7:0] rf_filter_coeff421_a;
reg [7:0] rf_filter_coeff421_b;
reg [7:0] rf_filter_coeff422_a;
reg [7:0] rf_filter_coeff422_b;
reg [7:0] rf_filter_coeff423_a;
reg [7:0] rf_filter_coeff423_b;
reg [7:0] rf_filter_coeff424_a;
reg [7:0] rf_filter_coeff424_b;
reg [7:0] rf_filter_coeff425_a;
reg [7:0] rf_filter_coeff425_b;
reg [7:0] rf_filter_coeff426_a;
reg [7:0] rf_filter_coeff426_b;
reg [7:0] rf_filter_coeff427_a;
reg [7:0] rf_filter_coeff427_b;
reg [7:0] rf_filter_coeff428_a;
reg [7:0] rf_filter_coeff428_b;
reg [7:0] rf_filter_coeff429_a;
reg [7:0] rf_filter_coeff429_b;
reg [7:0] rf_filter_coeff430_a;
reg [7:0] rf_filter_coeff430_b;
reg [7:0] rf_filter_coeff431_a;
reg [7:0] rf_filter_coeff431_b;
reg [7:0] rf_filter_coeff432_a;
reg [7:0] rf_filter_coeff432_b;
reg [7:0] rf_filter_coeff433_a;
reg [7:0] rf_filter_coeff433_b;
reg [7:0] rf_filter_coeff434_a;
reg [7:0] rf_filter_coeff434_b;
reg [7:0] rf_filter_coeff435_a;
reg [7:0] rf_filter_coeff435_b;
reg [7:0] rf_filter_coeff436_a;
reg [7:0] rf_filter_coeff436_b;
reg [7:0] rf_filter_coeff437_a;
reg [7:0] rf_filter_coeff437_b;
reg [7:0] rf_filter_coeff438_a;
reg [7:0] rf_filter_coeff438_b;
reg [7:0] rf_filter_coeff439_a;
reg [7:0] rf_filter_coeff439_b;
reg [7:0] rf_filter_coeff440_a;
reg [7:0] rf_filter_coeff440_b;
reg [7:0] rf_filter_coeff441_a;
reg [7:0] rf_filter_coeff441_b;
reg [7:0] rf_filter_coeff442_a;
reg [7:0] rf_filter_coeff442_b;
reg [7:0] rf_filter_coeff443_a;
reg [7:0] rf_filter_coeff443_b;
reg [7:0] rf_filter_coeff444_a;
reg [7:0] rf_filter_coeff444_b;
reg [7:0] rf_filter_coeff445_a;
reg [7:0] rf_filter_coeff445_b;
reg [7:0] rf_filter_coeff446_a;
reg [7:0] rf_filter_coeff446_b;
reg [7:0] rf_filter_coeff447_a;
reg [7:0] rf_filter_coeff447_b;
reg [7:0] rf_filter_coeff448_a;
reg [7:0] rf_filter_coeff448_b;
reg [7:0] rf_filter_coeff449_a;
reg [7:0] rf_filter_coeff449_b;
```

```
reg [7:0] rf_filter_coeff450_a;
reg [7:0] rf_filter_coeff450_b;
reg [7:0] rf_filter_coeff451_a;
reg [7:0] rf_filter_coeff451_b;
reg [7:0] rf_filter_coeff452_a;
reg [7:0] rf_filter_coeff452_b;
reg [7:0] rf_filter_coeff453_a;
reg [7:0] rf_filter_coeff453_b;
reg [7:0] rf_filter_coeff454_a;
reg [7:0] rf_filter_coeff454_b;
reg [7:0] rf_filter_coeff455_a;
reg [7:0] rf_filter_coeff455_b;
reg [7:0] rf_filter_coeff456_a;
reg [7:0] rf_filter_coeff456_b;
reg [7:0] rf_filter_coeff457_a;
reg [7:0] rf_filter_coeff457_b;
reg [7:0] rf_filter_coeff458_a;
reg [7:0] rf_filter_coeff458_b;
reg [7:0] rf_filter_coeff459_a;
reg [7:0] rf_filter_coeff459_b;
reg [7:0] rf_filter_coeff460_a;
reg [7:0] rf_filter_coeff460_b;
reg [7:0] rf_filter_coeff461_a;
reg [7:0] rf_filter_coeff461_b;
reg [7:0] rf_filter_coeff462_a;
reg [7:0] rf_filter_coeff462_b;
reg [7:0] rf_filter_coeff463_a;
reg [7:0] rf_filter_coeff463_b;
reg [7:0] rf_filter_coeff464_a;
reg [7:0] rf_filter_coeff464_b;
reg [7:0] rf_filter_coeff465_a;
reg [7:0] rf_filter_coeff465_b;
reg [7:0] rf_filter_coeff466_a;
reg [7:0] rf_filter_coeff466_b;
reg [7:0] rf_filter_coeff467_a;
reg [7:0] rf_filter_coeff467_b;
reg [7:0] rf_filter_coeff468_a;
reg [7:0] rf_filter_coeff468_b;
reg [7:0] rf_filter_coeff469_a;
reg [7:0] rf_filter_coeff469_b;
reg [7:0] rf_filter_coeff470_a;
reg [7:0] rf_filter_coeff470_b;
reg [7:0] rf_filter_coeff471_a;
reg [7:0] rf_filter_coeff471_b;
reg [7:0] rf_filter_coeff472_a;
reg [7:0] rf_filter_coeff472_b;
reg [7:0] rf_filter_coeff473_a;
reg [7:0] rf_filter_coeff473_b;
reg [7:0] rf_filter_coeff474_a;
reg [7:0] rf_filter_coeff474_b;
reg [7:0] rf_filter_coeff475_a;
reg [7:0] rf_filter_coeff475_b;
reg [7:0] rf_filter_coeff476_a;
reg [7:0] rf_filter_coeff476_b;
reg [7:0] rf_filter_coeff477_a;
reg [7:0] rf_filter_coeff477_b;
reg [7:0] rf_filter_coeff478_a;
reg [7:0] rf_filter_coeff478_b;
reg [7:0] rf_filter_coeff479_a;
reg [7:0] rf_filter_coeff479_b;
reg [7:0] rf_filter_coeff480_a;
reg [7:0] rf_filter_coeff480_b;
reg [7:0] rf_filter_coeff481_a;
reg [7:0] rf_filter_coeff481_b;
reg [7:0] rf_filter_coeff482_a;
reg [7:0] rf_filter_coeff482_b;
reg [7:0] rf_filter_coeff483_a;
reg [7:0] rf_filter_coeff483_b;
reg [7:0] rf_filter_coeff484_a;
reg [7:0] rf_filter_coeff484_b;
reg [7:0] rf_filter_coeff485_a;
```

```
reg [7:0] rf_filter_coeff485_b;
reg [7:0] rf_filter_coeff486_a;
reg [7:0] rf_filter_coeff486_b;
reg [7:0] rf_filter_coeff487_a;
reg [7:0] rf_filter_coeff487_b;
reg [7:0] rf_filter_coeff488_a;
reg [7:0] rf_filter_coeff488_b;
reg [7:0] rf_filter_coeff489_a;
reg [7:0] rf_filter_coeff489_b;
reg [7:0] rf_filter_coeff490_a;
reg [7:0] rf_filter_coeff490_b;
reg [7:0] rf_filter_coeff491_a;
reg [7:0] rf_filter_coeff491_b;
reg [7:0] rf_filter_coeff492_a;
reg [7:0] rf_filter_coeff492_b;
reg [7:0] rf_filter_coeff493_a;
reg [7:0] rf_filter_coeff493_b;
reg [7:0] rf_filter_coeff494_a;
reg [7:0] rf_filter_coeff494_b;
reg [7:0] rf_filter_coeff495_a;
reg [7:0] rf_filter_coeff495_b;
reg [7:0] rf_filter_coeff496_a;
reg [7:0] rf_filter_coeff496_b;
reg [7:0] rf_filter_coeff497_a;
reg [7:0] rf_filter_coeff497_b;
reg [7:0] rf_filter_coeff498_a;
reg [7:0] rf_filter_coeff498_b;
reg [7:0] rf_filter_coeff499_a;
reg [7:0] rf_filter_coeff499_b;
reg [7:0] rf_filter_coeff500_a;
reg [7:0] rf_filter_coeff500_b;
reg [7:0] rf_filter_coeff501_a;
reg [7:0] rf_filter_coeff501_b;
reg [7:0] rf_filter_coeff502_a;
reg [7:0] rf_filter_coeff502_b;
reg [7:0] rf_filter_coeff503_a;
reg [7:0] rf_filter_coeff503_b;
reg [7:0] rf_filter_coeff504_a;
reg [7:0] rf_filter_coeff504_b;
reg [7:0] rf_filter_coeff505_a;
reg [7:0] rf_filter_coeff505_b;
reg [7:0] rf_filter_coeff506_a;
reg [7:0] rf_filter_coeff506_b;
reg [7:0] rf_filter_coeff507_a;
reg [7:0] rf_filter_coeff507_b;
reg [7:0] rf_filter_coeff508_a;
reg [7:0] rf_filter_coeff508_b;
reg [7:0] rf_filter_coeff509_a;
reg [7:0] rf_filter_coeff509_b;
reg [7:0] rf_filter_coeff510_a;
reg [7:0] rf_filter_coeff510_b;
reg [7:0] rf_filter_coeff511_a;
reg [7:0] rf_filter_coeff511_b;

// Outputs
wire aud_in_rtr;
wire [31:0] aud_out;
wire aud_out_rts;

// Instantiate the Unit Under Test (UUT)
filter uut (
          .clk(clk),
          .rst_n(rst_n),
          .aud_in(aud_in),
          .aud_in_rts(aud_in_rts),
          .aud_in_rtr(aud_in_rtr),
          .aud_out(aud_out),
          .aud_out_rts(aud_out_rts),
          .aud_out_rtr(aud_out_rtr),
          .rf_filter_shift(rf_filter_shift),
          .rf_filter_clip_en(rf_filter_clip_en),
```

```
.rf_filter_coeff0_a(rf_filter_coeff0_a),
.rf_filter_coeff0_b(rf_filter_coeff0_b),
.rf_filter_coeff1_a(rf_filter_coeff1_a),
.rf_filter_coeff1_b(rf_filter_coeff1_b),
.rf_filter_coeff2_a(rf_filter_coeff2_a),
.rf_filter_coeff2_b(rf_filter_coeff2_b),
.rf_filter_coeff3_a(rf_filter_coeff3_a),
.rf_filter_coeff3_b(rf_filter_coeff3_b),
.rf_filter_coeff4_a(rf_filter_coeff4_a),
.rf_filter_coeff4_b(rf_filter_coeff4_b),
.rf_filter_coeff5_a(rf_filter_coeff5_a),
.rf_filter_coeff5_b(rf_filter_coeff5_b),
.rf_filter_coeff6_a(rf_filter_coeff6_a),
.rf_filter_coeff6_b(rf_filter_coeff6_b),
.rf_filter_coeff7_a(rf_filter_coeff7_a),
.rf_filter_coeff7_b(rf_filter_coeff7_b),
.rf_filter_coeff8_a(rf_filter_coeff8_a),
.rf_filter_coeff8_b(rf_filter_coeff8_b),
.rf_filter_coeff9_a(rf_filter_coeff9_a),
.rf_filter_coeff9_b(rf_filter_coeff9_b),
.rf_filter_coeff10_a(rf_filter_coeff10_a),
.rf_filter_coeff10_b(rf_filter_coeff10_b),
.rf_filter_coeff11_a(rf_filter_coeff11_a),
.rf_filter_coeff11_b(rf_filter_coeff11_b),
.rf_filter_coeff12_a(rf_filter_coeff12_a),
.rf_filter_coeff12_b(rf_filter_coeff12_b),
.rf_filter_coeff13_a(rf_filter_coeff13_a),
.rf_filter_coeff13_b(rf_filter_coeff13_b),
.rf_filter_coeff14_a(rf_filter_coeff14_a),
.rf_filter_coeff14_b(rf_filter_coeff14_b),
.rf_filter_coeff15_a(rf_filter_coeff15_a),
.rf_filter_coeff15_b(rf_filter_coeff15_b),
.rf_filter_coeff16_a(rf_filter_coeff16_a),
.rf_filter_coeff16_b(rf_filter_coeff16_b),
.rf_filter_coeff17_a(rf_filter_coeff17_a),
.rf_filter_coeff17_b(rf_filter_coeff17_b),
.rf_filter_coeff18_a(rf_filter_coeff18_a),
.rf_filter_coeff18_b(rf_filter_coeff18_b),
.rf_filter_coeff19_a(rf_filter_coeff19_a),
.rf_filter_coeff19_b(rf_filter_coeff19_b),
.rf_filter_coeff20_a(rf_filter_coeff20_a),
.rf_filter_coeff20_b(rf_filter_coeff20_b),
.rf_filter_coeff21_a(rf_filter_coeff21_a),
.rf_filter_coeff21_b(rf_filter_coeff21_b),
.rf_filter_coeff22_a(rf_filter_coeff22_a),
.rf_filter_coeff22_b(rf_filter_coeff22_b),
.rf_filter_coeff23_a(rf_filter_coeff23_a),
.rf_filter_coeff23_b(rf_filter_coeff23_b),
.rf_filter_coeff24_a(rf_filter_coeff24_a),
.rf_filter_coeff24_b(rf_filter_coeff24_b),
.rf_filter_coeff25_a(rf_filter_coeff25_a),
.rf_filter_coeff25_b(rf_filter_coeff25_b),
.rf_filter_coeff26_a(rf_filter_coeff26_a),
.rf_filter_coeff26_b(rf_filter_coeff26_b),
.rf_filter_coeff27_a(rf_filter_coeff27_a),
.rf_filter_coeff27_b(rf_filter_coeff27_b),
.rf_filter_coeff28_a(rf_filter_coeff28_a),
.rf_filter_coeff28_b(rf_filter_coeff28_b),
.rf_filter_coeff29_a(rf_filter_coeff29_a),
.rf_filter_coeff29_b(rf_filter_coeff29_b),
.rf_filter_coeff30_a(rf_filter_coeff30_a),
.rf_filter_coeff30_b(rf_filter_coeff30_b),
.rf_filter_coeff31_a(rf_filter_coeff31_a),
.rf_filter_coeff31_b(rf_filter_coeff31_b),
.rf_filter_coeff32_a(rf_filter_coeff32_a),
.rf_filter_coeff32_b(rf_filter_coeff32_b),
.rf_filter_coeff33_a(rf_filter_coeff33_a),
.rf_filter_coeff33_b(rf_filter_coeff33_b),
.rf_filter_coeff34_a(rf_filter_coeff34_a),
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
```

```
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
```

```
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
```

```
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
```

```
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
```

```
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
```

```
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
```

```
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
```

```
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
```

```
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
```

```
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
```

```
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
```

```
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
```

```
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
.rf_filter_coeff479_a(rf_filter_coeff479_a),
.rf_filter_coeff479_b(rf_filter_coeff479_b),
.rf_filter_coeff480_a(rf_filter_coeff480_a),
.rf_filter_coeff480_b(rf_filter_coeff480_b),
.rf_filter_coeff481_a(rf_filter_coeff481_a),
.rf_filter_coeff481_b(rf_filter_coeff481_b),
.rf_filter_coeff482_a(rf_filter_coeff482_a),
.rf_filter_coeff482_b(rf_filter_coeff482_b),
.rf_filter_coeff483_a(rf_filter_coeff483_a),
.rf_filter_coeff483_b(rf_filter_coeff483_b),
.rf_filter_coeff484_a(rf_filter_coeff484_a),
.rf_filter_coeff484_b(rf_filter_coeff484_b),
.rf_filter_coeff485_a(rf_filter_coeff485_a),
.rf_filter_coeff485_b(rf_filter_coeff485_b),
.rf_filter_coeff486_a(rf_filter_coeff486_a),
.rf_filter_coeff486_b(rf_filter_coeff486_b),
.rf_filter_coeff487_a(rf_filter_coeff487_a),
.rf_filter_coeff487_b(rf_filter_coeff487_b),
.rf_filter_coeff488_a(rf_filter_coeff488_a),
.rf_filter_coeff488_b(rf_filter_coeff488_b),
.rf_filter_coeff489_a(rf_filter_coeff489_a),
.rf_filter_coeff489_b(rf_filter_coeff489_b),
.rf_filter_coeff490_a(rf_filter_coeff490_a),
.rf_filter_coeff490_b(rf_filter_coeff490_b),
.rf_filter_coeff491_a(rf_filter_coeff491_a),
.rf_filter_coeff491_b(rf_filter_coeff491_b),
.rf_filter_coeff492_a(rf_filter_coeff492_a),
.rf_filter_coeff492_b(rf_filter_coeff492_b),
.rf_filter_coeff493_a(rf_filter_coeff493_a),
.rf_filter_coeff493_b(rf_filter_coeff493_b),
.rf_filter_coeff494_a(rf_filter_coeff494_a),
.rf_filter_coeff494_b(rf_filter_coeff494_b),
.rf_filter_coeff495_a(rf_filter_coeff495_a),
.rf_filter_coeff495_b(rf_filter_coeff495_b),
.rf_filter_coeff496_a(rf_filter_coeff496_a),
.rf_filter_coeff496_b(rf_filter_coeff496_b),
```

```
        .rf_filter_coeff497_a(rf_filter_coeff497_a),
        .rf_filter_coeff497_b(rf_filter_coeff497_b),
        .rf_filter_coeff498_a(rf_filter_coeff498_a),
        .rf_filter_coeff498_b(rf_filter_coeff498_b),
        .rf_filter_coeff499_a(rf_filter_coeff499_a),
        .rf_filter_coeff499_b(rf_filter_coeff499_b),
        .rf_filter_coeff500_a(rf_filter_coeff500_a),
        .rf_filter_coeff500_b(rf_filter_coeff500_b),
        .rf_filter_coeff501_a(rf_filter_coeff501_a),
        .rf_filter_coeff501_b(rf_filter_coeff501_b),
        .rf_filter_coeff502_a(rf_filter_coeff502_a),
        .rf_filter_coeff502_b(rf_filter_coeff502_b),
        .rf_filter_coeff503_a(rf_filter_coeff503_a),
        .rf_filter_coeff503_b(rf_filter_coeff503_b),
        .rf_filter_coeff504_a(rf_filter_coeff504_a),
        .rf_filter_coeff504_b(rf_filter_coeff504_b),
        .rf_filter_coeff505_a(rf_filter_coeff505_a),
        .rf_filter_coeff505_b(rf_filter_coeff505_b),
        .rf_filter_coeff506_a(rf_filter_coeff506_a),
        .rf_filter_coeff506_b(rf_filter_coeff506_b),
        .rf_filter_coeff507_a(rf_filter_coeff507_a),
        .rf_filter_coeff507_b(rf_filter_coeff507_b),
        .rf_filter_coeff508_a(rf_filter_coeff508_a),
        .rf_filter_coeff508_b(rf_filter_coeff508_b),
        .rf_filter_coeff509_a(rf_filter_coeff509_a),
        .rf_filter_coeff509_b(rf_filter_coeff509_b),
        .rf_filter_coeff510_a(rf_filter_coeff510_a),
        .rf_filter_coeff510_b(rf_filter_coeff510_b),
        .rf_filter_coeff511_a(rf_filter_coeff511_a),
        .rf_filter_coeff511_b(rf_filter_coeff511_b)
);

initial begin
        // Initialize Inputs
        clk = 0;
        rst_n = 0;
        aud_in = 0;
        aud_in_rts = 0;
        aud_out_rtr = 0;
        rf_filter_coeff0_a = 8'h01;
        rf_filter_coeff0_b = 8'h00;
        rf_filter_coeff1_a = 8'h02;
        rf_filter_coeff1_b = 8'h00;
        rf_filter_coeff2_a = 8'h03;
        rf_filter_coeff2_b = 8'h00;
        rf_filter_coeff3_a = 8'h04;
        rf_filter_coeff3_b = 8'h00;
        rf_filter_coeff4_a = 8'h05;
        rf_filter_coeff4_b = 8'h00;
        rf_filter_coeff5_a = 8'h06;
        rf_filter_coeff5_b = 8'h00;
        rf_filter_coeff6_a = 8'h07;
        rf_filter_coeff6_b = 8'h00;
        rf_filter_coeff7_a = 8'h08;
        rf_filter_coeff7_b = 8'h00;
        // Wait 100 ns for global reset to finish
        #100;
        rst_n = 1;
        aud_in_rts = 1;
        // Add stimulus here
        aud_in = 32'h00AA00AA;
        #150;
        aud_in = 32'h00AA003A;
        #150;
        aud_in = 32'h00DD00AA;
        #150;
        aud_in = 32'h001A00A2;
        #150;
        aud_in = 32'h001100DA;
        #150;
        aud_in = 32'h00AA00FF;
```

```
                #150;
                aud_in = 32'h00AF003A;
                #150;
                aud_in = 32'h00AA009F;
                #150;
                aud_in = 32'h00FF0022;
                #150;
                aud_in = 32'h00670029;
                #150;
                aud_in = 32'h00120074;
                #150;
                aud_in = 32'h004A0030;
                #150;
                aud_in = 32'h00C10021;
                #150;
                aud_in = 32'h00B90019;
                #150;
                aud_in = 32'h00A50003;
                #150;
                aud_in = 32'h00550032;
                #150;
        end
        always
                begin
                        forever #5 clk = ~clk;
                end

endmodule
```

filter_accumulator_tf.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_accumulator_tf.v
// Create Date:             11/3/2015
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps


module filter_accumulator_tf;

        // Inputs
        reg                             clk;
        reg                             rst_n;
        reg                             enable;
        reg                             load;
        reg             [15:0]          D;


        // Outputs
        wire            [19:0]          Q;

        // Instantiate the Unit Under Test (UUT)
        filter_accumulator uut (
    .clk(clk),
    .rst_n(rst_n),
    .enable(enable),
    .load(load),
    .D(D),
    .Q(Q)
    );


        initial begin
                // Initialize Inputs
                clk = 0;
                rst_n = 0;
                enable = 0;
                load = 0;

                #100;
                rst_n = 1;
                D = 16'h000A;
                enable = 1;
                load = 1;
                #10;
                load = 0;
                D = 16'hFFFF;
                #10;
                D = 16'h0334;
                #10;
                D = 16'hFFFF;
                #10;
                D = 16'hAA44;
                #100;
                enable = 0;
                #100;
                enable = 1;
                load = 1;
                D = 16'h0000;
                #100;
                load=0;
                D = 16'h0001;
                #100;
                rst_n = 0;
        end
```

```
        always
        begin
                forever #5 clk = ~clk;
        end

endmodule
```

filter_mux_tf.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_mux_tf.v
// Create Date:             11/23/2015
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps


module filter_mux_tf;

          // Inputs
          reg clk;
          reg rden;
          reg [9:0] rdptr;
          reg [7:0] rf_filter_coeff0_a;
          reg [7:0] rf_filter_coeff0_b;
          reg [7:0] rf_filter_coeff1_a;
          reg [7:0] rf_filter_coeff1_b;
          reg [7:0] rf_filter_coeff2_a;
          reg [7:0] rf_filter_coeff2_b;
          reg [7:0] rf_filter_coeff3_a;
          reg [7:0] rf_filter_coeff3_b;
          reg [7:0] rf_filter_coeff4_a;
          reg [7:0] rf_filter_coeff4_b;
          reg [7:0] rf_filter_coeff5_a;
          reg [7:0] rf_filter_coeff5_b;
          reg [7:0] rf_filter_coeff6_a;
          reg [7:0] rf_filter_coeff6_b;
          reg [7:0] rf_filter_coeff7_a;
          reg [7:0] rf_filter_coeff7_b;
          reg [7:0] rf_filter_coeff8_a;
          reg [7:0] rf_filter_coeff8_b;
          reg [7:0] rf_filter_coeff9_a;
          reg [7:0] rf_filter_coeff9_b;
          reg [7:0] rf_filter_coeff10_a;
          reg [7:0] rf_filter_coeff10_b;
          reg [7:0] rf_filter_coeff11_a;
          reg [7:0] rf_filter_coeff11_b;
          reg [7:0] rf_filter_coeff12_a;
          reg [7:0] rf_filter_coeff12_b;
          reg [7:0] rf_filter_coeff13_a;
          reg [7:0] rf_filter_coeff13_b;
          reg [7:0] rf_filter_coeff14_a;
          reg [7:0] rf_filter_coeff14_b;
          reg [7:0] rf_filter_coeff15_a;
          reg [7:0] rf_filter_coeff15_b;
          reg [7:0] rf_filter_coeff16_a;
          reg [7:0] rf_filter_coeff16_b;
          reg [7:0] rf_filter_coeff17_a;
          reg [7:0] rf_filter_coeff17_b;
          reg [7:0] rf_filter_coeff18_a;
          reg [7:0] rf_filter_coeff18_b;
          reg [7:0] rf_filter_coeff19_a;
          reg [7:0] rf_filter_coeff19_b;
          reg [7:0] rf_filter_coeff20_a;
          reg [7:0] rf_filter_coeff20_b;
          reg [7:0] rf_filter_coeff21_a;
          reg [7:0] rf_filter_coeff21_b;
          reg [7:0] rf_filter_coeff22_a;
          reg [7:0] rf_filter_coeff22_b;
          reg [7:0] rf_filter_coeff23_a;
          reg [7:0] rf_filter_coeff23_b;
          reg [7:0] rf_filter_coeff24_a;
          reg [7:0] rf_filter_coeff24_b;
          reg [7:0] rf_filter_coeff25_a;
          reg [7:0] rf_filter_coeff25_b;
```

```
reg [7:0] rf_filter_coeff26_a;
reg [7:0] rf_filter_coeff26_b;
reg [7:0] rf_filter_coeff27_a;
reg [7:0] rf_filter_coeff27_b;
reg [7:0] rf_filter_coeff28_a;
reg [7:0] rf_filter_coeff28_b;
reg [7:0] rf_filter_coeff29_a;
reg [7:0] rf_filter_coeff29_b;
reg [7:0] rf_filter_coeff30_a;
reg [7:0] rf_filter_coeff30_b;
reg [7:0] rf_filter_coeff31_a;
reg [7:0] rf_filter_coeff31_b;
reg [7:0] rf_filter_coeff32_a;
reg [7:0] rf_filter_coeff32_b;
reg [7:0] rf_filter_coeff33_a;
reg [7:0] rf_filter_coeff33_b;
reg [7:0] rf_filter_coeff34_a;
reg [7:0] rf_filter_coeff34_b;
reg [7:0] rf_filter_coeff35_a;
reg [7:0] rf_filter_coeff35_b;
reg [7:0] rf_filter_coeff36_a;
reg [7:0] rf_filter_coeff36_b;
reg [7:0] rf_filter_coeff37_a;
reg [7:0] rf_filter_coeff37_b;
reg [7:0] rf_filter_coeff38_a;
reg [7:0] rf_filter_coeff38_b;
reg [7:0] rf_filter_coeff39_a;
reg [7:0] rf_filter_coeff39_b;
reg [7:0] rf_filter_coeff40_a;
reg [7:0] rf_filter_coeff40_b;
reg [7:0] rf_filter_coeff41_a;
reg [7:0] rf_filter_coeff41_b;
reg [7:0] rf_filter_coeff42_a;
reg [7:0] rf_filter_coeff42_b;
reg [7:0] rf_filter_coeff43_a;
reg [7:0] rf_filter_coeff43_b;
reg [7:0] rf_filter_coeff44_a;
reg [7:0] rf_filter_coeff44_b;
reg [7:0] rf_filter_coeff45_a;
reg [7:0] rf_filter_coeff45_b;
reg [7:0] rf_filter_coeff46_a;
reg [7:0] rf_filter_coeff46_b;
reg [7:0] rf_filter_coeff47_a;
reg [7:0] rf_filter_coeff47_b;
reg [7:0] rf_filter_coeff48_a;
reg [7:0] rf_filter_coeff48_b;
reg [7:0] rf_filter_coeff49_a;
reg [7:0] rf_filter_coeff49_b;
reg [7:0] rf_filter_coeff50_a;
reg [7:0] rf_filter_coeff50_b;
reg [7:0] rf_filter_coeff51_a;
reg [7:0] rf_filter_coeff51_b;
reg [7:0] rf_filter_coeff52_a;
reg [7:0] rf_filter_coeff52_b;
reg [7:0] rf_filter_coeff53_a;
reg [7:0] rf_filter_coeff53_b;
reg [7:0] rf_filter_coeff54_a;
reg [7:0] rf_filter_coeff54_b;
reg [7:0] rf_filter_coeff55_a;
reg [7:0] rf_filter_coeff55_b;
reg [7:0] rf_filter_coeff56_a;
reg [7:0] rf_filter_coeff56_b;
reg [7:0] rf_filter_coeff57_a;
reg [7:0] rf_filter_coeff57_b;
reg [7:0] rf_filter_coeff58_a;
reg [7:0] rf_filter_coeff58_b;
reg [7:0] rf_filter_coeff59_a;
reg [7:0] rf_filter_coeff59_b;
reg [7:0] rf_filter_coeff60_a;
reg [7:0] rf_filter_coeff60_b;
reg [7:0] rf_filter_coeff61_a;
```

```
reg [7:0] rf_filter_coeff61_b;
reg [7:0] rf_filter_coeff62_a;
reg [7:0] rf_filter_coeff62_b;
reg [7:0] rf_filter_coeff63_a;
reg [7:0] rf_filter_coeff63_b;
reg [7:0] rf_filter_coeff64_a;
reg [7:0] rf_filter_coeff64_b;
reg [7:0] rf_filter_coeff65_a;
reg [7:0] rf_filter_coeff65_b;
reg [7:0] rf_filter_coeff66_a;
reg [7:0] rf_filter_coeff66_b;
reg [7:0] rf_filter_coeff67_a;
reg [7:0] rf_filter_coeff67_b;
reg [7:0] rf_filter_coeff68_a;
reg [7:0] rf_filter_coeff68_b;
reg [7:0] rf_filter_coeff69_a;
reg [7:0] rf_filter_coeff69_b;
reg [7:0] rf_filter_coeff70_a;
reg [7:0] rf_filter_coeff70_b;
reg [7:0] rf_filter_coeff71_a;
reg [7:0] rf_filter_coeff71_b;
reg [7:0] rf_filter_coeff72_a;
reg [7:0] rf_filter_coeff72_b;
reg [7:0] rf_filter_coeff73_a;
reg [7:0] rf_filter_coeff73_b;
reg [7:0] rf_filter_coeff74_a;
reg [7:0] rf_filter_coeff74_b;
reg [7:0] rf_filter_coeff75_a;
reg [7:0] rf_filter_coeff75_b;
reg [7:0] rf_filter_coeff76_a;
reg [7:0] rf_filter_coeff76_b;
reg [7:0] rf_filter_coeff77_a;
reg [7:0] rf_filter_coeff77_b;
reg [7:0] rf_filter_coeff78_a;
reg [7:0] rf_filter_coeff78_b;
reg [7:0] rf_filter_coeff79_a;
reg [7:0] rf_filter_coeff79_b;
reg [7:0] rf_filter_coeff80_a;
reg [7:0] rf_filter_coeff80_b;
reg [7:0] rf_filter_coeff81_a;
reg [7:0] rf_filter_coeff81_b;
reg [7:0] rf_filter_coeff82_a;
reg [7:0] rf_filter_coeff82_b;
reg [7:0] rf_filter_coeff83_a;
reg [7:0] rf_filter_coeff83_b;
reg [7:0] rf_filter_coeff84_a;
reg [7:0] rf_filter_coeff84_b;
reg [7:0] rf_filter_coeff85_a;
reg [7:0] rf_filter_coeff85_b;
reg [7:0] rf_filter_coeff86_a;
reg [7:0] rf_filter_coeff86_b;
reg [7:0] rf_filter_coeff87_a;
reg [7:0] rf_filter_coeff87_b;
reg [7:0] rf_filter_coeff88_a;
reg [7:0] rf_filter_coeff88_b;
reg [7:0] rf_filter_coeff89_a;
reg [7:0] rf_filter_coeff89_b;
reg [7:0] rf_filter_coeff90_a;
reg [7:0] rf_filter_coeff90_b;
reg [7:0] rf_filter_coeff91_a;
reg [7:0] rf_filter_coeff91_b;
reg [7:0] rf_filter_coeff92_a;
reg [7:0] rf_filter_coeff92_b;
reg [7:0] rf_filter_coeff93_a;
reg [7:0] rf_filter_coeff93_b;
reg [7:0] rf_filter_coeff94_a;
reg [7:0] rf_filter_coeff94_b;
reg [7:0] rf_filter_coeff95_a;
reg [7:0] rf_filter_coeff95_b;
reg [7:0] rf_filter_coeff96_a;
reg [7:0] rf_filter_coeff96_b;
```

```
reg [7:0] rf_filter_coeff97_a;
reg [7:0] rf_filter_coeff97_b;
reg [7:0] rf_filter_coeff98_a;
reg [7:0] rf_filter_coeff98_b;
reg [7:0] rf_filter_coeff99_a;
reg [7:0] rf_filter_coeff99_b;
reg [7:0] rf_filter_coeff100_a;
reg [7:0] rf_filter_coeff100_b;
reg [7:0] rf_filter_coeff101_a;
reg [7:0] rf_filter_coeff101_b;
reg [7:0] rf_filter_coeff102_a;
reg [7:0] rf_filter_coeff102_b;
reg [7:0] rf_filter_coeff103_a;
reg [7:0] rf_filter_coeff103_b;
reg [7:0] rf_filter_coeff104_a;
reg [7:0] rf_filter_coeff104_b;
reg [7:0] rf_filter_coeff105_a;
reg [7:0] rf_filter_coeff105_b;
reg [7:0] rf_filter_coeff106_a;
reg [7:0] rf_filter_coeff106_b;
reg [7:0] rf_filter_coeff107_a;
reg [7:0] rf_filter_coeff107_b;
reg [7:0] rf_filter_coeff108_a;
reg [7:0] rf_filter_coeff108_b;
reg [7:0] rf_filter_coeff109_a;
reg [7:0] rf_filter_coeff109_b;
reg [7:0] rf_filter_coeff110_a;
reg [7:0] rf_filter_coeff110_b;
reg [7:0] rf_filter_coeff111_a;
reg [7:0] rf_filter_coeff111_b;
reg [7:0] rf_filter_coeff112_a;
reg [7:0] rf_filter_coeff112_b;
reg [7:0] rf_filter_coeff113_a;
reg [7:0] rf_filter_coeff113_b;
reg [7:0] rf_filter_coeff114_a;
reg [7:0] rf_filter_coeff114_b;
reg [7:0] rf_filter_coeff115_a;
reg [7:0] rf_filter_coeff115_b;
reg [7:0] rf_filter_coeff116_a;
reg [7:0] rf_filter_coeff116_b;
reg [7:0] rf_filter_coeff117_a;
reg [7:0] rf_filter_coeff117_b;
reg [7:0] rf_filter_coeff118_a;
reg [7:0] rf_filter_coeff118_b;
reg [7:0] rf_filter_coeff119_a;
reg [7:0] rf_filter_coeff119_b;
reg [7:0] rf_filter_coeff120_a;
reg [7:0] rf_filter_coeff120_b;
reg [7:0] rf_filter_coeff121_a;
reg [7:0] rf_filter_coeff121_b;
reg [7:0] rf_filter_coeff122_a;
reg [7:0] rf_filter_coeff122_b;
reg [7:0] rf_filter_coeff123_a;
reg [7:0] rf_filter_coeff123_b;
reg [7:0] rf_filter_coeff124_a;
reg [7:0] rf_filter_coeff124_b;
reg [7:0] rf_filter_coeff125_a;
reg [7:0] rf_filter_coeff125_b;
reg [7:0] rf_filter_coeff126_a;
reg [7:0] rf_filter_coeff126_b;
reg [7:0] rf_filter_coeff127_a;
reg [7:0] rf_filter_coeff127_b;
reg [7:0] rf_filter_coeff128_a;
reg [7:0] rf_filter_coeff128_b;
reg [7:0] rf_filter_coeff129_a;
reg [7:0] rf_filter_coeff129_b;
reg [7:0] rf_filter_coeff130_a;
reg [7:0] rf_filter_coeff130_b;
reg [7:0] rf_filter_coeff131_a;
reg [7:0] rf_filter_coeff131_b;
reg [7:0] rf_filter_coeff132_a;
```

```
reg [7:0] rf_filter_coeff132_b;
reg [7:0] rf_filter_coeff133_a;
reg [7:0] rf_filter_coeff133_b;
reg [7:0] rf_filter_coeff134_a;
reg [7:0] rf_filter_coeff134_b;
reg [7:0] rf_filter_coeff135_a;
reg [7:0] rf_filter_coeff135_b;
reg [7:0] rf_filter_coeff136_a;
reg [7:0] rf_filter_coeff136_b;
reg [7:0] rf_filter_coeff137_a;
reg [7:0] rf_filter_coeff137_b;
reg [7:0] rf_filter_coeff138_a;
reg [7:0] rf_filter_coeff138_b;
reg [7:0] rf_filter_coeff139_a;
reg [7:0] rf_filter_coeff139_b;
reg [7:0] rf_filter_coeff140_a;
reg [7:0] rf_filter_coeff140_b;
reg [7:0] rf_filter_coeff141_a;
reg [7:0] rf_filter_coeff141_b;
reg [7:0] rf_filter_coeff142_a;
reg [7:0] rf_filter_coeff142_b;
reg [7:0] rf_filter_coeff143_a;
reg [7:0] rf_filter_coeff143_b;
reg [7:0] rf_filter_coeff144_a;
reg [7:0] rf_filter_coeff144_b;
reg [7:0] rf_filter_coeff145_a;
reg [7:0] rf_filter_coeff145_b;
reg [7:0] rf_filter_coeff146_a;
reg [7:0] rf_filter_coeff146_b;
reg [7:0] rf_filter_coeff147_a;
reg [7:0] rf_filter_coeff147_b;
reg [7:0] rf_filter_coeff148_a;
reg [7:0] rf_filter_coeff148_b;
reg [7:0] rf_filter_coeff149_a;
reg [7:0] rf_filter_coeff149_b;
reg [7:0] rf_filter_coeff150_a;
reg [7:0] rf_filter_coeff150_b;
reg [7:0] rf_filter_coeff151_a;
reg [7:0] rf_filter_coeff151_b;
reg [7:0] rf_filter_coeff152_a;
reg [7:0] rf_filter_coeff152_b;
reg [7:0] rf_filter_coeff153_a;
reg [7:0] rf_filter_coeff153_b;
reg [7:0] rf_filter_coeff154_a;
reg [7:0] rf_filter_coeff154_b;
reg [7:0] rf_filter_coeff155_a;
reg [7:0] rf_filter_coeff155_b;
reg [7:0] rf_filter_coeff156_a;
reg [7:0] rf_filter_coeff156_b;
reg [7:0] rf_filter_coeff157_a;
reg [7:0] rf_filter_coeff157_b;
reg [7:0] rf_filter_coeff158_a;
reg [7:0] rf_filter_coeff158_b;
reg [7:0] rf_filter_coeff159_a;
reg [7:0] rf_filter_coeff159_b;
reg [7:0] rf_filter_coeff160_a;
reg [7:0] rf_filter_coeff160_b;
reg [7:0] rf_filter_coeff161_a;
reg [7:0] rf_filter_coeff161_b;
reg [7:0] rf_filter_coeff162_a;
reg [7:0] rf_filter_coeff162_b;
reg [7:0] rf_filter_coeff163_a;
reg [7:0] rf_filter_coeff163_b;
reg [7:0] rf_filter_coeff164_a;
reg [7:0] rf_filter_coeff164_b;
reg [7:0] rf_filter_coeff165_a;
reg [7:0] rf_filter_coeff165_b;
reg [7:0] rf_filter_coeff166_a;
reg [7:0] rf_filter_coeff166_b;
reg [7:0] rf_filter_coeff167_a;
reg [7:0] rf_filter_coeff167_b;
```

```
reg [7:0] rf_filter_coeff168_a;
reg [7:0] rf_filter_coeff168_b;
reg [7:0] rf_filter_coeff169_a;
reg [7:0] rf_filter_coeff169_b;
reg [7:0] rf_filter_coeff170_a;
reg [7:0] rf_filter_coeff170_b;
reg [7:0] rf_filter_coeff171_a;
reg [7:0] rf_filter_coeff171_b;
reg [7:0] rf_filter_coeff172_a;
reg [7:0] rf_filter_coeff172_b;
reg [7:0] rf_filter_coeff173_a;
reg [7:0] rf_filter_coeff173_b;
reg [7:0] rf_filter_coeff174_a;
reg [7:0] rf_filter_coeff174_b;
reg [7:0] rf_filter_coeff175_a;
reg [7:0] rf_filter_coeff175_b;
reg [7:0] rf_filter_coeff176_a;
reg [7:0] rf_filter_coeff176_b;
reg [7:0] rf_filter_coeff177_a;
reg [7:0] rf_filter_coeff177_b;
reg [7:0] rf_filter_coeff178_a;
reg [7:0] rf_filter_coeff178_b;
reg [7:0] rf_filter_coeff179_a;
reg [7:0] rf_filter_coeff179_b;
reg [7:0] rf_filter_coeff180_a;
reg [7:0] rf_filter_coeff180_b;
reg [7:0] rf_filter_coeff181_a;
reg [7:0] rf_filter_coeff181_b;
reg [7:0] rf_filter_coeff182_a;
reg [7:0] rf_filter_coeff182_b;
reg [7:0] rf_filter_coeff183_a;
reg [7:0] rf_filter_coeff183_b;
reg [7:0] rf_filter_coeff184_a;
reg [7:0] rf_filter_coeff184_b;
reg [7:0] rf_filter_coeff185_a;
reg [7:0] rf_filter_coeff185_b;
reg [7:0] rf_filter_coeff186_a;
reg [7:0] rf_filter_coeff186_b;
reg [7:0] rf_filter_coeff187_a;
reg [7:0] rf_filter_coeff187_b;
reg [7:0] rf_filter_coeff188_a;
reg [7:0] rf_filter_coeff188_b;
reg [7:0] rf_filter_coeff189_a;
reg [7:0] rf_filter_coeff189_b;
reg [7:0] rf_filter_coeff190_a;
reg [7:0] rf_filter_coeff190_b;
reg [7:0] rf_filter_coeff191_a;
reg [7:0] rf_filter_coeff191_b;
reg [7:0] rf_filter_coeff192_a;
reg [7:0] rf_filter_coeff192_b;
reg [7:0] rf_filter_coeff193_a;
reg [7:0] rf_filter_coeff193_b;
reg [7:0] rf_filter_coeff194_a;
reg [7:0] rf_filter_coeff194_b;
reg [7:0] rf_filter_coeff195_a;
reg [7:0] rf_filter_coeff195_b;
reg [7:0] rf_filter_coeff196_a;
reg [7:0] rf_filter_coeff196_b;
reg [7:0] rf_filter_coeff197_a;
reg [7:0] rf_filter_coeff197_b;
reg [7:0] rf_filter_coeff198_a;
reg [7:0] rf_filter_coeff198_b;
reg [7:0] rf_filter_coeff199_a;
reg [7:0] rf_filter_coeff199_b;
reg [7:0] rf_filter_coeff200_a;
reg [7:0] rf_filter_coeff200_b;
reg [7:0] rf_filter_coeff201_a;
reg [7:0] rf_filter_coeff201_b;
reg [7:0] rf_filter_coeff202_a;
reg [7:0] rf_filter_coeff202_b;
reg [7:0] rf_filter_coeff203_a;
```

```
reg [7:0] rf_filter_coeff203_b;
reg [7:0] rf_filter_coeff204_a;
reg [7:0] rf_filter_coeff204_b;
reg [7:0] rf_filter_coeff205_a;
reg [7:0] rf_filter_coeff205_b;
reg [7:0] rf_filter_coeff206_a;
reg [7:0] rf_filter_coeff206_b;
reg [7:0] rf_filter_coeff207_a;
reg [7:0] rf_filter_coeff207_b;
reg [7:0] rf_filter_coeff208_a;
reg [7:0] rf_filter_coeff208_b;
reg [7:0] rf_filter_coeff209_a;
reg [7:0] rf_filter_coeff209_b;
reg [7:0] rf_filter_coeff210_a;
reg [7:0] rf_filter_coeff210_b;
reg [7:0] rf_filter_coeff211_a;
reg [7:0] rf_filter_coeff211_b;
reg [7:0] rf_filter_coeff212_a;
reg [7:0] rf_filter_coeff212_b;
reg [7:0] rf_filter_coeff213_a;
reg [7:0] rf_filter_coeff213_b;
reg [7:0] rf_filter_coeff214_a;
reg [7:0] rf_filter_coeff214_b;
reg [7:0] rf_filter_coeff215_a;
reg [7:0] rf_filter_coeff215_b;
reg [7:0] rf_filter_coeff216_a;
reg [7:0] rf_filter_coeff216_b;
reg [7:0] rf_filter_coeff217_a;
reg [7:0] rf_filter_coeff217_b;
reg [7:0] rf_filter_coeff218_a;
reg [7:0] rf_filter_coeff218_b;
reg [7:0] rf_filter_coeff219_a;
reg [7:0] rf_filter_coeff219_b;
reg [7:0] rf_filter_coeff220_a;
reg [7:0] rf_filter_coeff220_b;
reg [7:0] rf_filter_coeff221_a;
reg [7:0] rf_filter_coeff221_b;
reg [7:0] rf_filter_coeff222_a;
reg [7:0] rf_filter_coeff222_b;
reg [7:0] rf_filter_coeff223_a;
reg [7:0] rf_filter_coeff223_b;
reg [7:0] rf_filter_coeff224_a;
reg [7:0] rf_filter_coeff224_b;
reg [7:0] rf_filter_coeff225_a;
reg [7:0] rf_filter_coeff225_b;
reg [7:0] rf_filter_coeff226_a;
reg [7:0] rf_filter_coeff226_b;
reg [7:0] rf_filter_coeff227_a;
reg [7:0] rf_filter_coeff227_b;
reg [7:0] rf_filter_coeff228_a;
reg [7:0] rf_filter_coeff228_b;
reg [7:0] rf_filter_coeff229_a;
reg [7:0] rf_filter_coeff229_b;
reg [7:0] rf_filter_coeff230_a;
reg [7:0] rf_filter_coeff230_b;
reg [7:0] rf_filter_coeff231_a;
reg [7:0] rf_filter_coeff231_b;
reg [7:0] rf_filter_coeff232_a;
reg [7:0] rf_filter_coeff232_b;
reg [7:0] rf_filter_coeff233_a;
reg [7:0] rf_filter_coeff233_b;
reg [7:0] rf_filter_coeff234_a;
reg [7:0] rf_filter_coeff234_b;
reg [7:0] rf_filter_coeff235_a;
reg [7:0] rf_filter_coeff235_b;
reg [7:0] rf_filter_coeff236_a;
reg [7:0] rf_filter_coeff236_b;
reg [7:0] rf_filter_coeff237_a;
reg [7:0] rf_filter_coeff237_b;
reg [7:0] rf_filter_coeff238_a;
reg [7:0] rf_filter_coeff238_b;
```

```
reg [7:0] rf_filter_coeff239_a;
reg [7:0] rf_filter_coeff239_b;
reg [7:0] rf_filter_coeff240_a;
reg [7:0] rf_filter_coeff240_b;
reg [7:0] rf_filter_coeff241_a;
reg [7:0] rf_filter_coeff241_b;
reg [7:0] rf_filter_coeff242_a;
reg [7:0] rf_filter_coeff242_b;
reg [7:0] rf_filter_coeff243_a;
reg [7:0] rf_filter_coeff243_b;
reg [7:0] rf_filter_coeff244_a;
reg [7:0] rf_filter_coeff244_b;
reg [7:0] rf_filter_coeff245_a;
reg [7:0] rf_filter_coeff245_b;
reg [7:0] rf_filter_coeff246_a;
reg [7:0] rf_filter_coeff246_b;
reg [7:0] rf_filter_coeff247_a;
reg [7:0] rf_filter_coeff247_b;
reg [7:0] rf_filter_coeff248_a;
reg [7:0] rf_filter_coeff248_b;
reg [7:0] rf_filter_coeff249_a;
reg [7:0] rf_filter_coeff249_b;
reg [7:0] rf_filter_coeff250_a;
reg [7:0] rf_filter_coeff250_b;
reg [7:0] rf_filter_coeff251_a;
reg [7:0] rf_filter_coeff251_b;
reg [7:0] rf_filter_coeff252_a;
reg [7:0] rf_filter_coeff252_b;
reg [7:0] rf_filter_coeff253_a;
reg [7:0] rf_filter_coeff253_b;
reg [7:0] rf_filter_coeff254_a;
reg [7:0] rf_filter_coeff254_b;
reg [7:0] rf_filter_coeff255_a;
reg [7:0] rf_filter_coeff255_b;
reg [7:0] rf_filter_coeff256_a;
reg [7:0] rf_filter_coeff256_b;
reg [7:0] rf_filter_coeff257_a;
reg [7:0] rf_filter_coeff257_b;
reg [7:0] rf_filter_coeff258_a;
reg [7:0] rf_filter_coeff258_b;
reg [7:0] rf_filter_coeff259_a;
reg [7:0] rf_filter_coeff259_b;
reg [7:0] rf_filter_coeff260_a;
reg [7:0] rf_filter_coeff260_b;
reg [7:0] rf_filter_coeff261_a;
reg [7:0] rf_filter_coeff261_b;
reg [7:0] rf_filter_coeff262_a;
reg [7:0] rf_filter_coeff262_b;
reg [7:0] rf_filter_coeff263_a;
reg [7:0] rf_filter_coeff263_b;
reg [7:0] rf_filter_coeff264_a;
reg [7:0] rf_filter_coeff264_b;
reg [7:0] rf_filter_coeff265_a;
reg [7:0] rf_filter_coeff265_b;
reg [7:0] rf_filter_coeff266_a;
reg [7:0] rf_filter_coeff266_b;
reg [7:0] rf_filter_coeff267_a;
reg [7:0] rf_filter_coeff267_b;
reg [7:0] rf_filter_coeff268_a;
reg [7:0] rf_filter_coeff268_b;
reg [7:0] rf_filter_coeff269_a;
reg [7:0] rf_filter_coeff269_b;
reg [7:0] rf_filter_coeff270_a;
reg [7:0] rf_filter_coeff270_b;
reg [7:0] rf_filter_coeff271_a;
reg [7:0] rf_filter_coeff271_b;
reg [7:0] rf_filter_coeff272_a;
reg [7:0] rf_filter_coeff272_b;
reg [7:0] rf_filter_coeff273_a;
reg [7:0] rf_filter_coeff273_b;
reg [7:0] rf_filter_coeff274_a;
```

```
reg [7:0] rf_filter_coeff274_b;
reg [7:0] rf_filter_coeff275_a;
reg [7:0] rf_filter_coeff275_b;
reg [7:0] rf_filter_coeff276_a;
reg [7:0] rf_filter_coeff276_b;
reg [7:0] rf_filter_coeff277_a;
reg [7:0] rf_filter_coeff277_b;
reg [7:0] rf_filter_coeff278_a;
reg [7:0] rf_filter_coeff278_b;
reg [7:0] rf_filter_coeff279_a;
reg [7:0] rf_filter_coeff279_b;
reg [7:0] rf_filter_coeff280_a;
reg [7:0] rf_filter_coeff280_b;
reg [7:0] rf_filter_coeff281_a;
reg [7:0] rf_filter_coeff281_b;
reg [7:0] rf_filter_coeff282_a;
reg [7:0] rf_filter_coeff282_b;
reg [7:0] rf_filter_coeff283_a;
reg [7:0] rf_filter_coeff283_b;
reg [7:0] rf_filter_coeff284_a;
reg [7:0] rf_filter_coeff284_b;
reg [7:0] rf_filter_coeff285_a;
reg [7:0] rf_filter_coeff285_b;
reg [7:0] rf_filter_coeff286_a;
reg [7:0] rf_filter_coeff286_b;
reg [7:0] rf_filter_coeff287_a;
reg [7:0] rf_filter_coeff287_b;
reg [7:0] rf_filter_coeff288_a;
reg [7:0] rf_filter_coeff288_b;
reg [7:0] rf_filter_coeff289_a;
reg [7:0] rf_filter_coeff289_b;
reg [7:0] rf_filter_coeff290_a;
reg [7:0] rf_filter_coeff290_b;
reg [7:0] rf_filter_coeff291_a;
reg [7:0] rf_filter_coeff291_b;
reg [7:0] rf_filter_coeff292_a;
reg [7:0] rf_filter_coeff292_b;
reg [7:0] rf_filter_coeff293_a;
reg [7:0] rf_filter_coeff293_b;
reg [7:0] rf_filter_coeff294_a;
reg [7:0] rf_filter_coeff294_b;
reg [7:0] rf_filter_coeff295_a;
reg [7:0] rf_filter_coeff295_b;
reg [7:0] rf_filter_coeff296_a;
reg [7:0] rf_filter_coeff296_b;
reg [7:0] rf_filter_coeff297_a;
reg [7:0] rf_filter_coeff297_b;
reg [7:0] rf_filter_coeff298_a;
reg [7:0] rf_filter_coeff298_b;
reg [7:0] rf_filter_coeff299_a;
reg [7:0] rf_filter_coeff299_b;
reg [7:0] rf_filter_coeff300_a;
reg [7:0] rf_filter_coeff300_b;
reg [7:0] rf_filter_coeff301_a;
reg [7:0] rf_filter_coeff301_b;
reg [7:0] rf_filter_coeff302_a;
reg [7:0] rf_filter_coeff302_b;
reg [7:0] rf_filter_coeff303_a;
reg [7:0] rf_filter_coeff303_b;
reg [7:0] rf_filter_coeff304_a;
reg [7:0] rf_filter_coeff304_b;
reg [7:0] rf_filter_coeff305_a;
reg [7:0] rf_filter_coeff305_b;
reg [7:0] rf_filter_coeff306_a;
reg [7:0] rf_filter_coeff306_b;
reg [7:0] rf_filter_coeff307_a;
reg [7:0] rf_filter_coeff307_b;
reg [7:0] rf_filter_coeff308_a;
reg [7:0] rf_filter_coeff308_b;
reg [7:0] rf_filter_coeff309_a;
reg [7:0] rf_filter_coeff309_b;
```

```
reg [7:0] rf_filter_coeff310_a;
reg [7:0] rf_filter_coeff310_b;
reg [7:0] rf_filter_coeff311_a;
reg [7:0] rf_filter_coeff311_b;
reg [7:0] rf_filter_coeff312_a;
reg [7:0] rf_filter_coeff312_b;
reg [7:0] rf_filter_coeff313_a;
reg [7:0] rf_filter_coeff313_b;
reg [7:0] rf_filter_coeff314_a;
reg [7:0] rf_filter_coeff314_b;
reg [7:0] rf_filter_coeff315_a;
reg [7:0] rf_filter_coeff315_b;
reg [7:0] rf_filter_coeff316_a;
reg [7:0] rf_filter_coeff316_b;
reg [7:0] rf_filter_coeff317_a;
reg [7:0] rf_filter_coeff317_b;
reg [7:0] rf_filter_coeff318_a;
reg [7:0] rf_filter_coeff318_b;
reg [7:0] rf_filter_coeff319_a;
reg [7:0] rf_filter_coeff319_b;
reg [7:0] rf_filter_coeff320_a;
reg [7:0] rf_filter_coeff320_b;
reg [7:0] rf_filter_coeff321_a;
reg [7:0] rf_filter_coeff321_b;
reg [7:0] rf_filter_coeff322_a;
reg [7:0] rf_filter_coeff322_b;
reg [7:0] rf_filter_coeff323_a;
reg [7:0] rf_filter_coeff323_b;
reg [7:0] rf_filter_coeff324_a;
reg [7:0] rf_filter_coeff324_b;
reg [7:0] rf_filter_coeff325_a;
reg [7:0] rf_filter_coeff325_b;
reg [7:0] rf_filter_coeff326_a;
reg [7:0] rf_filter_coeff326_b;
reg [7:0] rf_filter_coeff327_a;
reg [7:0] rf_filter_coeff327_b;
reg [7:0] rf_filter_coeff328_a;
reg [7:0] rf_filter_coeff328_b;
reg [7:0] rf_filter_coeff329_a;
reg [7:0] rf_filter_coeff329_b;
reg [7:0] rf_filter_coeff330_a;
reg [7:0] rf_filter_coeff330_b;
reg [7:0] rf_filter_coeff331_a;
reg [7:0] rf_filter_coeff331_b;
reg [7:0] rf_filter_coeff332_a;
reg [7:0] rf_filter_coeff332_b;
reg [7:0] rf_filter_coeff333_a;
reg [7:0] rf_filter_coeff333_b;
reg [7:0] rf_filter_coeff334_a;
reg [7:0] rf_filter_coeff334_b;
reg [7:0] rf_filter_coeff335_a;
reg [7:0] rf_filter_coeff335_b;
reg [7:0] rf_filter_coeff336_a;
reg [7:0] rf_filter_coeff336_b;
reg [7:0] rf_filter_coeff337_a;
reg [7:0] rf_filter_coeff337_b;
reg [7:0] rf_filter_coeff338_a;
reg [7:0] rf_filter_coeff338_b;
reg [7:0] rf_filter_coeff339_a;
reg [7:0] rf_filter_coeff339_b;
reg [7:0] rf_filter_coeff340_a;
reg [7:0] rf_filter_coeff340_b;
reg [7:0] rf_filter_coeff341_a;
reg [7:0] rf_filter_coeff341_b;
reg [7:0] rf_filter_coeff342_a;
reg [7:0] rf_filter_coeff342_b;
reg [7:0] rf_filter_coeff343_a;
reg [7:0] rf_filter_coeff343_b;
reg [7:0] rf_filter_coeff344_a;
reg [7:0] rf_filter_coeff344_b;
reg [7:0] rf_filter_coeff345_a;
```

```
reg [7:0] rf_filter_coeff345_b;
reg [7:0] rf_filter_coeff346_a;
reg [7:0] rf_filter_coeff346_b;
reg [7:0] rf_filter_coeff347_a;
reg [7:0] rf_filter_coeff347_b;
reg [7:0] rf_filter_coeff348_a;
reg [7:0] rf_filter_coeff348_b;
reg [7:0] rf_filter_coeff349_a;
reg [7:0] rf_filter_coeff349_b;
reg [7:0] rf_filter_coeff350_a;
reg [7:0] rf_filter_coeff350_b;
reg [7:0] rf_filter_coeff351_a;
reg [7:0] rf_filter_coeff351_b;
reg [7:0] rf_filter_coeff352_a;
reg [7:0] rf_filter_coeff352_b;
reg [7:0] rf_filter_coeff353_a;
reg [7:0] rf_filter_coeff353_b;
reg [7:0] rf_filter_coeff354_a;
reg [7:0] rf_filter_coeff354_b;
reg [7:0] rf_filter_coeff355_a;
reg [7:0] rf_filter_coeff355_b;
reg [7:0] rf_filter_coeff356_a;
reg [7:0] rf_filter_coeff356_b;
reg [7:0] rf_filter_coeff357_a;
reg [7:0] rf_filter_coeff357_b;
reg [7:0] rf_filter_coeff358_a;
reg [7:0] rf_filter_coeff358_b;
reg [7:0] rf_filter_coeff359_a;
reg [7:0] rf_filter_coeff359_b;
reg [7:0] rf_filter_coeff360_a;
reg [7:0] rf_filter_coeff360_b;
reg [7:0] rf_filter_coeff361_a;
reg [7:0] rf_filter_coeff361_b;
reg [7:0] rf_filter_coeff362_a;
reg [7:0] rf_filter_coeff362_b;
reg [7:0] rf_filter_coeff363_a;
reg [7:0] rf_filter_coeff363_b;
reg [7:0] rf_filter_coeff364_a;
reg [7:0] rf_filter_coeff364_b;
reg [7:0] rf_filter_coeff365_a;
reg [7:0] rf_filter_coeff365_b;
reg [7:0] rf_filter_coeff366_a;
reg [7:0] rf_filter_coeff366_b;
reg [7:0] rf_filter_coeff367_a;
reg [7:0] rf_filter_coeff367_b;
reg [7:0] rf_filter_coeff368_a;
reg [7:0] rf_filter_coeff368_b;
reg [7:0] rf_filter_coeff369_a;
reg [7:0] rf_filter_coeff369_b;
reg [7:0] rf_filter_coeff370_a;
reg [7:0] rf_filter_coeff370_b;
reg [7:0] rf_filter_coeff371_a;
reg [7:0] rf_filter_coeff371_b;
reg [7:0] rf_filter_coeff372_a;
reg [7:0] rf_filter_coeff372_b;
reg [7:0] rf_filter_coeff373_a;
reg [7:0] rf_filter_coeff373_b;
reg [7:0] rf_filter_coeff374_a;
reg [7:0] rf_filter_coeff374_b;
reg [7:0] rf_filter_coeff375_a;
reg [7:0] rf_filter_coeff375_b;
reg [7:0] rf_filter_coeff376_a;
reg [7:0] rf_filter_coeff376_b;
reg [7:0] rf_filter_coeff377_a;
reg [7:0] rf_filter_coeff377_b;
reg [7:0] rf_filter_coeff378_a;
reg [7:0] rf_filter_coeff378_b;
reg [7:0] rf_filter_coeff379_a;
reg [7:0] rf_filter_coeff379_b;
reg [7:0] rf_filter_coeff380_a;
reg [7:0] rf_filter_coeff380_b;
```

```
reg [7:0] rf_filter_coeff381_a;
reg [7:0] rf_filter_coeff381_b;
reg [7:0] rf_filter_coeff382_a;
reg [7:0] rf_filter_coeff382_b;
reg [7:0] rf_filter_coeff383_a;
reg [7:0] rf_filter_coeff383_b;
reg [7:0] rf_filter_coeff384_a;
reg [7:0] rf_filter_coeff384_b;
reg [7:0] rf_filter_coeff385_a;
reg [7:0] rf_filter_coeff385_b;
reg [7:0] rf_filter_coeff386_a;
reg [7:0] rf_filter_coeff386_b;
reg [7:0] rf_filter_coeff387_a;
reg [7:0] rf_filter_coeff387_b;
reg [7:0] rf_filter_coeff388_a;
reg [7:0] rf_filter_coeff388_b;
reg [7:0] rf_filter_coeff389_a;
reg [7:0] rf_filter_coeff389_b;
reg [7:0] rf_filter_coeff390_a;
reg [7:0] rf_filter_coeff390_b;
reg [7:0] rf_filter_coeff391_a;
reg [7:0] rf_filter_coeff391_b;
reg [7:0] rf_filter_coeff392_a;
reg [7:0] rf_filter_coeff392_b;
reg [7:0] rf_filter_coeff393_a;
reg [7:0] rf_filter_coeff393_b;
reg [7:0] rf_filter_coeff394_a;
reg [7:0] rf_filter_coeff394_b;
reg [7:0] rf_filter_coeff395_a;
reg [7:0] rf_filter_coeff395_b;
reg [7:0] rf_filter_coeff396_a;
reg [7:0] rf_filter_coeff396_b;
reg [7:0] rf_filter_coeff397_a;
reg [7:0] rf_filter_coeff397_b;
reg [7:0] rf_filter_coeff398_a;
reg [7:0] rf_filter_coeff398_b;
reg [7:0] rf_filter_coeff399_a;
reg [7:0] rf_filter_coeff399_b;
reg [7:0] rf_filter_coeff400_a;
reg [7:0] rf_filter_coeff400_b;
reg [7:0] rf_filter_coeff401_a;
reg [7:0] rf_filter_coeff401_b;
reg [7:0] rf_filter_coeff402_a;
reg [7:0] rf_filter_coeff402_b;
reg [7:0] rf_filter_coeff403_a;
reg [7:0] rf_filter_coeff403_b;
reg [7:0] rf_filter_coeff404_a;
reg [7:0] rf_filter_coeff404_b;
reg [7:0] rf_filter_coeff405_a;
reg [7:0] rf_filter_coeff405_b;
reg [7:0] rf_filter_coeff406_a;
reg [7:0] rf_filter_coeff406_b;
reg [7:0] rf_filter_coeff407_a;
reg [7:0] rf_filter_coeff407_b;
reg [7:0] rf_filter_coeff408_a;
reg [7:0] rf_filter_coeff408_b;
reg [7:0] rf_filter_coeff409_a;
reg [7:0] rf_filter_coeff409_b;
reg [7:0] rf_filter_coeff410_a;
reg [7:0] rf_filter_coeff410_b;
reg [7:0] rf_filter_coeff411_a;
reg [7:0] rf_filter_coeff411_b;
reg [7:0] rf_filter_coeff412_a;
reg [7:0] rf_filter_coeff412_b;
reg [7:0] rf_filter_coeff413_a;
reg [7:0] rf_filter_coeff413_b;
reg [7:0] rf_filter_coeff414_a;
reg [7:0] rf_filter_coeff414_b;
reg [7:0] rf_filter_coeff415_a;
reg [7:0] rf_filter_coeff415_b;
reg [7:0] rf_filter_coeff416_a;
```

```
reg [7:0] rf_filter_coeff416_b;
reg [7:0] rf_filter_coeff417_a;
reg [7:0] rf_filter_coeff417_b;
reg [7:0] rf_filter_coeff418_a;
reg [7:0] rf_filter_coeff418_b;
reg [7:0] rf_filter_coeff419_a;
reg [7:0] rf_filter_coeff419_b;
reg [7:0] rf_filter_coeff420_a;
reg [7:0] rf_filter_coeff420_b;
reg [7:0] rf_filter_coeff421_a;
reg [7:0] rf_filter_coeff421_b;
reg [7:0] rf_filter_coeff422_a;
reg [7:0] rf_filter_coeff422_b;
reg [7:0] rf_filter_coeff423_a;
reg [7:0] rf_filter_coeff423_b;
reg [7:0] rf_filter_coeff424_a;
reg [7:0] rf_filter_coeff424_b;
reg [7:0] rf_filter_coeff425_a;
reg [7:0] rf_filter_coeff425_b;
reg [7:0] rf_filter_coeff426_a;
reg [7:0] rf_filter_coeff426_b;
reg [7:0] rf_filter_coeff427_a;
reg [7:0] rf_filter_coeff427_b;
reg [7:0] rf_filter_coeff428_a;
reg [7:0] rf_filter_coeff428_b;
reg [7:0] rf_filter_coeff429_a;
reg [7:0] rf_filter_coeff429_b;
reg [7:0] rf_filter_coeff430_a;
reg [7:0] rf_filter_coeff430_b;
reg [7:0] rf_filter_coeff431_a;
reg [7:0] rf_filter_coeff431_b;
reg [7:0] rf_filter_coeff432_a;
reg [7:0] rf_filter_coeff432_b;
reg [7:0] rf_filter_coeff433_a;
reg [7:0] rf_filter_coeff433_b;
reg [7:0] rf_filter_coeff434_a;
reg [7:0] rf_filter_coeff434_b;
reg [7:0] rf_filter_coeff435_a;
reg [7:0] rf_filter_coeff435_b;
reg [7:0] rf_filter_coeff436_a;
reg [7:0] rf_filter_coeff436_b;
reg [7:0] rf_filter_coeff437_a;
reg [7:0] rf_filter_coeff437_b;
reg [7:0] rf_filter_coeff438_a;
reg [7:0] rf_filter_coeff438_b;
reg [7:0] rf_filter_coeff439_a;
reg [7:0] rf_filter_coeff439_b;
reg [7:0] rf_filter_coeff440_a;
reg [7:0] rf_filter_coeff440_b;
reg [7:0] rf_filter_coeff441_a;
reg [7:0] rf_filter_coeff441_b;
reg [7:0] rf_filter_coeff442_a;
reg [7:0] rf_filter_coeff442_b;
reg [7:0] rf_filter_coeff443_a;
reg [7:0] rf_filter_coeff443_b;
reg [7:0] rf_filter_coeff444_a;
reg [7:0] rf_filter_coeff444_b;
reg [7:0] rf_filter_coeff445_a;
reg [7:0] rf_filter_coeff445_b;
reg [7:0] rf_filter_coeff446_a;
reg [7:0] rf_filter_coeff446_b;
reg [7:0] rf_filter_coeff447_a;
reg [7:0] rf_filter_coeff447_b;
reg [7:0] rf_filter_coeff448_a;
reg [7:0] rf_filter_coeff448_b;
reg [7:0] rf_filter_coeff449_a;
reg [7:0] rf_filter_coeff449_b;
reg [7:0] rf_filter_coeff450_a;
reg [7:0] rf_filter_coeff450_b;
reg [7:0] rf_filter_coeff451_a;
reg [7:0] rf_filter_coeff451_b;
```

```
reg [7:0] rf_filter_coeff452_a;
reg [7:0] rf_filter_coeff452_b;
reg [7:0] rf_filter_coeff453_a;
reg [7:0] rf_filter_coeff453_b;
reg [7:0] rf_filter_coeff454_a;
reg [7:0] rf_filter_coeff454_b;
reg [7:0] rf_filter_coeff455_a;
reg [7:0] rf_filter_coeff455_b;
reg [7:0] rf_filter_coeff456_a;
reg [7:0] rf_filter_coeff456_b;
reg [7:0] rf_filter_coeff457_a;
reg [7:0] rf_filter_coeff457_b;
reg [7:0] rf_filter_coeff458_a;
reg [7:0] rf_filter_coeff458_b;
reg [7:0] rf_filter_coeff459_a;
reg [7:0] rf_filter_coeff459_b;
reg [7:0] rf_filter_coeff460_a;
reg [7:0] rf_filter_coeff460_b;
reg [7:0] rf_filter_coeff461_a;
reg [7:0] rf_filter_coeff461_b;
reg [7:0] rf_filter_coeff462_a;
reg [7:0] rf_filter_coeff462_b;
reg [7:0] rf_filter_coeff463_a;
reg [7:0] rf_filter_coeff463_b;
reg [7:0] rf_filter_coeff464_a;
reg [7:0] rf_filter_coeff464_b;
reg [7:0] rf_filter_coeff465_a;
reg [7:0] rf_filter_coeff465_b;
reg [7:0] rf_filter_coeff466_a;
reg [7:0] rf_filter_coeff466_b;
reg [7:0] rf_filter_coeff467_a;
reg [7:0] rf_filter_coeff467_b;
reg [7:0] rf_filter_coeff468_a;
reg [7:0] rf_filter_coeff468_b;
reg [7:0] rf_filter_coeff469_a;
reg [7:0] rf_filter_coeff469_b;
reg [7:0] rf_filter_coeff470_a;
reg [7:0] rf_filter_coeff470_b;
reg [7:0] rf_filter_coeff471_a;
reg [7:0] rf_filter_coeff471_b;
reg [7:0] rf_filter_coeff472_a;
reg [7:0] rf_filter_coeff472_b;
reg [7:0] rf_filter_coeff473_a;
reg [7:0] rf_filter_coeff473_b;
reg [7:0] rf_filter_coeff474_a;
reg [7:0] rf_filter_coeff474_b;
reg [7:0] rf_filter_coeff475_a;
reg [7:0] rf_filter_coeff475_b;
reg [7:0] rf_filter_coeff476_a;
reg [7:0] rf_filter_coeff476_b;
reg [7:0] rf_filter_coeff477_a;
reg [7:0] rf_filter_coeff477_b;
reg [7:0] rf_filter_coeff478_a;
reg [7:0] rf_filter_coeff478_b;
reg [7:0] rf_filter_coeff479_a;
reg [7:0] rf_filter_coeff479_b;
reg [7:0] rf_filter_coeff480_a;
reg [7:0] rf_filter_coeff480_b;
reg [7:0] rf_filter_coeff481_a;
reg [7:0] rf_filter_coeff481_b;
reg [7:0] rf_filter_coeff482_a;
reg [7:0] rf_filter_coeff482_b;
reg [7:0] rf_filter_coeff483_a;
reg [7:0] rf_filter_coeff483_b;
reg [7:0] rf_filter_coeff484_a;
reg [7:0] rf_filter_coeff484_b;
reg [7:0] rf_filter_coeff485_a;
reg [7:0] rf_filter_coeff485_b;
reg [7:0] rf_filter_coeff486_a;
reg [7:0] rf_filter_coeff486_b;
reg [7:0] rf_filter_coeff487_a;
```

```
reg [7:0] rf_filter_coeff487_b;
reg [7:0] rf_filter_coeff488_a;
reg [7:0] rf_filter_coeff488_b;
reg [7:0] rf_filter_coeff489_a;
reg [7:0] rf_filter_coeff489_b;
reg [7:0] rf_filter_coeff490_a;
reg [7:0] rf_filter_coeff490_b;
reg [7:0] rf_filter_coeff491_a;
reg [7:0] rf_filter_coeff491_b;
reg [7:0] rf_filter_coeff492_a;
reg [7:0] rf_filter_coeff492_b;
reg [7:0] rf_filter_coeff493_a;
reg [7:0] rf_filter_coeff493_b;
reg [7:0] rf_filter_coeff494_a;
reg [7:0] rf_filter_coeff494_b;
reg [7:0] rf_filter_coeff495_a;
reg [7:0] rf_filter_coeff495_b;
reg [7:0] rf_filter_coeff496_a;
reg [7:0] rf_filter_coeff496_b;
reg [7:0] rf_filter_coeff497_a;
reg [7:0] rf_filter_coeff497_b;
reg [7:0] rf_filter_coeff498_a;
reg [7:0] rf_filter_coeff498_b;
reg [7:0] rf_filter_coeff499_a;
reg [7:0] rf_filter_coeff499_b;
reg [7:0] rf_filter_coeff500_a;
reg [7:0] rf_filter_coeff500_b;
reg [7:0] rf_filter_coeff501_a;
reg [7:0] rf_filter_coeff501_b;
reg [7:0] rf_filter_coeff502_a;
reg [7:0] rf_filter_coeff502_b;
reg [7:0] rf_filter_coeff503_a;
reg [7:0] rf_filter_coeff503_b;
reg [7:0] rf_filter_coeff504_a;
reg [7:0] rf_filter_coeff504_b;
reg [7:0] rf_filter_coeff505_a;
reg [7:0] rf_filter_coeff505_b;
reg [7:0] rf_filter_coeff506_a;
reg [7:0] rf_filter_coeff506_b;
reg [7:0] rf_filter_coeff507_a;
reg [7:0] rf_filter_coeff507_b;
reg [7:0] rf_filter_coeff508_a;
reg [7:0] rf_filter_coeff508_b;
reg [7:0] rf_filter_coeff509_a;
reg [7:0] rf_filter_coeff509_b;
reg [7:0] rf_filter_coeff510_a;
reg [7:0] rf_filter_coeff510_b;
reg [7:0] rf_filter_coeff511_a;
reg [7:0] rf_filter_coeff511_b;

// Outputs
wire [15:0] rddata;

// Instantiate the Unit Under Test (UUT)
filter_mux uut (
          .clk(clk),
          .rden(rden),
          .rdptr(rdptr),
          .rddata(rddata),
          .rf_filter_coeff0_a(rf_filter_coeff0_a),
          .rf_filter_coeff0_b(rf_filter_coeff0_b),
          .rf_filter_coeff1_a(rf_filter_coeff1_a),
          .rf_filter_coeff1_b(rf_filter_coeff1_b),
          .rf_filter_coeff2_a(rf_filter_coeff2_a),
          .rf_filter_coeff2_b(rf_filter_coeff2_b),
          .rf_filter_coeff3_a(rf_filter_coeff3_a),
          .rf_filter_coeff3_b(rf_filter_coeff3_b),
          .rf_filter_coeff4_a(rf_filter_coeff4_a),
          .rf_filter_coeff4_b(rf_filter_coeff4_b),
          .rf_filter_coeff5_a(rf_filter_coeff5_a),
          .rf_filter_coeff5_b(rf_filter_coeff5_b),
```

```
.rf_filter_coeff6_a(rf_filter_coeff6_a),
.rf_filter_coeff6_b(rf_filter_coeff6_b),
.rf_filter_coeff7_a(rf_filter_coeff7_a),
.rf_filter_coeff7_b(rf_filter_coeff7_b),
.rf_filter_coeff8_a(rf_filter_coeff8_a),
.rf_filter_coeff8_b(rf_filter_coeff8_b),
.rf_filter_coeff9_a(rf_filter_coeff9_a),
.rf_filter_coeff9_b(rf_filter_coeff9_b),
.rf_filter_coeff10_a(rf_filter_coeff10_a),
.rf_filter_coeff10_b(rf_filter_coeff10_b),
.rf_filter_coeff11_a(rf_filter_coeff11_a),
.rf_filter_coeff11_b(rf_filter_coeff11_b),
.rf_filter_coeff12_a(rf_filter_coeff12_a),
.rf_filter_coeff12_b(rf_filter_coeff12_b),
.rf_filter_coeff13_a(rf_filter_coeff13_a),
.rf_filter_coeff13_b(rf_filter_coeff13_b),
.rf_filter_coeff14_a(rf_filter_coeff14_a),
.rf_filter_coeff14_b(rf_filter_coeff14_b),
.rf_filter_coeff15_a(rf_filter_coeff15_a),
.rf_filter_coeff15_b(rf_filter_coeff15_b),
.rf_filter_coeff16_a(rf_filter_coeff16_a),
.rf_filter_coeff16_b(rf_filter_coeff16_b),
.rf_filter_coeff17_a(rf_filter_coeff17_a),
.rf_filter_coeff17_b(rf_filter_coeff17_b),
.rf_filter_coeff18_a(rf_filter_coeff18_a),
.rf_filter_coeff18_b(rf_filter_coeff18_b),
.rf_filter_coeff19_a(rf_filter_coeff19_a),
.rf_filter_coeff19_b(rf_filter_coeff19_b),
.rf_filter_coeff20_a(rf_filter_coeff20_a),
.rf_filter_coeff20_b(rf_filter_coeff20_b),
.rf_filter_coeff21_a(rf_filter_coeff21_a),
.rf_filter_coeff21_b(rf_filter_coeff21_b),
.rf_filter_coeff22_a(rf_filter_coeff22_a),
.rf_filter_coeff22_b(rf_filter_coeff22_b),
.rf_filter_coeff23_a(rf_filter_coeff23_a),
.rf_filter_coeff23_b(rf_filter_coeff23_b),
.rf_filter_coeff24_a(rf_filter_coeff24_a),
.rf_filter_coeff24_b(rf_filter_coeff24_b),
.rf_filter_coeff25_a(rf_filter_coeff25_a),
.rf_filter_coeff25_b(rf_filter_coeff25_b),
.rf_filter_coeff26_a(rf_filter_coeff26_a),
.rf_filter_coeff26_b(rf_filter_coeff26_b),
.rf_filter_coeff27_a(rf_filter_coeff27_a),
.rf_filter_coeff27_b(rf_filter_coeff27_b),
.rf_filter_coeff28_a(rf_filter_coeff28_a),
.rf_filter_coeff28_b(rf_filter_coeff28_b),
.rf_filter_coeff29_a(rf_filter_coeff29_a),
.rf_filter_coeff29_b(rf_filter_coeff29_b),
.rf_filter_coeff30_a(rf_filter_coeff30_a),
.rf_filter_coeff30_b(rf_filter_coeff30_b),
.rf_filter_coeff31_a(rf_filter_coeff31_a),
.rf_filter_coeff31_b(rf_filter_coeff31_b),
.rf_filter_coeff32_a(rf_filter_coeff32_a),
.rf_filter_coeff32_b(rf_filter_coeff32_b),
.rf_filter_coeff33_a(rf_filter_coeff33_a),
.rf_filter_coeff33_b(rf_filter_coeff33_b),
.rf_filter_coeff34_a(rf_filter_coeff34_a),
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
```

```
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
```

```
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
```

```
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
```

```
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
```

```
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
```

```
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
```

```
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
```

```
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
```

```
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
```

```
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
```

```
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
```

```
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
```

```
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
.rf_filter_coeff479_a(rf_filter_coeff479_a),
.rf_filter_coeff479_b(rf_filter_coeff479_b),
.rf_filter_coeff480_a(rf_filter_coeff480_a),
.rf_filter_coeff480_b(rf_filter_coeff480_b),
.rf_filter_coeff481_a(rf_filter_coeff481_a),
.rf_filter_coeff481_b(rf_filter_coeff481_b),
.rf_filter_coeff482_a(rf_filter_coeff482_a),
.rf_filter_coeff482_b(rf_filter_coeff482_b),
.rf_filter_coeff483_a(rf_filter_coeff483_a),
.rf_filter_coeff483_b(rf_filter_coeff483_b),
.rf_filter_coeff484_a(rf_filter_coeff484_a),
.rf_filter_coeff484_b(rf_filter_coeff484_b),
.rf_filter_coeff485_a(rf_filter_coeff485_a),
.rf_filter_coeff485_b(rf_filter_coeff485_b),
.rf_filter_coeff486_a(rf_filter_coeff486_a),
.rf_filter_coeff486_b(rf_filter_coeff486_b),
.rf_filter_coeff487_a(rf_filter_coeff487_a),
.rf_filter_coeff487_b(rf_filter_coeff487_b),
.rf_filter_coeff488_a(rf_filter_coeff488_a),
.rf_filter_coeff488_b(rf_filter_coeff488_b),
.rf_filter_coeff489_a(rf_filter_coeff489_a),
.rf_filter_coeff489_b(rf_filter_coeff489_b),
.rf_filter_coeff490_a(rf_filter_coeff490_a),
.rf_filter_coeff490_b(rf_filter_coeff490_b),
.rf_filter_coeff491_a(rf_filter_coeff491_a),
.rf_filter_coeff491_b(rf_filter_coeff491_b),
.rf_filter_coeff492_a(rf_filter_coeff492_a),
.rf_filter_coeff492_b(rf_filter_coeff492_b),
.rf_filter_coeff493_a(rf_filter_coeff493_a),
.rf_filter_coeff493_b(rf_filter_coeff493_b),
.rf_filter_coeff494_a(rf_filter_coeff494_a),
.rf_filter_coeff494_b(rf_filter_coeff494_b),
.rf_filter_coeff495_a(rf_filter_coeff495_a),
.rf_filter_coeff495_b(rf_filter_coeff495_b),
.rf_filter_coeff496_a(rf_filter_coeff496_a),
.rf_filter_coeff496_b(rf_filter_coeff496_b),
.rf_filter_coeff497_a(rf_filter_coeff497_a),
.rf_filter_coeff497_b(rf_filter_coeff497_b),
.rf_filter_coeff498_a(rf_filter_coeff498_a),
.rf_filter_coeff498_b(rf_filter_coeff498_b),
.rf_filter_coeff499_a(rf_filter_coeff499_a),
.rf_filter_coeff499_b(rf_filter_coeff499_b),
.rf_filter_coeff500_a(rf_filter_coeff500_a),
.rf_filter_coeff500_b(rf_filter_coeff500_b),
.rf_filter_coeff501_a(rf_filter_coeff501_a),
.rf_filter_coeff501_b(rf_filter_coeff501_b),
.rf_filter_coeff502_a(rf_filter_coeff502_a),
.rf_filter_coeff502_b(rf_filter_coeff502_b),
```

```
        .rf_filter_coeff503_a(rf_filter_coeff503_a),
        .rf_filter_coeff503_b(rf_filter_coeff503_b),
        .rf_filter_coeff504_a(rf_filter_coeff504_a),
        .rf_filter_coeff504_b(rf_filter_coeff504_b),
        .rf_filter_coeff505_a(rf_filter_coeff505_a),
        .rf_filter_coeff505_b(rf_filter_coeff505_b),
        .rf_filter_coeff506_a(rf_filter_coeff506_a),
        .rf_filter_coeff506_b(rf_filter_coeff506_b),
        .rf_filter_coeff507_a(rf_filter_coeff507_a),
        .rf_filter_coeff507_b(rf_filter_coeff507_b),
        .rf_filter_coeff508_a(rf_filter_coeff508_a),
        .rf_filter_coeff508_b(rf_filter_coeff508_b),
        .rf_filter_coeff509_a(rf_filter_coeff509_a),
        .rf_filter_coeff509_b(rf_filter_coeff509_b),
        .rf_filter_coeff510_a(rf_filter_coeff510_a),
        .rf_filter_coeff510_b(rf_filter_coeff510_b),
        .rf_filter_coeff511_a(rf_filter_coeff511_a),
        .rf_filter_coeff511_b(rf_filter_coeff511_b)
);

initial begin
        // Initialize Inputs
        clk = 0;
        rden = 0;
        rdptr = 0;
        rf_filter_coeff0_a = 8'h00;
        rf_filter_coeff0_b = 8'h08;
        rf_filter_coeff1_a = 8'h01;
        rf_filter_coeff1_b = 8'h07;
        rf_filter_coeff2_a = 8'h02;
        rf_filter_coeff2_b = 8'h06;
        rf_filter_coeff3_a = 8'h03;
        rf_filter_coeff3_b = 8'h05;
        rf_filter_coeff4_a = 8'h04;
        rf_filter_coeff4_b = 8'h04;
        rf_filter_coeff5_a = 8'h05;
        rf_filter_coeff5_b = 8'h03;
        rf_filter_coeff6_a = 8'h06;
        rf_filter_coeff6_b = 8'h02;
        rf_filter_coeff7_a = 8'h07;
        rf_filter_coeff7_b = 8'h01;
        rf_filter_coeff8_a = 8'h08;
        rf_filter_coeff8_b = 8'h00;
        rf_filter_coeff9_a = 0;
        rf_filter_coeff9_b = 0;
        rf_filter_coeff10_a = 0;
        rf_filter_coeff10_b = 0;
        rf_filter_coeff11_a = 0;
        rf_filter_coeff11_b = 0;
        rf_filter_coeff12_a = 0;
        rf_filter_coeff12_b = 0;
        rf_filter_coeff13_a = 0;
        rf_filter_coeff13_b = 0;
        rf_filter_coeff14_a = 0;
        rf_filter_coeff14_b = 0;
        rf_filter_coeff15_a = 0;
        rf_filter_coeff15_b = 0;
        rf_filter_coeff16_a = 0;
        rf_filter_coeff16_b = 0;
        rf_filter_coeff17_a = 7;
        rf_filter_coeff17_b = 0;
        rf_filter_coeff18_a = 0;
        rf_filter_coeff18_b = 0;
        rf_filter_coeff19_a = 0;
        rf_filter_coeff19_b = 0;
        rf_filter_coeff20_a = 0;
        rf_filter_coeff20_b = 0;
        rf_filter_coeff21_a = 0;
        rf_filter_coeff21_b = 0;
        rf_filter_coeff22_a = 0;
        rf_filter_coeff22_b = 0;
```

```
rf_filter_coeff23_a = 0;
rf_filter_coeff23_b = 0;
rf_filter_coeff24_a = 0;
rf_filter_coeff24_b = 0;
rf_filter_coeff25_a = 0;
rf_filter_coeff25_b = 0;
rf_filter_coeff26_a = 0;
rf_filter_coeff26_b = 0;
rf_filter_coeff27_a = 0;
rf_filter_coeff27_b = 0;
rf_filter_coeff28_a = 0;
rf_filter_coeff28_b = 0;
rf_filter_coeff29_a = 0;
rf_filter_coeff29_b = 0;
rf_filter_coeff30_a = 0;
rf_filter_coeff30_b = 0;
rf_filter_coeff31_a = 0;
rf_filter_coeff31_b = 0;
rf_filter_coeff32_a = 0;
rf_filter_coeff32_b = 0;
rf_filter_coeff33_a = 0;
rf_filter_coeff33_b = 0;
rf_filter_coeff34_a = 0;
rf_filter_coeff34_b = 0;
rf_filter_coeff35_a = 0;
rf_filter_coeff35_b = 0;
rf_filter_coeff36_a = 0;
rf_filter_coeff36_b = 0;
rf_filter_coeff37_a = 0;
rf_filter_coeff37_b = 0;
rf_filter_coeff38_a = 0;
rf_filter_coeff38_b = 0;
rf_filter_coeff39_a = 0;
rf_filter_coeff39_b = 0;
rf_filter_coeff40_a = 0;
rf_filter_coeff40_b = 0;
rf_filter_coeff41_a = 0;
rf_filter_coeff41_b = 0;
rf_filter_coeff42_a = 0;
rf_filter_coeff42_b = 0;
rf_filter_coeff43_a = 0;
rf_filter_coeff43_b = 0;
rf_filter_coeff44_a = 0;
rf_filter_coeff44_b = 0;
rf_filter_coeff45_a = 0;
rf_filter_coeff45_b = 0;
rf_filter_coeff46_a = 0;
rf_filter_coeff46_b = 0;
rf_filter_coeff47_a = 0;
rf_filter_coeff47_b = 0;
rf_filter_coeff48_a = 0;
rf_filter_coeff48_b = 0;
rf_filter_coeff49_a = 0;
rf_filter_coeff49_b = 0;
rf_filter_coeff50_a = 0;
rf_filter_coeff50_b = 0;
rf_filter_coeff51_a = 0;
rf_filter_coeff51_b = 0;
rf_filter_coeff52_a = 0;
rf_filter_coeff52_b = 0;
rf_filter_coeff53_a = 0;
rf_filter_coeff53_b = 0;
rf_filter_coeff54_a = 0;
rf_filter_coeff54_b = 0;
rf_filter_coeff55_a = 0;
rf_filter_coeff55_b = 0;
rf_filter_coeff56_a = 0;
rf_filter_coeff56_b = 0;
rf_filter_coeff57_a = 0;
rf_filter_coeff57_b = 0;
rf_filter_coeff58_a = 0;
```

```
rf_filter_coeff58_b = 0;
rf_filter_coeff59_a = 0;
rf_filter_coeff59_b = 0;
rf_filter_coeff60_a = 0;
rf_filter_coeff60_b = 0;
rf_filter_coeff61_a = 0;
rf_filter_coeff61_b = 0;
rf_filter_coeff62_a = 0;
rf_filter_coeff62_b = 0;
rf_filter_coeff63_a = 0;
rf_filter_coeff63_b = 0;
rf_filter_coeff64_a = 0;
rf_filter_coeff64_b = 0;
rf_filter_coeff65_a = 0;
rf_filter_coeff65_b = 0;
rf_filter_coeff66_a = 0;
rf_filter_coeff66_b = 0;
rf_filter_coeff67_a = 0;
rf_filter_coeff67_b = 0;
rf_filter_coeff68_a = 0;
rf_filter_coeff68_b = 0;
rf_filter_coeff69_a = 0;
rf_filter_coeff69_b = 0;
rf_filter_coeff70_a = 0;
rf_filter_coeff70_b = 0;
rf_filter_coeff71_a = 0;
rf_filter_coeff71_b = 0;
rf_filter_coeff72_a = 0;
rf_filter_coeff72_b = 0;
rf_filter_coeff73_a = 0;
rf_filter_coeff73_b = 0;
rf_filter_coeff74_a = 0;
rf_filter_coeff74_b = 0;
rf_filter_coeff75_a = 0;
rf_filter_coeff75_b = 0;
rf_filter_coeff76_a = 0;
rf_filter_coeff76_b = 0;
rf_filter_coeff77_a = 0;
rf_filter_coeff77_b = 0;
rf_filter_coeff78_a = 0;
rf_filter_coeff78_b = 0;
rf_filter_coeff79_a = 0;
rf_filter_coeff79_b = 0;
rf_filter_coeff80_a = 0;
rf_filter_coeff80_b = 0;
rf_filter_coeff81_a = 0;
rf_filter_coeff81_b = 0;
rf_filter_coeff82_a = 0;
rf_filter_coeff82_b = 0;
rf_filter_coeff83_a = 0;
rf_filter_coeff83_b = 0;
rf_filter_coeff84_a = 0;
rf_filter_coeff84_b = 0;
rf_filter_coeff85_a = 0;
rf_filter_coeff85_b = 0;
rf_filter_coeff86_a = 0;
rf_filter_coeff86_b = 0;
rf_filter_coeff87_a = 0;
rf_filter_coeff87_b = 0;
rf_filter_coeff88_a = 0;
rf_filter_coeff88_b = 0;
rf_filter_coeff89_a = 0;
rf_filter_coeff89_b = 0;
rf_filter_coeff90_a = 0;
rf_filter_coeff90_b = 0;
rf_filter_coeff91_a = 0;
rf_filter_coeff91_b = 0;
rf_filter_coeff92_a = 0;
rf_filter_coeff92_b = 0;
rf_filter_coeff93_a = 0;
rf_filter_coeff93_b = 0;
```

```
rf_filter_coeff94_a = 0;
rf_filter_coeff94_b = 0;
rf_filter_coeff95_a = 0;
rf_filter_coeff95_b = 0;
rf_filter_coeff96_a = 0;
rf_filter_coeff96_b = 0;
rf_filter_coeff97_a = 0;
rf_filter_coeff97_b = 0;
rf_filter_coeff98_a = 0;
rf_filter_coeff98_b = 0;
rf_filter_coeff99_a = 0;
rf_filter_coeff99_b = 0;
rf_filter_coeff100_a = 0;
rf_filter_coeff100_b = 0;
rf_filter_coeff101_a = 0;
rf_filter_coeff101_b = 0;
rf_filter_coeff102_a = 0;
rf_filter_coeff102_b = 0;
rf_filter_coeff103_a = 0;
rf_filter_coeff103_b = 0;
rf_filter_coeff104_a = 0;
rf_filter_coeff104_b = 0;
rf_filter_coeff105_a = 0;
rf_filter_coeff105_b = 0;
rf_filter_coeff106_a = 0;
rf_filter_coeff106_b = 0;
rf_filter_coeff107_a = 0;
rf_filter_coeff107_b = 0;
rf_filter_coeff108_a = 0;
rf_filter_coeff108_b = 0;
rf_filter_coeff109_a = 0;
rf_filter_coeff109_b = 0;
rf_filter_coeff110_a = 0;
rf_filter_coeff110_b = 0;
rf_filter_coeff111_a = 0;
rf_filter_coeff111_b = 0;
rf_filter_coeff112_a = 0;
rf_filter_coeff112_b = 0;
rf_filter_coeff113_a = 0;
rf_filter_coeff113_b = 0;
rf_filter_coeff114_a = 0;
rf_filter_coeff114_b = 0;
rf_filter_coeff115_a = 0;
rf_filter_coeff115_b = 0;
rf_filter_coeff116_a = 0;
rf_filter_coeff116_b = 0;
rf_filter_coeff117_a = 0;
rf_filter_coeff117_b = 0;
rf_filter_coeff118_a = 0;
rf_filter_coeff118_b = 0;
rf_filter_coeff119_a = 0;
rf_filter_coeff119_b = 0;
rf_filter_coeff120_a = 0;
rf_filter_coeff120_b = 0;
rf_filter_coeff121_a = 0;
rf_filter_coeff121_b = 0;
rf_filter_coeff122_a = 0;
rf_filter_coeff122_b = 0;
rf_filter_coeff123_a = 0;
rf_filter_coeff123_b = 0;
rf_filter_coeff124_a = 0;
rf_filter_coeff124_b = 0;
rf_filter_coeff125_a = 0;
rf_filter_coeff125_b = 0;
rf_filter_coeff126_a = 0;
rf_filter_coeff126_b = 0;
rf_filter_coeff127_a = 0;
rf_filter_coeff127_b = 0;
rf_filter_coeff128_a = 0;
rf_filter_coeff128_b = 0;
rf_filter_coeff129_a = 0;
```

```
rf_filter_coeff129_b = 0;
rf_filter_coeff130_a = 0;
rf_filter_coeff130_b = 0;
rf_filter_coeff131_a = 0;
rf_filter_coeff131_b = 0;
rf_filter_coeff132_a = 0;
rf_filter_coeff132_b = 0;
rf_filter_coeff133_a = 0;
rf_filter_coeff133_b = 0;
rf_filter_coeff134_a = 0;
rf_filter_coeff134_b = 0;
rf_filter_coeff135_a = 0;
rf_filter_coeff135_b = 0;
rf_filter_coeff136_a = 0;
rf_filter_coeff136_b = 0;
rf_filter_coeff137_a = 0;
rf_filter_coeff137_b = 0;
rf_filter_coeff138_a = 0;
rf_filter_coeff138_b = 0;
rf_filter_coeff139_a = 0;
rf_filter_coeff139_b = 0;
rf_filter_coeff140_a = 0;
rf_filter_coeff140_b = 0;
rf_filter_coeff141_a = 0;
rf_filter_coeff141_b = 0;
rf_filter_coeff142_a = 0;
rf_filter_coeff142_b = 0;
rf_filter_coeff143_a = 0;
rf_filter_coeff143_b = 0;
rf_filter_coeff144_a = 0;
rf_filter_coeff144_b = 0;
rf_filter_coeff145_a = 0;
rf_filter_coeff145_b = 0;
rf_filter_coeff146_a = 0;
rf_filter_coeff146_b = 0;
rf_filter_coeff147_a = 0;
rf_filter_coeff147_b = 0;
rf_filter_coeff148_a = 0;
rf_filter_coeff148_b = 0;
rf_filter_coeff149_a = 0;
rf_filter_coeff149_b = 0;
rf_filter_coeff150_a = 0;
rf_filter_coeff150_b = 0;
rf_filter_coeff151_a = 0;
rf_filter_coeff151_b = 0;
rf_filter_coeff152_a = 0;
rf_filter_coeff152_b = 0;
rf_filter_coeff153_a = 0;
rf_filter_coeff153_b = 0;
rf_filter_coeff154_a = 0;
rf_filter_coeff154_b = 0;
rf_filter_coeff155_a = 0;
rf_filter_coeff155_b = 0;
rf_filter_coeff156_a = 0;
rf_filter_coeff156_b = 0;
rf_filter_coeff157_a = 0;
rf_filter_coeff157_b = 0;
rf_filter_coeff158_a = 0;
rf_filter_coeff158_b = 0;
rf_filter_coeff159_a = 0;
rf_filter_coeff159_b = 0;
rf_filter_coeff160_a = 0;
rf_filter_coeff160_b = 0;
rf_filter_coeff161_a = 0;
rf_filter_coeff161_b = 0;
rf_filter_coeff162_a = 0;
rf_filter_coeff162_b = 0;
rf_filter_coeff163_a = 0;
rf_filter_coeff163_b = 0;
rf_filter_coeff164_a = 0;
rf_filter_coeff164_b = 0;
```

```
rf_filter_coeff165_a = 0;
rf_filter_coeff165_b = 0;
rf_filter_coeff166_a = 0;
rf_filter_coeff166_b = 0;
rf_filter_coeff167_a = 0;
rf_filter_coeff167_b = 0;
rf_filter_coeff168_a = 0;
rf_filter_coeff168_b = 0;
rf_filter_coeff169_a = 0;
rf_filter_coeff169_b = 0;
rf_filter_coeff170_a = 0;
rf_filter_coeff170_b = 0;
rf_filter_coeff171_a = 0;
rf_filter_coeff171_b = 0;
rf_filter_coeff172_a = 0;
rf_filter_coeff172_b = 0;
rf_filter_coeff173_a = 0;
rf_filter_coeff173_b = 0;
rf_filter_coeff174_a = 0;
rf_filter_coeff174_b = 0;
rf_filter_coeff175_a = 0;
rf_filter_coeff175_b = 0;
rf_filter_coeff176_a = 0;
rf_filter_coeff176_b = 0;
rf_filter_coeff177_a = 0;
rf_filter_coeff177_b = 0;
rf_filter_coeff178_a = 0;
rf_filter_coeff178_b = 0;
rf_filter_coeff179_a = 0;
rf_filter_coeff179_b = 0;
rf_filter_coeff180_a = 0;
rf_filter_coeff180_b = 0;
rf_filter_coeff181_a = 0;
rf_filter_coeff181_b = 0;
rf_filter_coeff182_a = 0;
rf_filter_coeff182_b = 0;
rf_filter_coeff183_a = 0;
rf_filter_coeff183_b = 0;
rf_filter_coeff184_a = 0;
rf_filter_coeff184_b = 0;
rf_filter_coeff185_a = 0;
rf_filter_coeff185_b = 0;
rf_filter_coeff186_a = 0;
rf_filter_coeff186_b = 0;
rf_filter_coeff187_a = 0;
rf_filter_coeff187_b = 0;
rf_filter_coeff188_a = 0;
rf_filter_coeff188_b = 0;
rf_filter_coeff189_a = 0;
rf_filter_coeff189_b = 0;
rf_filter_coeff190_a = 0;
rf_filter_coeff190_b = 0;
rf_filter_coeff191_a = 0;
rf_filter_coeff191_b = 0;
rf_filter_coeff192_a = 0;
rf_filter_coeff192_b = 0;
rf_filter_coeff193_a = 0;
rf_filter_coeff193_b = 0;
rf_filter_coeff194_a = 0;
rf_filter_coeff194_b = 0;
rf_filter_coeff195_a = 0;
rf_filter_coeff195_b = 0;
rf_filter_coeff196_a = 0;
rf_filter_coeff196_b = 0;
rf_filter_coeff197_a = 0;
rf_filter_coeff197_b = 0;
rf_filter_coeff198_a = 0;
rf_filter_coeff198_b = 0;
rf_filter_coeff199_a = 0;
rf_filter_coeff199_b = 0;
rf_filter_coeff200_a = 0;
```

```
rf_filter_coeff200_b = 0;
rf_filter_coeff201_a = 0;
rf_filter_coeff201_b = 0;
rf_filter_coeff202_a = 0;
rf_filter_coeff202_b = 0;
rf_filter_coeff203_a = 0;
rf_filter_coeff203_b = 0;
rf_filter_coeff204_a = 0;
rf_filter_coeff204_b = 0;
rf_filter_coeff205_a = 0;
rf_filter_coeff205_b = 0;
rf_filter_coeff206_a = 0;
rf_filter_coeff206_b = 0;
rf_filter_coeff207_a = 0;
rf_filter_coeff207_b = 0;
rf_filter_coeff208_a = 0;
rf_filter_coeff208_b = 0;
rf_filter_coeff209_a = 0;
rf_filter_coeff209_b = 0;
rf_filter_coeff210_a = 0;
rf_filter_coeff210_b = 0;
rf_filter_coeff211_a = 0;
rf_filter_coeff211_b = 0;
rf_filter_coeff212_a = 0;
rf_filter_coeff212_b = 0;
rf_filter_coeff213_a = 0;
rf_filter_coeff213_b = 0;
rf_filter_coeff214_a = 0;
rf_filter_coeff214_b = 0;
rf_filter_coeff215_a = 0;
rf_filter_coeff215_b = 0;
rf_filter_coeff216_a = 0;
rf_filter_coeff216_b = 0;
rf_filter_coeff217_a = 0;
rf_filter_coeff217_b = 0;
rf_filter_coeff218_a = 0;
rf_filter_coeff218_b = 0;
rf_filter_coeff219_a = 0;
rf_filter_coeff219_b = 0;
rf_filter_coeff220_a = 0;
rf_filter_coeff220_b = 0;
rf_filter_coeff221_a = 0;
rf_filter_coeff221_b = 0;
rf_filter_coeff222_a = 0;
rf_filter_coeff222_b = 0;
rf_filter_coeff223_a = 0;
rf_filter_coeff223_b = 0;
rf_filter_coeff224_a = 0;
rf_filter_coeff224_b = 0;
rf_filter_coeff225_a = 0;
rf_filter_coeff225_b = 0;
rf_filter_coeff226_a = 0;
rf_filter_coeff226_b = 0;
rf_filter_coeff227_a = 0;
rf_filter_coeff227_b = 0;
rf_filter_coeff228_a = 0;
rf_filter_coeff228_b = 0;
rf_filter_coeff229_a = 0;
rf_filter_coeff229_b = 0;
rf_filter_coeff230_a = 0;
rf_filter_coeff230_b = 0;
rf_filter_coeff231_a = 0;
rf_filter_coeff231_b = 0;
rf_filter_coeff232_a = 0;
rf_filter_coeff232_b = 0;
rf_filter_coeff233_a = 0;
rf_filter_coeff233_b = 0;
rf_filter_coeff234_a = 0;
rf_filter_coeff234_b = 0;
rf_filter_coeff235_a = 0;
rf_filter_coeff235_b = 0;
```

```
rf_filter_coeff236_a = 0;
rf_filter_coeff236_b = 0;
rf_filter_coeff237_a = 0;
rf_filter_coeff237_b = 0;
rf_filter_coeff238_a = 0;
rf_filter_coeff238_b = 0;
rf_filter_coeff239_a = 0;
rf_filter_coeff239_b = 0;
rf_filter_coeff240_a = 0;
rf_filter_coeff240_b = 0;
rf_filter_coeff241_a = 0;
rf_filter_coeff241_b = 0;
rf_filter_coeff242_a = 0;
rf_filter_coeff242_b = 0;
rf_filter_coeff243_a = 0;
rf_filter_coeff243_b = 0;
rf_filter_coeff244_a = 0;
rf_filter_coeff244_b = 0;
rf_filter_coeff245_a = 0;
rf_filter_coeff245_b = 0;
rf_filter_coeff246_a = 0;
rf_filter_coeff246_b = 0;
rf_filter_coeff247_a = 0;
rf_filter_coeff247_b = 0;
rf_filter_coeff248_a = 0;
rf_filter_coeff248_b = 0;
rf_filter_coeff249_a = 0;
rf_filter_coeff249_b = 0;
rf_filter_coeff250_a = 0;
rf_filter_coeff250_b = 0;
rf_filter_coeff251_a = 0;
rf_filter_coeff251_b = 0;
rf_filter_coeff252_a = 0;
rf_filter_coeff252_b = 0;
rf_filter_coeff253_a = 0;
rf_filter_coeff253_b = 0;
rf_filter_coeff254_a = 0;
rf_filter_coeff254_b = 0;
rf_filter_coeff255_a = 0;
rf_filter_coeff255_b = 0;
rf_filter_coeff256_a = 0;
rf_filter_coeff256_b = 0;
rf_filter_coeff257_a = 0;
rf_filter_coeff257_b = 0;
rf_filter_coeff258_a = 0;
rf_filter_coeff258_b = 0;
rf_filter_coeff259_a = 0;
rf_filter_coeff259_b = 0;
rf_filter_coeff260_a = 0;
rf_filter_coeff260_b = 0;
rf_filter_coeff261_a = 0;
rf_filter_coeff261_b = 0;
rf_filter_coeff262_a = 0;
rf_filter_coeff262_b = 0;
rf_filter_coeff263_a = 0;
rf_filter_coeff263_b = 0;
rf_filter_coeff264_a = 0;
rf_filter_coeff264_b = 0;
rf_filter_coeff265_a = 0;
rf_filter_coeff265_b = 0;
rf_filter_coeff266_a = 0;
rf_filter_coeff266_b = 0;
rf_filter_coeff267_a = 0;
rf_filter_coeff267_b = 0;
rf_filter_coeff268_a = 0;
rf_filter_coeff268_b = 0;
rf_filter_coeff269_a = 0;
rf_filter_coeff269_b = 0;
rf_filter_coeff270_a = 0;
rf_filter_coeff270_b = 0;
rf_filter_coeff271_a = 0;
```

```
rf_filter_coeff271_b = 0;
rf_filter_coeff272_a = 0;
rf_filter_coeff272_b = 0;
rf_filter_coeff273_a = 0;
rf_filter_coeff273_b = 0;
rf_filter_coeff274_a = 0;
rf_filter_coeff274_b = 0;
rf_filter_coeff275_a = 0;
rf_filter_coeff275_b = 0;
rf_filter_coeff276_a = 0;
rf_filter_coeff276_b = 0;
rf_filter_coeff277_a = 0;
rf_filter_coeff277_b = 0;
rf_filter_coeff278_a = 0;
rf_filter_coeff278_b = 0;
rf_filter_coeff279_a = 0;
rf_filter_coeff279_b = 0;
rf_filter_coeff280_a = 0;
rf_filter_coeff280_b = 0;
rf_filter_coeff281_a = 0;
rf_filter_coeff281_b = 0;
rf_filter_coeff282_a = 0;
rf_filter_coeff282_b = 0;
rf_filter_coeff283_a = 0;
rf_filter_coeff283_b = 0;
rf_filter_coeff284_a = 0;
rf_filter_coeff284_b = 0;
rf_filter_coeff285_a = 0;
rf_filter_coeff285_b = 0;
rf_filter_coeff286_a = 0;
rf_filter_coeff286_b = 0;
rf_filter_coeff287_a = 0;
rf_filter_coeff287_b = 0;
rf_filter_coeff288_a = 0;
rf_filter_coeff288_b = 0;
rf_filter_coeff289_a = 0;
rf_filter_coeff289_b = 0;
rf_filter_coeff290_a = 0;
rf_filter_coeff290_b = 0;
rf_filter_coeff291_a = 0;
rf_filter_coeff291_b = 0;
rf_filter_coeff292_a = 0;
rf_filter_coeff292_b = 0;
rf_filter_coeff293_a = 0;
rf_filter_coeff293_b = 0;
rf_filter_coeff294_a = 0;
rf_filter_coeff294_b = 0;
rf_filter_coeff295_a = 0;
rf_filter_coeff295_b = 0;
rf_filter_coeff296_a = 0;
rf_filter_coeff296_b = 0;
rf_filter_coeff297_a = 0;
rf_filter_coeff297_b = 0;
rf_filter_coeff298_a = 0;
rf_filter_coeff298_b = 0;
rf_filter_coeff299_a = 0;
rf_filter_coeff299_b = 0;
rf_filter_coeff300_a = 0;
rf_filter_coeff300_b = 0;
rf_filter_coeff301_a = 0;
rf_filter_coeff301_b = 0;
rf_filter_coeff302_a = 0;
rf_filter_coeff302_b = 0;
rf_filter_coeff303_a = 0;
rf_filter_coeff303_b = 0;
rf_filter_coeff304_a = 0;
rf_filter_coeff304_b = 0;
rf_filter_coeff305_a = 0;
rf_filter_coeff305_b = 0;
rf_filter_coeff306_a = 0;
rf_filter_coeff306_b = 0;
```

```
rf_filter_coeff307_a = 0;
rf_filter_coeff307_b = 0;
rf_filter_coeff308_a = 0;
rf_filter_coeff308_b = 0;
rf_filter_coeff309_a = 0;
rf_filter_coeff309_b = 0;
rf_filter_coeff310_a = 0;
rf_filter_coeff310_b = 0;
rf_filter_coeff311_a = 0;
rf_filter_coeff311_b = 0;
rf_filter_coeff312_a = 0;
rf_filter_coeff312_b = 0;
rf_filter_coeff313_a = 0;
rf_filter_coeff313_b = 0;
rf_filter_coeff314_a = 0;
rf_filter_coeff314_b = 0;
rf_filter_coeff315_a = 0;
rf_filter_coeff315_b = 0;
rf_filter_coeff316_a = 0;
rf_filter_coeff316_b = 0;
rf_filter_coeff317_a = 0;
rf_filter_coeff317_b = 0;
rf_filter_coeff318_a = 0;
rf_filter_coeff318_b = 0;
rf_filter_coeff319_a = 0;
rf_filter_coeff319_b = 0;
rf_filter_coeff320_a = 0;
rf_filter_coeff320_b = 0;
rf_filter_coeff321_a = 0;
rf_filter_coeff321_b = 0;
rf_filter_coeff322_a = 0;
rf_filter_coeff322_b = 0;
rf_filter_coeff323_a = 0;
rf_filter_coeff323_b = 0;
rf_filter_coeff324_a = 0;
rf_filter_coeff324_b = 0;
rf_filter_coeff325_a = 0;
rf_filter_coeff325_b = 0;
rf_filter_coeff326_a = 0;
rf_filter_coeff326_b = 0;
rf_filter_coeff327_a = 0;
rf_filter_coeff327_b = 0;
rf_filter_coeff328_a = 0;
rf_filter_coeff328_b = 0;
rf_filter_coeff329_a = 0;
rf_filter_coeff329_b = 0;
rf_filter_coeff330_a = 0;
rf_filter_coeff330_b = 0;
rf_filter_coeff331_a = 0;
rf_filter_coeff331_b = 0;
rf_filter_coeff332_a = 0;
rf_filter_coeff332_b = 0;
rf_filter_coeff333_a = 0;
rf_filter_coeff333_b = 0;
rf_filter_coeff334_a = 0;
rf_filter_coeff334_b = 0;
rf_filter_coeff335_a = 0;
rf_filter_coeff335_b = 0;
rf_filter_coeff336_a = 0;
rf_filter_coeff336_b = 0;
rf_filter_coeff337_a = 0;
rf_filter_coeff337_b = 0;
rf_filter_coeff338_a = 0;
rf_filter_coeff338_b = 0;
rf_filter_coeff339_a = 0;
rf_filter_coeff339_b = 0;
rf_filter_coeff340_a = 0;
rf_filter_coeff340_b = 0;
rf_filter_coeff341_a = 0;
rf_filter_coeff341_b = 0;
rf_filter_coeff342_a = 0;
```

```
rf_filter_coeff342_b = 0;
rf_filter_coeff343_a = 0;
rf_filter_coeff343_b = 0;
rf_filter_coeff344_a = 0;
rf_filter_coeff344_b = 0;
rf_filter_coeff345_a = 0;
rf_filter_coeff345_b = 0;
rf_filter_coeff346_a = 0;
rf_filter_coeff346_b = 0;
rf_filter_coeff347_a = 0;
rf_filter_coeff347_b = 0;
rf_filter_coeff348_a = 0;
rf_filter_coeff348_b = 0;
rf_filter_coeff349_a = 0;
rf_filter_coeff349_b = 0;
rf_filter_coeff350_a = 0;
rf_filter_coeff350_b = 0;
rf_filter_coeff351_a = 0;
rf_filter_coeff351_b = 0;
rf_filter_coeff352_a = 0;
rf_filter_coeff352_b = 0;
rf_filter_coeff353_a = 0;
rf_filter_coeff353_b = 0;
rf_filter_coeff354_a = 0;
rf_filter_coeff354_b = 0;
rf_filter_coeff355_a = 0;
rf_filter_coeff355_b = 0;
rf_filter_coeff356_a = 0;
rf_filter_coeff356_b = 0;
rf_filter_coeff357_a = 0;
rf_filter_coeff357_b = 0;
rf_filter_coeff358_a = 0;
rf_filter_coeff358_b = 0;
rf_filter_coeff359_a = 0;
rf_filter_coeff359_b = 0;
rf_filter_coeff360_a = 0;
rf_filter_coeff360_b = 0;
rf_filter_coeff361_a = 0;
rf_filter_coeff361_b = 0;
rf_filter_coeff362_a = 0;
rf_filter_coeff362_b = 0;
rf_filter_coeff363_a = 0;
rf_filter_coeff363_b = 0;
rf_filter_coeff364_a = 0;
rf_filter_coeff364_b = 0;
rf_filter_coeff365_a = 0;
rf_filter_coeff365_b = 0;
rf_filter_coeff366_a = 0;
rf_filter_coeff366_b = 0;
rf_filter_coeff367_a = 0;
rf_filter_coeff367_b = 0;
rf_filter_coeff368_a = 0;
rf_filter_coeff368_b = 0;
rf_filter_coeff369_a = 0;
rf_filter_coeff369_b = 0;
rf_filter_coeff370_a = 0;
rf_filter_coeff370_b = 0;
rf_filter_coeff371_a = 0;
rf_filter_coeff371_b = 0;
rf_filter_coeff372_a = 0;
rf_filter_coeff372_b = 0;
rf_filter_coeff373_a = 0;
rf_filter_coeff373_b = 0;
rf_filter_coeff374_a = 0;
rf_filter_coeff374_b = 0;
rf_filter_coeff375_a = 0;
rf_filter_coeff375_b = 0;
rf_filter_coeff376_a = 0;
rf_filter_coeff376_b = 0;
rf_filter_coeff377_a = 0;
rf_filter_coeff377_b = 0;
```

```
rf_filter_coeff378_a = 0;
rf_filter_coeff378_b = 0;
rf_filter_coeff379_a = 0;
rf_filter_coeff379_b = 0;
rf_filter_coeff380_a = 0;
rf_filter_coeff380_b = 0;
rf_filter_coeff381_a = 0;
rf_filter_coeff381_b = 0;
rf_filter_coeff382_a = 0;
rf_filter_coeff382_b = 0;
rf_filter_coeff383_a = 0;
rf_filter_coeff383_b = 0;
rf_filter_coeff384_a = 0;
rf_filter_coeff384_b = 0;
rf_filter_coeff385_a = 0;
rf_filter_coeff385_b = 0;
rf_filter_coeff386_a = 0;
rf_filter_coeff386_b = 0;
rf_filter_coeff387_a = 0;
rf_filter_coeff387_b = 0;
rf_filter_coeff388_a = 0;
rf_filter_coeff388_b = 0;
rf_filter_coeff389_a = 0;
rf_filter_coeff389_b = 0;
rf_filter_coeff390_a = 0;
rf_filter_coeff390_b = 0;
rf_filter_coeff391_a = 0;
rf_filter_coeff391_b = 0;
rf_filter_coeff392_a = 0;
rf_filter_coeff392_b = 0;
rf_filter_coeff393_a = 0;
rf_filter_coeff393_b = 0;
rf_filter_coeff394_a = 0;
rf_filter_coeff394_b = 0;
rf_filter_coeff395_a = 0;
rf_filter_coeff395_b = 0;
rf_filter_coeff396_a = 0;
rf_filter_coeff396_b = 0;
rf_filter_coeff397_a = 0;
rf_filter_coeff397_b = 0;
rf_filter_coeff398_a = 0;
rf_filter_coeff398_b = 0;
rf_filter_coeff399_a = 0;
rf_filter_coeff399_b = 0;
rf_filter_coeff400_a = 0;
rf_filter_coeff400_b = 0;
rf_filter_coeff401_a = 0;
rf_filter_coeff401_b = 0;
rf_filter_coeff402_a = 0;
rf_filter_coeff402_b = 0;
rf_filter_coeff403_a = 0;
rf_filter_coeff403_b = 0;
rf_filter_coeff404_a = 0;
rf_filter_coeff404_b = 0;
rf_filter_coeff405_a = 0;
rf_filter_coeff405_b = 0;
rf_filter_coeff406_a = 0;
rf_filter_coeff406_b = 0;
rf_filter_coeff407_a = 0;
rf_filter_coeff407_b = 0;
rf_filter_coeff408_a = 0;
rf_filter_coeff408_b = 0;
rf_filter_coeff409_a = 0;
rf_filter_coeff409_b = 0;
rf_filter_coeff410_a = 0;
rf_filter_coeff410_b = 0;
rf_filter_coeff411_a = 0;
rf_filter_coeff411_b = 0;
rf_filter_coeff412_a = 0;
rf_filter_coeff412_b = 0;
rf_filter_coeff413_a = 0;
```

```
rf_filter_coeff413_b = 0;
rf_filter_coeff414_a = 0;
rf_filter_coeff414_b = 0;
rf_filter_coeff415_a = 0;
rf_filter_coeff415_b = 0;
rf_filter_coeff416_a = 0;
rf_filter_coeff416_b = 0;
rf_filter_coeff417_a = 0;
rf_filter_coeff417_b = 0;
rf_filter_coeff418_a = 0;
rf_filter_coeff418_b = 0;
rf_filter_coeff419_a = 0;
rf_filter_coeff419_b = 0;
rf_filter_coeff420_a = 0;
rf_filter_coeff420_b = 0;
rf_filter_coeff421_a = 0;
rf_filter_coeff421_b = 0;
rf_filter_coeff422_a = 0;
rf_filter_coeff422_b = 0;
rf_filter_coeff423_a = 0;
rf_filter_coeff423_b = 0;
rf_filter_coeff424_a = 0;
rf_filter_coeff424_b = 0;
rf_filter_coeff425_a = 0;
rf_filter_coeff425_b = 0;
rf_filter_coeff426_a = 0;
rf_filter_coeff426_b = 0;
rf_filter_coeff427_a = 0;
rf_filter_coeff427_b = 0;
rf_filter_coeff428_a = 0;
rf_filter_coeff428_b = 0;
rf_filter_coeff429_a = 0;
rf_filter_coeff429_b = 0;
rf_filter_coeff430_a = 0;
rf_filter_coeff430_b = 0;
rf_filter_coeff431_a = 0;
rf_filter_coeff431_b = 0;
rf_filter_coeff432_a = 0;
rf_filter_coeff432_b = 0;
rf_filter_coeff433_a = 0;
rf_filter_coeff433_b = 0;
rf_filter_coeff434_a = 0;
rf_filter_coeff434_b = 0;
rf_filter_coeff435_a = 0;
rf_filter_coeff435_b = 0;
rf_filter_coeff436_a = 0;
rf_filter_coeff436_b = 0;
rf_filter_coeff437_a = 0;
rf_filter_coeff437_b = 0;
rf_filter_coeff438_a = 0;
rf_filter_coeff438_b = 0;
rf_filter_coeff439_a = 0;
rf_filter_coeff439_b = 0;
rf_filter_coeff440_a = 0;
rf_filter_coeff440_b = 0;
rf_filter_coeff441_a = 0;
rf_filter_coeff441_b = 0;
rf_filter_coeff442_a = 0;
rf_filter_coeff442_b = 0;
rf_filter_coeff443_a = 0;
rf_filter_coeff443_b = 0;
rf_filter_coeff444_a = 0;
rf_filter_coeff444_b = 0;
rf_filter_coeff445_a = 0;
rf_filter_coeff445_b = 0;
rf_filter_coeff446_a = 0;
rf_filter_coeff446_b = 0;
rf_filter_coeff447_a = 0;
rf_filter_coeff447_b = 0;
rf_filter_coeff448_a = 0;
rf_filter_coeff448_b = 0;
```

```
rf_filter_coeff449_a = 0;
rf_filter_coeff449_b = 0;
rf_filter_coeff450_a = 0;
rf_filter_coeff450_b = 0;
rf_filter_coeff451_a = 0;
rf_filter_coeff451_b = 0;
rf_filter_coeff452_a = 0;
rf_filter_coeff452_b = 0;
rf_filter_coeff453_a = 0;
rf_filter_coeff453_b = 0;
rf_filter_coeff454_a = 0;
rf_filter_coeff454_b = 0;
rf_filter_coeff455_a = 0;
rf_filter_coeff455_b = 0;
rf_filter_coeff456_a = 0;
rf_filter_coeff456_b = 0;
rf_filter_coeff457_a = 0;
rf_filter_coeff457_b = 0;
rf_filter_coeff458_a = 0;
rf_filter_coeff458_b = 0;
rf_filter_coeff459_a = 0;
rf_filter_coeff459_b = 0;
rf_filter_coeff460_a = 0;
rf_filter_coeff460_b = 0;
rf_filter_coeff461_a = 0;
rf_filter_coeff461_b = 0;
rf_filter_coeff462_a = 0;
rf_filter_coeff462_b = 0;
rf_filter_coeff463_a = 0;
rf_filter_coeff463_b = 0;
rf_filter_coeff464_a = 0;
rf_filter_coeff464_b = 0;
rf_filter_coeff465_a = 0;
rf_filter_coeff465_b = 0;
rf_filter_coeff466_a = 0;
rf_filter_coeff466_b = 0;
rf_filter_coeff467_a = 0;
rf_filter_coeff467_b = 0;
rf_filter_coeff468_a = 0;
rf_filter_coeff468_b = 0;
rf_filter_coeff469_a = 0;
rf_filter_coeff469_b = 0;
rf_filter_coeff470_a = 0;
rf_filter_coeff470_b = 0;
rf_filter_coeff471_a = 0;
rf_filter_coeff471_b = 0;
rf_filter_coeff472_a = 0;
rf_filter_coeff472_b = 0;
rf_filter_coeff473_a = 0;
rf_filter_coeff473_b = 0;
rf_filter_coeff474_a = 0;
rf_filter_coeff474_b = 0;
rf_filter_coeff475_a = 0;
rf_filter_coeff475_b = 0;
rf_filter_coeff476_a = 0;
rf_filter_coeff476_b = 0;
rf_filter_coeff477_a = 0;
rf_filter_coeff477_b = 0;
rf_filter_coeff478_a = 0;
rf_filter_coeff478_b = 0;
rf_filter_coeff479_a = 0;
rf_filter_coeff479_b = 0;
rf_filter_coeff480_a = 0;
rf_filter_coeff480_b = 0;
rf_filter_coeff481_a = 0;
rf_filter_coeff481_b = 0;
rf_filter_coeff482_a = 0;
rf_filter_coeff482_b = 0;
rf_filter_coeff483_a = 0;
rf_filter_coeff483_b = 0;
rf_filter_coeff484_a = 0;
```

```
rf_filter_coeff484_b = 0;
rf_filter_coeff485_a = 0;
rf_filter_coeff485_b = 0;
rf_filter_coeff486_a = 0;
rf_filter_coeff486_b = 0;
rf_filter_coeff487_a = 0;
rf_filter_coeff487_b = 0;
rf_filter_coeff488_a = 0;
rf_filter_coeff488_b = 0;
rf_filter_coeff489_a = 0;
rf_filter_coeff489_b = 0;
rf_filter_coeff490_a = 0;
rf_filter_coeff490_b = 0;
rf_filter_coeff491_a = 0;
rf_filter_coeff491_b = 0;
rf_filter_coeff492_a = 0;
rf_filter_coeff492_b = 0;
rf_filter_coeff493_a = 0;
rf_filter_coeff493_b = 0;
rf_filter_coeff494_a = 0;
rf_filter_coeff494_b = 0;
rf_filter_coeff495_a = 0;
rf_filter_coeff495_b = 0;
rf_filter_coeff496_a = 0;
rf_filter_coeff496_b = 0;
rf_filter_coeff497_a = 0;
rf_filter_coeff497_b = 0;
rf_filter_coeff498_a = 0;
rf_filter_coeff498_b = 0;
rf_filter_coeff499_a = 0;
rf_filter_coeff499_b = 0;
rf_filter_coeff500_a = 0;
rf_filter_coeff500_b = 0;
rf_filter_coeff501_a = 0;
rf_filter_coeff501_b = 0;
rf_filter_coeff502_a = 0;
rf_filter_coeff502_b = 0;
rf_filter_coeff503_a = 0;
rf_filter_coeff503_b = 0;
rf_filter_coeff504_a = 0;
rf_filter_coeff504_b = 0;
rf_filter_coeff505_a = 0;
rf_filter_coeff505_b = 0;
rf_filter_coeff506_a = 0;
rf_filter_coeff506_b = 0;
rf_filter_coeff507_a = 0;
rf_filter_coeff507_b = 0;
rf_filter_coeff508_a = 0;
rf_filter_coeff508_b = 0;
rf_filter_coeff509_a = 0;
rf_filter_coeff509_b = 0;
rf_filter_coeff510_a = 0;
rf_filter_coeff510_b = 0;
rf_filter_coeff511_a = 0;
rf_filter_coeff511_b = 0;

// Wait 100 ns for global reset to finish
#100;

// Add stimulus here
rden = 1;
rdptr = 0;
#100;
rdptr = 1;
#100;
rdptr = 2;
#100;
rdptr = 3;
#100;
rdptr = 4;
#100;
```

```
                rdptr = 5;
                #100;
                rdptr = 6;
                #100;
                rdptr = 7;
                #100;
                rdptr = 8;
        end

        always
        begin
                forever #5 clk = ~clk;
        end

endmodule
```

filter_round_truncate_tf.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_round_truncate_tf.v
// Create Date:             ???
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module filter_round_truncate_tf;

          // Inputs
          reg                               clk;
          reg                               rst_n;
          reg             [39:0]            acc_in;
          reg                               rf_sat;
          reg             [ 2:0]            rf_shift;
          reg                               trig_out_clear;
          reg                               trig_filter_ovf_flag_clear;

          // Outputs
          wire            [15:0]            filter_out;
          wire                              ro_filter_ovf_flag;

          filter_round_truncate uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .acc_in(acc_in),
                    .rf_sat(rf_sat),
                    .rf_shift(rf_shift),
                    .trig_filter_ovf_flag_clear(trig_filter_ovf_flag_clear),
                    .filter_out(filter_out),
                    .ro_filter_ovf_flag(ro_filter_ovf_flag)
          );

          initial begin
                    // Initialize Inputs
                    clk = 0;
                    rst_n = 0;
                    acc_in = 0;
                    rf_sat = 0;
                    rf_shift = 0;

                    #100;
                    rst_n = 1;
                    rf_shift = 3'b000;
        acc_in = 40'h0000000001;
                    #100;
                    rf_sat = 1'b1;


          end

          always
          begin
                    forever #5 clk = ~clk;
          end

endmodule
```

filter_stm_tf.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             filter_stm_tf.v
// Create Date:             ???
// Last Modification:       3/25/2016
// Author:                  Dhruvit Naik
// Description:             ????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module filter_stm_tf;

          // Inputs
          reg                               clk;
          reg                               rst_n;
          reg                               filter_aud_in_rts;
          reg              [31:0]           filter_aud_in;
          reg              [15:0]           rf_filter_coeff;

          // Outputs
          wire                              filter_aud_in_rtr;
          wire                              do_transfer;
          wire                              do_multiply_1st;
          wire                              do_multiply;
    wire              [39:0]          filter_aud_out;

          filter_stm uut (
    .clk(clk),


    .rst_n(rst_n),


    .filter_aud_in_rts(filter_aud_in_rts),
    .filter_aud_in_rtr(filter_aud_in_rtr),
    .filter_aud_out_rts(filter_aud_out_rts),
    .filter_aud_out_rtr(filter_aud_out_rtr),
    .filter_aud_in(filter_aud_in),

    .accumulator_load(accumulator_load),
    .accumulator_enable(accumulator_enable),
    .accumulator_in_left(accumulator_in_left),
    .accumulator_in_right(accumulator_in_right),
    .rf_filter_coeff(rf_filter_coeff),
    .mux_re(mux_re),



.mux_rdptr(mux_rdptr)

    );

          initial begin
                    // Initialize Inputs
                    clk = 0;
                    rst_n = 0;

                    #100;

          end

          always
    begin
        forever #5 clk = ~clk;
    end

endmodule
```

i2c_reg_test_automated.v:

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   13:57:50 02/03/2016
// Design Name:   i2c_reg_test
// Module Name:   H:/GitHub/Chip-Design/proj_asic/rtl/i2c_reg_test/i2c_reg_test_automated.v
// Project Name:  i2c_reg_test
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: i2c_reg_test
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module i2c_reg_test_automated;

	// Inputs
	reg clk;
	reg reset;
	reg i2c_scl;
	reg i2c_sda_in;
	reg [2:0] i2c_addr_bits;

	// Outputs
	wire i2c_sda_out;

	//INTERNAL VARIABLES

		reg [3:0] bit_count;
		reg [2:0] master_state;
		reg [9:0] data_byte;
		reg slave_acknowledgement;
		reg [7:0] test_data [1023:0];
		reg [11:0] reg_addr_test_data;
		reg [7:0] slave_addr_test_data;
		reg [6:0] count;

	// Instantiate the Unit Under Test (UUT)
	i2c_reg_test uut (
		.clk(clk),
		.reset(reset),
		.i2c_scl(i2c_scl),
		.i2c_sda_in(i2c_sda_in),
		.i2c_sda_out(i2c_sda_out),
		.i2c_addr_bits(i2c_addr_bits)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
		reset = 1;
		i2c_scl = 1;
		i2c_sda_in = 1;
		i2c_addr_bits = 3'b101;

		//test data
		test_data[0] = 8'h0;
		test_data[1] = 8'h1;
```

```
test_data[2] = 8'h2;
test_data[3] = 8'h3;
test_data[4] = 8'h4;
test_data[5] = 8'h5;
test_data[6] = 8'h6;
test_data[7] = 8'h7;
test_data[8] = 8'h8;
test_data[9] = 8'h9;
test_data[10] = 8'h0a;
test_data[11] = 8'h0b;
test_data[12] = 8'hc;
test_data[13] = 8'hd;
test_data[14] = 8'he;
test_data[15] = 8'hf;
test_data[16] = 8'h10;
test_data[17] = 8'h11;
test_data[18] = 8'h12;
test_data[19] = 8'h13;
test_data[20] = 8'h14;
test_data[21] = 8'h15;
test_data[22] = 8'h16;
test_data[23] = 8'h17;
test_data[24] = 8'h18;
test_data[25] = 8'h19;
test_data[26] = 8'h1a;
test_data[27] = 8'h1b;
test_data[28] = 8'h1c;
test_data[29] = 8'h1d;
test_data[30] = 8'h1e;
test_data[31] = 8'h1f;
test_data[32] = 8'h20;
test_data[33] = 8'h21;
test_data[34] = 8'h22;
test_data[35] = 8'h23;
test_data[36] = 8'h24;
test_data[37] = 8'h25;
test_data[38] = 8'h26;
test_data[39] = 8'h27;
test_data[40] = 8'h28;
test_data[41] = 8'h29;
test_data[42] = 8'h2a;
test_data[43] = 8'h2b;
test_data[44] = 8'h2c;
test_data[45] = 8'h2d;
test_data[46] = 8'h2e;
test_data[47] = 8'h2f;
test_data[48] = 8'h30;
test_data[49] = 8'h31;
test_data[50] = 8'h32;
test_data[51] = 8'h33;
test_data[52] = 8'h34;
test_data[53] = 8'h35;
test_data[54] = 8'h36;
test_data[55] = 8'h37;
test_data[56] = 8'h38;
test_data[57] = 8'h39;
test_data[58] = 8'h3a;
test_data[59] = 8'h3b;
test_data[60] = 8'h3c;
test_data[61] = 8'h3d;
test_data[62] = 8'h3e;
test_data[63] = 8'h3f;
test_data[64] = 8'h40;
test_data[65] = 8'h41;
test_data[66] = 8'h42;
test_data[67] = 8'h43;
test_data[68] = 8'h44;
test_data[69] = 8'h45;
test_data[70] = 8'h46;
test_data[71] = 8'h47;
test_data[72] = 8'h48;
```

```
test_data[73] = 8'h49;
test_data[74] = 8'h4a;
test_data[75] = 8'h4b;
test_data[76] = 8'h4c;
test_data[77] = 8'h4d;
test_data[78] = 8'h4e;
test_data[79] = 8'h4f;
test_data[80] = 8'h50;
test_data[81] = 8'h51;
test_data[82] = 8'h52;
test_data[83] = 8'h53;
test_data[84] = 8'h54;
test_data[85] = 8'h55;
test_data[86] = 8'h56;
test_data[87] = 8'h57;
test_data[88] = 8'h58;
test_data[89] = 8'h59;
test_data[90] = 8'h5a;
test_data[91] = 8'h5b;
test_data[92] = 8'h5c;
test_data[93] = 8'h5d;
test_data[94] = 8'h5e;
test_data[95] = 8'h5f;
test_data[96] = 8'h60;
test_data[97] = 8'h61;
test_data[98] = 8'h62;
test_data[99] = 8'h63;
test_data[100] = 8'h64;
test_data[101] = 8'h65;
test_data[102] = 8'h66;
test_data[103] = 8'h67;
test_data[104] = 8'h68;
test_data[105] = 8'h69;
test_data[106] = 8'h6a;
test_data[107] = 8'h6b;
test_data[108] = 8'h6c;
test_data[109] = 8'h6d;
test_data[110] = 8'h6e;
test_data[111] = 8'h6f;
test_data[112] = 8'h70;
test_data[113] = 8'h71;
test_data[114] = 8'h72;
test_data[115] = 8'h73;
test_data[116] = 8'h74;
test_data[117] = 8'h75;
test_data[118] = 8'h76;
test_data[119] = 8'h77;
test_data[120] = 8'h78;
test_data[121] = 8'h79;
test_data[122] = 8'h7a;
test_data[123] = 8'h7b;
test_data[124] = 8'h7c;
test_data[125] = 8'h7d;
test_data[126] = 8'h7e;
test_data[127] = 8'h7f;
test_data[128] = 8'h80;
test_data[129] = 8'h81;
test_data[130] = 8'h82;
test_data[131] = 8'h83;
test_data[132] = 8'h84;
test_data[133] = 8'h85;
test_data[134] = 8'h86;
test_data[135] = 8'h87;
test_data[136] = 8'h88;
test_data[137] = 8'h89;
test_data[138] = 8'h8a;
test_data[139] = 8'h8b;
test_data[140] = 8'h8c;
test_data[141] = 8'h8d;
test_data[142] = 8'h8e;
test_data[143] = 8'h8f;
```

```
test_data[144] = 8'h90;
test_data[145] = 8'h91;
test_data[146] = 8'h92;
test_data[147] = 8'h93;
test_data[148] = 8'h94;
test_data[149] = 8'h95;
test_data[150] = 8'h96;
test_data[151] = 8'h97;
test_data[152] = 8'h98;
test_data[153] = 8'h99;
test_data[154] = 8'h9a;
test_data[155] = 8'h9b;
test_data[156] = 8'h9c;
test_data[157] = 8'h9d;
test_data[158] = 8'h9e;
test_data[159] = 8'h9f;
test_data[160] = 8'ha0;
test_data[161] = 8'ha1;
test_data[162] = 8'ha2;
test_data[163] = 8'ha3;
test_data[164] = 8'ha4;
test_data[165] = 8'ha5;
test_data[166] = 8'ha6;
test_data[167] = 8'ha7;
test_data[168] = 8'ha8;
test_data[169] = 8'ha9;
test_data[170] = 8'haa;
test_data[171] = 8'hab;
test_data[172] = 8'hac;
test_data[173] = 8'had;
test_data[174] = 8'hae;
test_data[175] = 8'haf;
test_data[176] = 8'hb0;
test_data[177] = 8'hb1;
test_data[178] = 8'hb2;
test_data[179] = 8'hb3;
test_data[180] = 8'hb4;
test_data[181] = 8'hb5;
test_data[182] = 8'hb6;
test_data[183] = 8'hb7;
test_data[184] = 8'hb8;
test_data[185] = 8'hb9;
test_data[186] = 8'hba;
test_data[187] = 8'hbb;
test_data[188] = 8'hbc;
test_data[189] = 8'hbd;
test_data[190] = 8'hbe;
test_data[191] = 8'hbf;
test_data[192] = 8'hc0;
test_data[193] = 8'hc1;
test_data[194] = 8'hc2;
test_data[195] = 8'hc3;
test_data[196] = 8'hc4;
test_data[197] = 8'hc5;
test_data[198] = 8'hc6;
test_data[199] = 8'hc7;
test_data[200] = 8'hc8;
test_data[201] = 8'hc9;
test_data[202] = 8'hca;
test_data[203] = 8'hcb;
test_data[204] = 8'hcc;
test_data[205] = 8'hcd;
test_data[206] = 8'hce;
test_data[207] = 8'hcf;
test_data[208] = 8'hd0;
test_data[209] = 8'hd1;
test_data[210] = 8'hd2;
test_data[211] = 8'hd3;
test_data[212] = 8'hd4;
test_data[213] = 8'hd5;
test_data[214] = 8'hd6;
```

```
test_data[215] = 8'hd7;
test_data[216] = 8'hd8;
test_data[217] = 8'hd9;
test_data[218] = 8'hda;
test_data[219] = 8'hdb;
test_data[220] = 8'hdc;
test_data[221] = 8'hdd;
test_data[222] = 8'hde;
test_data[223] = 8'hdf;
test_data[224] = 8'he0;
test_data[225] = 8'he1;
test_data[226] = 8'he2;
test_data[227] = 8'he3;
test_data[228] = 8'he4;
test_data[229] = 8'he5;
test_data[230] = 8'he6;
test_data[231] = 8'he7;
test_data[232] = 8'he8;
test_data[233] = 8'he9;
test_data[234] = 8'hea;
test_data[235] = 8'heb;
test_data[236] = 8'hec;
test_data[237] = 8'hed;
test_data[238] = 8'hee;
test_data[239] = 8'hef;
test_data[240] = 8'hf0;
test_data[241] = 8'hf1;
test_data[242] = 8'hf2;
test_data[243] = 8'hf3;
test_data[244] = 8'hf4;
test_data[245] = 8'hf5;
test_data[246] = 8'hf6;
test_data[247] = 8'hf7;
test_data[248] = 8'hf8;
test_data[249] = 8'hf9;
test_data[250] = 8'hfa;
test_data[251] = 8'hfb;
test_data[252] = 8'hfc;
test_data[253] = 8'hfd;
test_data[254] = 8'hfe;
test_data[255] = 8'hff;
test_data[256] = 8'h0;
test_data[257] = 8'h1;
test_data[258] = 8'h2;
test_data[259] = 8'h3;
test_data[260] = 8'h4;
test_data[261] = 8'h5;
test_data[262] = 8'h6;
test_data[263] = 8'h7;
test_data[264] = 8'h8;
test_data[265] = 8'h9;
test_data[266] = 8'ha;
test_data[267] = 8'hb;
test_data[268] = 8'hc;
test_data[269] = 8'hd;
test_data[270] = 8'he;
test_data[271] = 8'hf;
test_data[272] = 8'h10;
test_data[273] = 8'h11;
test_data[274] = 8'h12;
test_data[275] = 8'h13;
test_data[276] = 8'h14;
test_data[277] = 8'h15;
test_data[278] = 8'h16;
test_data[279] = 8'h17;
test_data[280] = 8'h18;
test_data[281] = 8'h19;
test_data[282] = 8'h1a;
test_data[283] = 8'h1b;
test_data[284] = 8'h1c;
test_data[285] = 8'h1d;
```

```
test_data[286] = 8'h1e;
test_data[287] = 8'h1f;
test_data[288] = 8'h20;
test_data[289] = 8'h21;
test_data[290] = 8'h22;
test_data[291] = 8'h23;
test_data[292] = 8'h24;
test_data[293] = 8'h25;
test_data[294] = 8'h26;
test_data[295] = 8'h27;
test_data[296] = 8'h28;
test_data[297] = 8'h29;
test_data[298] = 8'h2a;
test_data[299] = 8'h2b;
test_data[300] = 8'h2c;
test_data[301] = 8'h2d;
test_data[302] = 8'h2e;
test_data[303] = 8'h2f;
test_data[304] = 8'h30;
test_data[305] = 8'h31;
test_data[306] = 8'h32;
test_data[307] = 8'h33;
test_data[308] = 8'h34;
test_data[309] = 8'h35;
test_data[310] = 8'h36;
test_data[311] = 8'h37;
test_data[312] = 8'h38;
test_data[313] = 8'h39;
test_data[314] = 8'h3a;
test_data[315] = 8'h3b;
test_data[316] = 8'h3c;
test_data[317] = 8'h3d;
test_data[318] = 8'h3e;
test_data[319] = 8'h3f;
test_data[320] = 8'h40;
test_data[321] = 8'h41;
test_data[322] = 8'h42;
test_data[323] = 8'h43;
test_data[324] = 8'h44;
test_data[325] = 8'h45;
test_data[326] = 8'h46;
test_data[327] = 8'h47;
test_data[328] = 8'h48;
test_data[329] = 8'h49;
test_data[330] = 8'h4a;
test_data[331] = 8'h4b;
test_data[332] = 8'h4c;
test_data[333] = 8'h4d;
test_data[334] = 8'h4e;
test_data[335] = 8'h4f;
test_data[336] = 8'h50;
test_data[337] = 8'h51;
test_data[338] = 8'h52;
test_data[339] = 8'h53;
test_data[340] = 8'h54;
test_data[341] = 8'h55;
test_data[342] = 8'h56;
test_data[343] = 8'h57;
test_data[344] = 8'h58;
test_data[345] = 8'h59;
test_data[346] = 8'h5a;
test_data[347] = 8'h5b;
test_data[348] = 8'h5c;
test_data[349] = 8'h5d;
test_data[350] = 8'h5e;
test_data[351] = 8'h5f;
test_data[352] = 8'h60;
test_data[353] = 8'h61;
test_data[354] = 8'h62;
test_data[355] = 8'h63;
test_data[356] = 8'h64;
```

```
test_data[357] = 8'h65;
test_data[358] = 8'h66;
test_data[359] = 8'h67;
test_data[360] = 8'h68;
test_data[361] = 8'h69;
test_data[362] = 8'h6a;
test_data[363] = 8'h6b;
test_data[364] = 8'h6c;
test_data[365] = 8'h6d;
test_data[366] = 8'h6e;
test_data[367] = 8'h6f;
test_data[368] = 8'h70;
test_data[369] = 8'h71;
test_data[370] = 8'h72;
test_data[371] = 8'h73;
test_data[372] = 8'h74;
test_data[373] = 8'h75;
test_data[374] = 8'h76;
test_data[375] = 8'h77;
test_data[376] = 8'h78;
test_data[377] = 8'h79;
test_data[378] = 8'h7a;
test_data[379] = 8'h7b;
test_data[380] = 8'h7c;
test_data[381] = 8'h7d;
test_data[382] = 8'h7e;
test_data[383] = 8'h7f;
test_data[384] = 8'h80;
test_data[385] = 8'h81;
test_data[386] = 8'h82;
test_data[387] = 8'h83;
test_data[388] = 8'h84;
test_data[389] = 8'h85;
test_data[390] = 8'h86;
test_data[391] = 8'h87;
test_data[392] = 8'h88;
test_data[393] = 8'h89;
test_data[394] = 8'h8a;
test_data[395] = 8'h8b;
test_data[396] = 8'h8c;
test_data[397] = 8'h8d;
test_data[398] = 8'h8e;
test_data[399] = 8'h8f;
test_data[400] = 8'h90;
test_data[401] = 8'h91;
test_data[402] = 8'h92;
test_data[403] = 8'h93;
test_data[404] = 8'h94;
test_data[405] = 8'h95;
test_data[406] = 8'h96;
test_data[407] = 8'h97;
test_data[408] = 8'h98;
test_data[409] = 8'h99;
test_data[410] = 8'h9a;
test_data[411] = 8'h9b;
test_data[412] = 8'h9c;
test_data[413] = 8'h9d;
test_data[414] = 8'h9e;
test_data[415] = 8'h9f;
test_data[416] = 8'ha0;
test_data[417] = 8'ha1;
test_data[418] = 8'ha2;
test_data[419] = 8'ha3;
test_data[420] = 8'ha4;
test_data[421] = 8'ha5;
test_data[422] = 8'ha6;
test_data[423] = 8'ha7;
test_data[424] = 8'ha8;
test_data[425] = 8'ha9;
test_data[426] = 8'haa;
test_data[427] = 8'hab;
```

```
test_data[428] = 8'hac;
test_data[429] = 8'had;
test_data[430] = 8'hae;
test_data[431] = 8'haf;
test_data[432] = 8'hb0;
test_data[433] = 8'hb1;
test_data[434] = 8'hb2;
test_data[435] = 8'hb3;
test_data[436] = 8'hb4;
test_data[437] = 8'hb5;
test_data[438] = 8'hb6;
test_data[439] = 8'hb7;
test_data[440] = 8'hb8;
test_data[441] = 8'hb9;
test_data[442] = 8'hba;
test_data[443] = 8'hbb;
test_data[444] = 8'hbc;
test_data[445] = 8'hbd;
test_data[446] = 8'hbe;
test_data[447] = 8'hbf;
test_data[448] = 8'hc0;
test_data[449] = 8'hc1;
test_data[450] = 8'hc2;
test_data[451] = 8'hc3;
test_data[452] = 8'hc4;
test_data[453] = 8'hc5;
test_data[454] = 8'hc6;
test_data[455] = 8'hc7;
test_data[456] = 8'hc8;
test_data[457] = 8'hc9;
test_data[458] = 8'hca;
test_data[459] = 8'hcb;
test_data[460] = 8'hcc;
test_data[461] = 8'hcd;
test_data[462] = 8'hce;
test_data[463] = 8'hcf;
test_data[464] = 8'hd0;
test_data[465] = 8'hd1;
test_data[466] = 8'hd2;
test_data[467] = 8'hd3;
test_data[468] = 8'hd4;
test_data[469] = 8'hd5;
test_data[470] = 8'hd6;
test_data[471] = 8'hd7;
test_data[472] = 8'hd8;
test_data[473] = 8'hd9;
test_data[474] = 8'hda;
test_data[475] = 8'hdb;
test_data[476] = 8'hdc;
test_data[477] = 8'hdd;
test_data[478] = 8'hde;
test_data[479] = 8'hdf;
test_data[480] = 8'he0;
test_data[481] = 8'he1;
test_data[482] = 8'he2;
test_data[483] = 8'he3;
test_data[484] = 8'he4;
test_data[485] = 8'he5;
test_data[486] = 8'he6;
test_data[487] = 8'he7;
test_data[488] = 8'he8;
test_data[489] = 8'he9;
test_data[490] = 8'hea;
test_data[491] = 8'heb;
test_data[492] = 8'hec;
test_data[493] = 8'hed;
test_data[494] = 8'hee;
test_data[495] = 8'hef;
test_data[496] = 8'hf0;
test_data[497] = 8'hf1;
test_data[498] = 8'hf2;
```

```
test_data[499] = 8'hf3;
test_data[500] = 8'hf4;
test_data[501] = 8'hf5;
test_data[502] = 8'hf6;
test_data[503] = 8'hf7;
test_data[504] = 8'hf8;
test_data[505] = 8'hf9;
test_data[506] = 8'hfa;
test_data[507] = 8'hfb;
test_data[508] = 8'hfc;
test_data[509] = 8'hfd;
test_data[510] = 8'hfe;
test_data[511] = 8'hff;
test_data[512] = 8'h0;
test_data[513] = 8'h1;
test_data[514] = 8'h2;
test_data[515] = 8'h3;
test_data[516] = 8'h4;
test_data[517] = 8'h5;
test_data[518] = 8'h6;
test_data[519] = 8'h7;
test_data[520] = 8'h8;
test_data[521] = 8'h9;
test_data[522] = 8'ha;
test_data[523] = 8'hb;
test_data[524] = 8'hc;
test_data[525] = 8'hd;
test_data[526] = 8'he;
test_data[527] = 8'hf;
test_data[528] = 8'h10;
test_data[529] = 8'h11;
test_data[530] = 8'h12;
test_data[531] = 8'h13;
test_data[532] = 8'h14;
test_data[533] = 8'h15;
test_data[534] = 8'h16;
test_data[535] = 8'h17;
test_data[536] = 8'h18;
test_data[537] = 8'h19;
test_data[538] = 8'h1a;
test_data[539] = 8'h1b;
test_data[540] = 8'h1c;
test_data[541] = 8'h1d;
test_data[542] = 8'h1e;
test_data[543] = 8'h1f;
test_data[544] = 8'h20;
test_data[545] = 8'h21;
test_data[546] = 8'h22;
test_data[547] = 8'h23;
test_data[548] = 8'h24;
test_data[549] = 8'h25;
test_data[550] = 8'h26;
test_data[551] = 8'h27;
test_data[552] = 8'h28;
test_data[553] = 8'h29;
test_data[554] = 8'h2a;
test_data[555] = 8'h2b;
test_data[556] = 8'h2c;
test_data[557] = 8'h2d;
test_data[558] = 8'h2e;
test_data[559] = 8'h2f;
test_data[560] = 8'h30;
test_data[561] = 8'h31;
test_data[562] = 8'h32;
test_data[563] = 8'h33;
test_data[564] = 8'h34;
test_data[565] = 8'h35;
test_data[566] = 8'h36;
test_data[567] = 8'h37;
test_data[568] = 8'h38;
test_data[569] = 8'h39;
```

```
test_data[570] = 8'h3a;
test_data[571] = 8'h3b;
test_data[572] = 8'h3c;
test_data[573] = 8'h3d;
test_data[574] = 8'h3e;
test_data[575] = 8'h3f;
test_data[576] = 8'h40;
test_data[577] = 8'h41;
test_data[578] = 8'h42;
test_data[579] = 8'h43;
test_data[580] = 8'h44;
test_data[581] = 8'h45;
test_data[582] = 8'h46;
test_data[583] = 8'h47;
test_data[584] = 8'h48;
test_data[585] = 8'h49;
test_data[586] = 8'h4a;
test_data[587] = 8'h4b;
test_data[588] = 8'h4c;
test_data[589] = 8'h4d;
test_data[590] = 8'h4e;
test_data[591] = 8'h4f;
test_data[592] = 8'h50;
test_data[593] = 8'h51;
test_data[594] = 8'h52;
test_data[595] = 8'h53;
test_data[596] = 8'h54;
test_data[597] = 8'h55;
test_data[598] = 8'h56;
test_data[599] = 8'h57;
test_data[600] = 8'h58;
test_data[601] = 8'h59;
test_data[602] = 8'h5a;
test_data[603] = 8'h5b;
test_data[604] = 8'h5c;
test_data[605] = 8'h5d;
test_data[606] = 8'h5e;
test_data[607] = 8'h5f;
test_data[608] = 8'h60;
test_data[609] = 8'h61;
test_data[610] = 8'h62;
test_data[611] = 8'h63;
test_data[612] = 8'h64;
test_data[613] = 8'h65;
test_data[614] = 8'h66;
test_data[615] = 8'h67;
test_data[616] = 8'h68;
test_data[617] = 8'h69;
test_data[618] = 8'h6a;
test_data[619] = 8'h6b;
test_data[620] = 8'h6c;
test_data[621] = 8'h6d;
test_data[622] = 8'h6e;
test_data[623] = 8'h6f;
test_data[624] = 8'h70;
test_data[625] = 8'h71;
test_data[626] = 8'h72;
test_data[627] = 8'h73;
test_data[628] = 8'h74;
test_data[629] = 8'h75;
test_data[630] = 8'h76;
test_data[631] = 8'h77;
test_data[632] = 8'h78;
test_data[633] = 8'h79;
test_data[634] = 8'h7a;
test_data[635] = 8'h7b;
test_data[636] = 8'h7c;
test_data[637] = 8'h7d;
test_data[638] = 8'h7e;
test_data[639] = 8'h7f;
test_data[640] = 8'h80;
```

```
test_data[641] = 8'h81;
test_data[642] = 8'h82;
test_data[643] = 8'h83;
test_data[644] = 8'h84;
test_data[645] = 8'h85;
test_data[646] = 8'h86;
test_data[647] = 8'h87;
test_data[648] = 8'h88;
test_data[649] = 8'h89;
test_data[650] = 8'h8a;
test_data[651] = 8'h8b;
test_data[652] = 8'h8c;
test_data[653] = 8'h8d;
test_data[654] = 8'h8e;
test_data[655] = 8'h8f;
test_data[656] = 8'h90;
test_data[657] = 8'h91;
test_data[658] = 8'h92;
test_data[659] = 8'h93;
test_data[660] = 8'h94;
test_data[661] = 8'h95;
test_data[662] = 8'h96;
test_data[663] = 8'h97;
test_data[664] = 8'h98;
test_data[665] = 8'h99;
test_data[666] = 8'h9a;
test_data[667] = 8'h9b;
test_data[668] = 8'h9c;
test_data[669] = 8'h9d;
test_data[670] = 8'h9e;
test_data[671] = 8'h9f;
test_data[672] = 8'ha0;
test_data[673] = 8'ha1;
test_data[674] = 8'ha2;
test_data[675] = 8'ha3;
test_data[676] = 8'ha4;
test_data[677] = 8'ha5;
test_data[678] = 8'ha6;
test_data[679] = 8'ha7;
test_data[680] = 8'ha8;
test_data[681] = 8'ha9;
test_data[682] = 8'haa;
test_data[683] = 8'hab;
test_data[684] = 8'hac;
test_data[685] = 8'had;
test_data[686] = 8'hae;
test_data[687] = 8'haf;
test_data[688] = 8'hb0;
test_data[689] = 8'hb1;
test_data[690] = 8'hb2;
test_data[691] = 8'hb3;
test_data[692] = 8'hb4;
test_data[693] = 8'hb5;
test_data[694] = 8'hb6;
test_data[695] = 8'hb7;
test_data[696] = 8'hb8;
test_data[697] = 8'hb9;
test_data[698] = 8'hba;
test_data[699] = 8'hbb;
test_data[700] = 8'hbc;
test_data[701] = 8'hbd;
test_data[702] = 8'hbe;
test_data[703] = 8'hbf;
test_data[704] = 8'hc0;
test_data[705] = 8'hc1;
test_data[706] = 8'hc2;
test_data[707] = 8'hc3;
test_data[708] = 8'hc4;
test_data[709] = 8'hc5;
test_data[710] = 8'hc6;
test_data[711] = 8'hc7;
```

```
test_data[712] = 8'hc8;
test_data[713] = 8'hc9;
test_data[714] = 8'hca;
test_data[715] = 8'hcb;
test_data[716] = 8'hcc;
test_data[717] = 8'hcd;
test_data[718] = 8'hce;
test_data[719] = 8'hcf;
test_data[720] = 8'hd0;
test_data[721] = 8'hd1;
test_data[722] = 8'hd2;
test_data[723] = 8'hd3;
test_data[724] = 8'hd4;
test_data[725] = 8'hd5;
test_data[726] = 8'hd6;
test_data[727] = 8'hd7;
test_data[728] = 8'hd8;
test_data[729] = 8'hd9;
test_data[730] = 8'hda;
test_data[731] = 8'hdb;
test_data[732] = 8'hdc;
test_data[733] = 8'hdd;
test_data[734] = 8'hde;
test_data[735] = 8'hdf;
test_data[736] = 8'he0;
test_data[737] = 8'he1;
test_data[738] = 8'he2;
test_data[739] = 8'he3;
test_data[740] = 8'he4;
test_data[741] = 8'he5;
test_data[742] = 8'he6;
test_data[743] = 8'he7;
test_data[744] = 8'he8;
test_data[745] = 8'he9;
test_data[746] = 8'hea;
test_data[747] = 8'heb;
test_data[748] = 8'hec;
test_data[749] = 8'hed;
test_data[750] = 8'hee;
test_data[751] = 8'hef;
test_data[752] = 8'hf0;
test_data[753] = 8'hf1;
test_data[754] = 8'hf2;
test_data[755] = 8'hf3;
test_data[756] = 8'hf4;
test_data[757] = 8'hf5;
test_data[758] = 8'hf6;
test_data[759] = 8'hf7;
test_data[760] = 8'hf8;
test_data[761] = 8'hf9;
test_data[762] = 8'hfa;
test_data[763] = 8'hfb;
test_data[764] = 8'hfc;
test_data[765] = 8'hfd;
test_data[766] = 8'hfe;
test_data[767] = 8'hff;
test_data[768] = 8'h0;
test_data[769] = 8'h1;
test_data[770] = 8'h2;
test_data[771] = 8'h3;
test_data[772] = 8'h4;
test_data[773] = 8'h5;
test_data[774] = 8'h6;
test_data[775] = 8'h7;
test_data[776] = 8'h8;
test_data[777] = 8'h9;
test_data[778] = 8'ha;
test_data[779] = 8'hb;
test_data[780] = 8'hc;
test_data[781] = 8'hd;
test_data[782] = 8'he;
```

```
test_data[783] = 8'hf;
test_data[784] = 8'h10;
test_data[785] = 8'h11;
test_data[786] = 8'h12;
test_data[787] = 8'h13;
test_data[788] = 8'h14;
test_data[789] = 8'h15;
test_data[790] = 8'h16;
test_data[791] = 8'h17;
test_data[792] = 8'h18;
test_data[793] = 8'h19;
test_data[794] = 8'h1a;
test_data[795] = 8'h1b;
test_data[796] = 8'h1c;
test_data[797] = 8'h1d;
test_data[798] = 8'h1e;
test_data[799] = 8'h1f;
test_data[800] = 8'h20;
test_data[801] = 8'h21;
test_data[802] = 8'h22;
test_data[803] = 8'h23;
test_data[804] = 8'h24;
test_data[805] = 8'h25;
test_data[806] = 8'h26;
test_data[807] = 8'h27;
test_data[808] = 8'h28;
test_data[809] = 8'h29;
test_data[810] = 8'h2a;
test_data[811] = 8'h2b;
test_data[812] = 8'h2c;
test_data[813] = 8'h2d;
test_data[814] = 8'h2e;
test_data[815] = 8'h2f;
test_data[816] = 8'h30;
test_data[817] = 8'h31;
test_data[818] = 8'h32;
test_data[819] = 8'h33;
test_data[820] = 8'h34;
test_data[821] = 8'h35;
test_data[822] = 8'h36;
test_data[823] = 8'h37;
test_data[824] = 8'h38;
test_data[825] = 8'h39;
test_data[826] = 8'h3a;
test_data[827] = 8'h3b;
test_data[828] = 8'h3c;
test_data[829] = 8'h3d;
test_data[830] = 8'h3e;
test_data[831] = 8'h3f;
test_data[832] = 8'h40;
test_data[833] = 8'h41;
test_data[834] = 8'h42;
test_data[835] = 8'h43;
test_data[836] = 8'h44;
test_data[837] = 8'h45;
test_data[838] = 8'h46;
test_data[839] = 8'h47;
test_data[840] = 8'h48;
test_data[841] = 8'h49;
test_data[842] = 8'h4a;
test_data[843] = 8'h4b;
test_data[844] = 8'h4c;
test_data[845] = 8'h4d;
test_data[846] = 8'h4e;
test_data[847] = 8'h4f;
test_data[848] = 8'h50;
test_data[849] = 8'h51;
test_data[850] = 8'h52;
test_data[851] = 8'h53;
test_data[852] = 8'h54;
test_data[853] = 8'h55;
```

```
test_data[854] = 8'h56;
test_data[855] = 8'h57;
test_data[856] = 8'h58;
test_data[857] = 8'h59;
test_data[858] = 8'h5a;
test_data[859] = 8'h5b;
test_data[860] = 8'h5c;
test_data[861] = 8'h5d;
test_data[862] = 8'h5e;
test_data[863] = 8'h5f;
test_data[864] = 8'h60;
test_data[865] = 8'h61;
test_data[866] = 8'h62;
test_data[867] = 8'h63;
test_data[868] = 8'h64;
test_data[869] = 8'h65;
test_data[870] = 8'h66;
test_data[871] = 8'h67;
test_data[872] = 8'h68;
test_data[873] = 8'h69;
test_data[874] = 8'h6a;
test_data[875] = 8'h6b;
test_data[876] = 8'h6c;
test_data[877] = 8'h6d;
test_data[878] = 8'h6e;
test_data[879] = 8'h6f;
test_data[880] = 8'h70;
test_data[881] = 8'h71;
test_data[882] = 8'h72;
test_data[883] = 8'h73;
test_data[884] = 8'h74;
test_data[885] = 8'h75;
test_data[886] = 8'h76;
test_data[887] = 8'h77;
test_data[888] = 8'h78;
test_data[889] = 8'h79;
test_data[890] = 8'h7a;
test_data[891] = 8'h7b;
test_data[892] = 8'h7c;
test_data[893] = 8'h7d;
test_data[894] = 8'h7e;
test_data[895] = 8'h7f;
test_data[896] = 8'h80;
test_data[897] = 8'h81;
test_data[898] = 8'h82;
test_data[899] = 8'h83;
test_data[900] = 8'h84;
test_data[901] = 8'h85;
test_data[902] = 8'h86;
test_data[903] = 8'h87;
test_data[904] = 8'h88;
test_data[905] = 8'h89;
test_data[906] = 8'h8a;
test_data[907] = 8'h8b;
test_data[908] = 8'h8c;
test_data[909] = 8'h8d;
test_data[910] = 8'h8e;
test_data[911] = 8'h8f;
test_data[912] = 8'h90;
test_data[913] = 8'h91;
test_data[914] = 8'h92;
test_data[915] = 8'h93;
test_data[916] = 8'h94;
test_data[917] = 8'h95;
test_data[918] = 8'h96;
test_data[919] = 8'h97;
test_data[920] = 8'h98;
test_data[921] = 8'h99;
test_data[922] = 8'h9a;
test_data[923] = 8'h9b;
test_data[924] = 8'h9c;
```

```
test_data[925] = 8'h9d;
test_data[926] = 8'h9e;
test_data[927] = 8'h9f;
test_data[928] = 8'ha0;
test_data[929] = 8'ha1;
test_data[930] = 8'ha2;
test_data[931] = 8'ha3;
test_data[932] = 8'ha4;
test_data[933] = 8'ha5;
test_data[934] = 8'ha6;
test_data[935] = 8'ha7;
test_data[936] = 8'ha8;
test_data[937] = 8'ha9;
test_data[938] = 8'haa;
test_data[939] = 8'hab;
test_data[940] = 8'hac;
test_data[941] = 8'had;
test_data[942] = 8'hae;
test_data[943] = 8'haf;
test_data[944] = 8'hb0;
test_data[945] = 8'hb1;
test_data[946] = 8'hb2;
test_data[947] = 8'hb3;
test_data[948] = 8'hb4;
test_data[949] = 8'hb5;
test_data[950] = 8'hb6;
test_data[951] = 8'hb7;
test_data[952] = 8'hb8;
test_data[953] = 8'hb9;
test_data[954] = 8'hba;
test_data[955] = 8'hbb;
test_data[956] = 8'hbc;
test_data[957] = 8'hbd;
test_data[958] = 8'hbe;
test_data[959] = 8'hbf;
test_data[960] = 8'hc0;
test_data[961] = 8'hc1;
test_data[962] = 8'hc2;
test_data[963] = 8'hc3;
test_data[964] = 8'hc4;
test_data[965] = 8'hc5;
test_data[966] = 8'hc6;
test_data[967] = 8'hc7;
test_data[968] = 8'hc8;
test_data[969] = 8'hc9;
test_data[970] = 8'hca;
test_data[971] = 8'hcb;
test_data[972] = 8'hcc;
test_data[973] = 8'hcd;
test_data[974] = 8'hce;
test_data[975] = 8'hcf;
test_data[976] = 8'hd0;
test_data[977] = 8'hd1;
test_data[978] = 8'hd2;
test_data[979] = 8'hd3;
test_data[980] = 8'hd4;
test_data[981] = 8'hd5;
test_data[982] = 8'hd6;
test_data[983] = 8'hd7;
test_data[984] = 8'hd8;
test_data[985] = 8'hd9;
test_data[986] = 8'hda;
test_data[987] = 8'hdb;
test_data[988] = 8'hdc;
test_data[989] = 8'hdd;
test_data[990] = 8'hde;
test_data[991] = 8'hdf;
test_data[992] = 8'he0;
test_data[993] = 8'he1;
test_data[994] = 8'he2;
test_data[995] = 8'he3;
```

```
                    test_data[996] = 8'he4;
                    test_data[997] = 8'he5;
                    test_data[998] = 8'he6;
                    test_data[999] = 8'he7;
                    test_data[1000] = 8'he8;
                    test_data[1001] = 8'he9;
                    test_data[1002] = 8'hea;
                    test_data[1003] = 8'heb;
                    test_data[1004] = 8'hec;
                    test_data[1005] = 8'hed;
                    test_data[1006] = 8'hee;
                    test_data[1007] = 8'hef;
                    test_data[1008] = 8'hf0;
                    test_data[1009] = 8'hf1;
                    test_data[1010] = 8'hf2;
                    test_data[1011] = 8'hf3;
                    test_data[1012] = 8'hf4;
                    test_data[1013] = 8'hf5;
                    test_data[1014] = 8'hf6;
                    test_data[1015] = 8'hf7;
                    test_data[1016] = 8'hf8;
                    test_data[1017] = 8'hf9;
                    test_data[1018] = 8'hfa;
                    test_data[1019] = 8'hfb;
                    test_data[1020] = 8'hfc;
                    test_data[1021] = 8'hfd;
                    test_data[1022] = 8'hfe;
                    test_data[1023] = 8'hff;

                    reg_addr_test_data = 11'b00000000001; //11'h400 = start of coefficient
registers
                    slave_addr_test_data = 8'b11010101;             //current slave address w/
3'b101 as i2c_addr_bits

                    #100
                    reset = 0;
                    #100
                    reset = 1;

          end

                    // Wait 100 ns for global reset to finish
                    //#100;

always #50 clk = !clk;
always #1250 i2c_scl = !i2c_scl;

always @ (posedge clk or negedge reset)
begin
                    count = count + 1;
          if(!reset)
          begin
          count <= 0;
          end
          if (count == 25)                       //reset count on full scl cycle
          begin
          count <= 0;
          end
end




always @ (posedge clk or negedge reset)
begin
        if(!reset)
        begin
            bit_count<=0;
            master_state<=3'b000;
                                        data_byte <= 0;
```

```
                                slave_acknowledgement <= 0;

end

 else if ((master_state == 3'b000) & (count == 25))  //start condition
 begin
 i2c_sda_in <= 0;
 master_state <= 3'b001;
 end

 else if ((master_state == 3'b001))    //serialize slave address and RW bit
 begin
           if(count == 19)
           begin
           bit_count <= bit_count + 1;
                     if (bit_count == 8)
                     begin
                     i2c_sda_in <= 1;
                     end
                     if (bit_count < 8)
                     begin
                     i2c_sda_in <= slave_addr_test_data [bit_count];
                     end
           end

           else if((count == 6) & (bit_count == 9))
           begin
           bit_count <= 0;
           slave_acknowledgement <= i2c_sda_out;
           end

           else if(slave_acknowledgement == 1)
           begin
           slave_acknowledgement <= 0;
           master_state <= 3'b010;
           end
 end

 else if ((master_state == 3'b010))    //serialize reg address part 1
 begin
           if(count == 19)
           begin
           bit_count <= bit_count + 1;
                     if (bit_count == 8)
                     begin
                     i2c_sda_in <= 1;
                     end
                     if (bit_count < 8)
                     begin
                     i2c_sda_in <= reg_addr_test_data [bit_count];
                     end
           end

           else if((count == 6) & (bit_count == 9))
           begin
           bit_count <= 0;
           slave_acknowledgement <= i2c_sda_out;
           end

           else if(slave_acknowledgement == 1)
           begin
           slave_acknowledgement <= 0;
           master_state <= 3'b011;
           end
 end

 else if ((master_state == 3'b011))    //serialize reg address part 2
 begin
           if(count == 19)
           begin
           bit_count <= bit_count + 1;
```

```
                        if (bit_count == 8)
                        begin
                        i2c_sda_in <= 1;
                        end
                        if (bit_count < 8)
                        begin
                        i2c_sda_in <= reg_addr_test_data [bit_count+8];
                        end
                end

                else if((count == 6) & (bit_count == 9))
                begin
                bit_count <= 0;
                slave_acknowledgement <= i2c_sda_out;
                end

                else if(slave_acknowledgement == 1)
                begin
                slave_acknowledgement <= 0;
                master_state <= 3'b100;
                end

        end

        else if ((master_state == 3'b100) & (data_byte < 1024) & (slave_addr_test_data [7] ==
1))     //serialize data
        begin
//              if(count == 19)
//              begin
//              i2c_sda_in <= test_data [data_byte] [bit_count];
//              bit_count <= bit_count + 1;
//                      if (bit_count == 9)
//                      begin
//                      i2c_sda_in <= 1;
//                      end
//              end
                if(count == 19)
                begin
                bit_count <= bit_count + 1;
                        if (bit_count == 8)
                        begin
                        i2c_sda_in <= 1;
                        end
                        if (bit_count < 8)
                        begin
                        i2c_sda_in <= test_data [data_byte] [bit_count];
                        end
                end

                else if((count == 6) & (bit_count == 9))
                begin
                bit_count <= 0;
                slave_acknowledgement <= i2c_sda_out;
                end

                else if(slave_acknowledgement == 1)
                begin
                slave_acknowledgement <= 0;
                data_byte <= data_byte + 1;
                end

                else if (data_byte == 1023)
                begin
                data_byte <= 0;
                master_state <= 3'b101;                 //enter stop condition
                slave_acknowledgement <= 0;
                bit_count <= 0;
                end
        end

        else if (master_state == 3'b101)                // stop condition
```

```
                    begin
                    if (count == 19)
                    begin
                    i2c_sda_in <= 0;
                    end

                    if (count == 3)
                    begin
                    i2c_sda_in <=1;
                    master_state <= 3'b110;
                    end
                    ////////////////////////////enter read algorithm
                    end


end

endmodule
```

i2c_reg_testbench_write_readback.v:

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   16:13:22 02/02/2016
// Design Name:   i2c_reg_test
// Module Name:   H:/GitHub/Chip-
Design/proj_asic/rtl/i2c_reg_test/i2c_reg_testbench_write_readback.v
// Project Name:  i2c_reg_test
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: i2c_reg_test
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module i2c_reg_testbench_write_readback;

          // Inputs
          reg clk;
          reg reset;
          reg i2c_scl;
          reg i2c_sda_in;
          reg [2:0] i2c_addr_bits;

          // Outputs
          wire i2c_sda_out;

          // Instantiate the Unit Under Test (UUT)
          i2c_reg_test uut (
                    .clk(clk),
                    .reset(reset),
                    .i2c_scl(i2c_scl),
                    .i2c_sda_in(i2c_sda_in),
                    .i2c_sda_out(i2c_sda_out),
                    .i2c_addr_bits(i2c_addr_bits)
          );

          initial begin
                    // Initialize Inputs
                    clk = 0;
                    reset = 0;
                    i2c_scl = 0;
                    i2c_sda_in = 0;
                    i2c_addr_bits = 0;

                    // Wait 100 ns for global reset to finish
                    #100;
                    end

                    always #50 clk = !clk;
                    always #1250 i2c_scl = !i2c_scl;
                    //always #200 i2c_sda = !i2c_sda;


                    initial begin
                    reset = 1;
                    #200
```

```
reset = 0;
#100
reset = 1;

i2c_scl = 1;
i2c_sda_in = 1;
i2c_addr_bits = 3'b101;

#2825
i2c_sda_in = 0; //Start Condition
#1500
i2c_sda_in = 1; //Slave Address Start
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 1; //RW Bit

#2500
i2c_sda_in = 1; //nothing
#2500
i2c_sda_in = 1;    //Data addr MSB [10]
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;    //Data Addr LSB [3]

#2500
i2c_sda_in = 0;    //nothing
#2500
i2c_sda_in = 0;    //Data addr MSB    [2]
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;    //Data addr LSB    [0]
#2500
i2c_sda_in = 1;    //Data Addr Part 2 MSB extra
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 1;   //Data Addr Part 2 LSB extra
#2500
i2c_sda_in = 1;    //nothing

#2500
i2c_sda_in = 1;    //Data MSB
#2500
i2c_sda_in = 0;
```

```
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0;    //Data LSB
#2500
i2c_sda_in = 0;    //nothing
#2500
i2c_sda_in = 0;
#500
i2c_sda_in = 1;

//#1250
//i2c_sda_in = 1;  //reset to high drive

//start read back
#2825
i2c_sda_in = 0; //Start Condition
#1500
i2c_sda_in = 1; //Slave Address Start
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 1;
#2500
i2c_sda_in = 0; //RW Bit

#2500
i2c_sda_in = 1; //nothing
#2500
i2c_sda_in = 1;    //Data addr MSB [10]
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;    //Data Addr LSB [3]

#2500
i2c_sda_in = 0;    //nothing
#2500
i2c_sda_in = 0;    //Data addr MSB    [2]
#2500
i2c_sda_in = 0;
#2500
i2c_sda_in = 0;    //Data addr LSB    [0]
#2500
i2c_sda_in = 1;    //Data Addr Part 2 MSB extra
```

```
                #2500
                i2c_sda_in = 1;
                #2500
                i2c_sda_in = 1;
                #2500
                i2c_sda_in = 1;
                #2500
                i2c_sda_in = 1;   //Data Addr Part 2 LSB extra
                #2500
                i2c_sda_in = 1;    //nothing

                #2500
                #1250
                i2c_sda_in = 1;    //stop condition


                #20000                                //Random reset press
                reset = 0;
                #10
                reset = 1;                   //reset unpress
                // Add stimulus here

        end

endmodule
```

bist_test.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   bist_test.v
// Create Date:   10/20/2015
// Last Edit:     3/20/16
// Author:        Zachary Nelson
//
// Description:     Creates 32 bit sawtooth wave
//                  Ensuring:
//                      BIST starts at correct value
//                      Increments by specified amount
//                      Resets to starting value when reaching upper limit value
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module bist_test;

    // Inputs
    reg                             clk;
    wire                            rst_n;
    wire                            sck_transition;
    reg         [11:0]              rf_bist_start_val;
    reg         [7:0]               rf_bist_inc;
    reg         [11:0]              rf_bist_up_limit;

    // Internal Variables
    reg                             i2si_sck;
    reg                             sck_d1;                             // serial clock delay
    reg         [31:0]              count;
    reg         [31:0]              sck_cnt;                            // serial clock counter
    reg         [31:0]              cyc_per_half_sck = 40;              // about (100 MHz / 1.44
MHz)/2


    // Outputs
    wire        [31:0]              i2si_bist_out_data;
    wire                            i2si_bist_out_xfc;

    // Instantiate the Unit Under Test (UUT)
    i2si_bist_gen uut (
        .clk(clk),
        .rst_n(rst_n),
        .sck_transition(sck_transition),
        .rf_bist_start_val(rf_bist_start_val),
        .rf_bist_inc(rf_bist_inc),
        .rf_bist_up_limit(rf_bist_up_limit),
        .i2si_bist_out_data(i2si_bist_out_data),
        .i2si_bist_out_xfc(i2si_bist_out_xfc)
    );

    initial
    begin
        // Initialize Inputs
        clk = 0;
        i2si_sck=0;
        rf_bist_start_val = 12'h001;
        rf_bist_inc = 12'h001;
        rf_bist_up_limit = 12'h019;
    end

    always
    begin
        count = 0;                                              // set clock counter to zero
    forever
        begin
            #5 clk = ~clk;                                      // 100 MHz clock
```

```
            count = count + 1;                                  // increment clock counter
        end
    end


    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt<=0;                                         // counts master clock cycles,
causes sck to toggle each time it hits cyc_per_half_sck
            i2si_sck<=0;                                        // serial clock
            sck_d1<=0;                                          // serial clock delayed by one
clock cycle
        end
        else
            sck_cnt <= sck_cnt + 1;                             // increment serial clock counter
        sck_d1<=i2si_sck;                                       // generate 1 cycle delay of
i2si_sck


        if (sck_cnt == cyc_per_half_sck-1)                  // cyc_per_half_sck ~ (100 MHz/1.44
MHz)/2
        begin
            sck_cnt <= 0;                                   // reset serial clock counter
            i2si_sck <= ~i2si_sck;                          // toggle serial clock
        end
    end


    assign sck_transition = i2si_sck & ~sck_d1;
    assign rst_n = !(count < 20);

endmodule
```

fifo_test.v:

```
`define BUF_WIDTH 3 // set the buffer width equal to 3
`define DATA_SIZE 32 // no. of bits for fifo data

module fifo_test();

    reg                         clk;
//Master clock
    reg                         rst_n;
//Reset
    reg                         fifo_inp_rts;
//Write enabled
    reg                         fifo_out_rtr;
//Read enabled
    reg[`DATA_SIZE-1:0]         fifo_inp_data;
//Buffer input
    reg[`DATA_SIZE-1:0]         tempdata;
//Temporary data
    wire [`DATA_SIZE-1:0]       fifo_out_data;
//Buffer output

    fifo ff( .clk(clk), .rst_n(rst_n), .fifo_inp_data(fifo_inp_data),
.fifo_out_data(fifo_out_data),
             .fifo_inp_rts(fifo_inp_rts), .fifo_out_rtr(fifo_out_rtr),
.fifo_out_rts(fifo_out_rts),
             .fifo_inp_rtr(fifo_inp_rtr));

    initial
    begin
       clk = 0;
// clock starts at false
       rst_n = 0;
// reset starts at true
       fifo_out_rtr = 0;
// read enabled starts at false
       fifo_inp_rts= 0;
// write enabled starts at false
       tempdata = 0;
// temporary data starts at zero
       fifo_inp_data = 0;
// buffer input is intially zero
        #15 rst_n = 1;
// after 15ns change the reset to false

       queue(1);
// queue the value of 1
       fork
// queue the value of 2 and dequeue at the same time
            queue(2);
            dequeue(tempdata);
       join

            queue(10);
// queue the value of 10
            queue(20);
// queue the value of 20
            queue(30);
// queue the value of 30
            queue(40);
// queue the value of 40
            queue(50);
// queue the value of 50
            queue(60);
// queue the value of 60
            queue(70);
// queue the value of 70
            queue(80);
// queue the value of 80
            queue(90);
// queue the value of 90
```

```
            queue(100);
// queue the value of 100
            queue(110);
// queue the value of 110
            queue(120);
// queue the value of 120
            queue(130);
// queue the value of 130

            dequeue(tempdata);
// dequeue
            queue(tempdata);
// queue the data that was just dequeued
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            queue(140);
// queue the value of 140
            dequeue(tempdata);
// dequeue
            queue(tempdata);
// queue the data that was just dequeued
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            dequeue(tempdata);
// dequeue
            queue(5);
// queue the value of 5
            dequeue(tempdata);
// dequeue
    end

    always
       #5 clk = ~clk;
// change the clock every 5ns

    task queue;
// define the queue task
    input[7:0] data;
// the data to be queued
       if( !fifo_inp_rtr )
// if buffer is full display warning
                $display("---Cannot queue: Buffer Full---");
            else
// if buffer is not full
            begin
               $display("Queued ",data );
// display that the data was queued
```

```
                fifo_inp_data = data;
// the input to the buffer is set as the data
                fifo_inp_rts = 1;
// write is enabled
                @(posedge clk);
// checks if clock is at postive edge
                #1 fifo_inp_rts = 0;
// set write enabled equal to zero then
            end
    endtask

    task dequeue;
// define the dequeue task
    output [7:0] data;
// the data to be dequeued
        if( !fifo_out_rts )
// if the buffer is empty display a warning
             $display("---Cannot dequeue: Buffer Empty---");
        else
// if buffer is not empty
            begin
                fifo_out_rtr = 1;
// read is enabled
             @(posedge clk);
// checks if clock is at postive edge
             #1 fifo_out_rtr = 0;
// set read enabled qual to zero then
             data = fifo_out_data;
// the data is set as the output of the buffer
             $display("Dequeued ", data);
// display that the data was queued
            end
    endtask

endmodule
```

deserializer_testbench.v:

```
`define N 11 // number of test elements

module i2si_deserializer_testbench;

   // Inputs
    reg                 clk;
// master clock
    wire                sck_transition;
// sck level to pulse converter
    reg                 rf_i2si_en;
// i2si enable
    reg [15:0]          test_data [`N-1:0] [0:1];
// [Bits Per Word] test_data [# of entities in test] [Left/Right]

    // Outputs
    wire [15:0]         out_lft;
// left audio dataF
    wire [15:0]         out_rgt;
// right audio data
    wire                out_xfc;
// transfer complete
    wire                rst_n;
// reset
    wire                in_sd;
// serial data
    wire                in_ws;
// word select

    // Internal Variables
    reg                 i2si_sck;
    reg                 sck_d1;
// serial clock delay
    reg [31:0]          count;
// clock counter
    reg [31:0]          sck_cnt;
// serial clock counter
    reg [31:0]          bit_cnt;
// bit number counter
    reg                 lr_cnt;
// left right counter
    reg [31:0]          word_cnt;
// word counter
    reg [31:0]          cyc_per_half_sck = 40;
// about (100 MHz / 1.44 MHz)/2
    reg [31:0]          bit_tc =  15;
// number of bits in a word

    // Instantiate the Unit Under Test (UUT)
    i2si_deserializer uut (
        .clk(clk),
        .rst_n(rst_n),
        .sck_transition(sck_transition),
        .in_ws(in_ws),
        .in_sd(in_sd),
        .rf_i2si_en(rf_i2si_en),
        .out_lft(out_lft),
        .out_rgt(out_rgt),
        .out_xfc(out_xfc)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        rf_i2si_en = 0;

        // Test Data
        test_data [0] [0] = 16'hAAAA;
        test_data [0] [1] = 16'hFFFF;
        test_data [1] [0] = 16'h1478;
```

```
        test_data [1] [1] = 16'hA3B9;
        test_data [2] [0] = 16'hCDD7;
        test_data [2] [1] = 16'hBABA;
        test_data [3] [0] = 16'h4444;
        test_data [3] [1] = 16'hAAAA;
        test_data [4] [0] = 16'h7398;
        test_data [4] [1] = 16'hFFDD;
        test_data [5] [0] = 16'h1111;
        test_data [5] [1] = 16'h5982;
        test_data [6] [0] = 16'h0001;
        test_data [6] [1] = 16'hFFFF;
        test_data [7] [0] = 16'h1478;
        test_data [7] [1] = 16'hA3B9;
        test_data [8] [0] = 16'hF8D5;
        test_data [8] [1] = 16'hD55A;
        test_data [9] [0] = 16'h99C5;
        test_data [9] [1] = 16'h7435;
        test_data [10] [0] = 16'h69D9;
        test_data [10] [1] = 16'hABCD;

        #694
        rf_i2si_en = 1;
// enable i2si after 694ns
    end

    always
    begin
        count = 0;
// set clock counter to zero
    forever
        begin
            #5 clk = ~clk;
// 100 MHz clock
            count = count + 1;
// increment clock counter
        end
    end

    assign rst_n = !(count < 20);
// turn on reset not after 10 clock cycles
    assign sck_transition = i2si_sck & ~sck_d1;
    assign in_ws = ((0<=bit_cnt& bit_cnt<=16'd14)&lr_cnt==1)|((bit_cnt==16'd15)&(lr_cnt==0));
    assign in_sd = test_data [word_cnt][lr_cnt][bit_tc-bit_cnt];
// assign serial data from the test_data

    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt<=0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            bit_cnt<=0;
// count number of bits
            word_cnt<=0;
// count the word number
            lr_cnt <= 0;
// left=0 and right=1
            i2si_sck<=0;
// serial clock
            sck_d1<=0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
```

```
                i2si_sck <= ~i2si_sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1<=i2si_sck;
// generate 1 cycle delay of i2si_sck
            if(i2si_sck & ~sck_d1)
// on a positive transition of sck...

            begin
                if (bit_cnt==bit_tc)
// bit_tc = 15
                begin
                    if (lr_cnt == 1)
// if right
                    begin
                        word_cnt<=word_cnt+1;
// words in the testbench array
                        lr_cnt<=0;
// set to left
                    end
                    else
                        lr_cnt<=1;
// set to right
                    bit_cnt<=0;
// reset bit counter
                end
                else
                    bit_cnt<=bit_cnt+1;
// increment bit counter
            end
        end
    end

endmodule
```

i2s_in_test.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   i2s_in_test.v
// Create Date:   11/27/2015
// Last Edit:     3/20/16
// Author:        Kevin Cao, Zachary Nelson
//
// Description:     Creates N number 32 bit words specified by the programmer to be inputted.
//                  Compares the inputted and outputted words.
//                  Outputs the success and failure of the comparisons in i2s_in_test_output.txt
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define N 11 // number of test elements

module i2s_in_test;

          // Inputs
          reg                               clk;
          wire                              rst_n;

          reg                               inp_sck;
          wire                              inp_ws;
          wire                              inp_sd;

          wire                              rf_i2si_en;
          reg     [11:0]                    rf_bist_start_val;
          reg     [ 7:0]                    rf_bist_inc;
          reg     [11:0]                    rf_bist_up_limit;
          reg                               rf_mux_en;

    wire                              i2si_rtr;

    reg                               trig_fifo_overrun_clr;

          // Outputs
          wire    [31:0]                    i2si_data;
          wire                              i2si_rts;
          wire                              ro_fifo_overrun;
          wire                              sync_sck;


    // Internal Variables
    reg                               sck_d1;
// serial clock delay
    integer                           count;
// clock counter
    integer                           sck_cnt;
// serial clock counter
    integer                           bit_cnt;
// bit number counter
    integer                           lr_cnt;
// left right counter
    integer                           word_cnt;
// word counter
    parameter                         cyc_per_half_sck = 33;
// about (100 MHz / 1.536 MHz)/2
    parameter                         bit_tc =  15;
// number of bits in a word


    reg     [15:0]                    test_data [`N-1:0] [0:1];
// [Bits Per Word] test_data [# of entities in test] [Left/Right]


    integer                           match_count = 0;
// Counter to help find initial matching words for word and i2si_data
    integer                           compare_count = 0;
// Counter to help compare word and i2si_data after intial match is found
```

```
    integer                         pass_count = 0;
// Counts number of successful comparisons
    integer                         fail_count = 0;
// Counts number of failed comparisons
    integer                         cycle_count = 0;
// Count to help keep track of how many i2si_rtr && i2si_rts transitions occurred
    reg                             match_found = 0;
// Boolean that determines if intial match was found
    reg                             begin_comparison = 0;
// Boolean that determines if to start comparing words after intial match is found
    reg                             test_failed = 1;
// Boolean that determines if no matches were found and the test failed
    reg                             skipped_two_cycles = 0;
// Boolean that determines if the first two transitions of i2si_rtr and i2si_rts was skipped
    reg     [31:0]                  word;
// Stores the expected word, to be compared with i2si_data

    integer                         out;
// Helps create output text file

          // Instantiate the Unit Under Test (UUT)
          i2s_in uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .inp_sck(inp_sck),
                    .inp_ws(inp_ws),
                    .inp_sd(inp_sd),
                    .rf_i2si_en(rf_i2si_en),
                    .rf_bist_start_val(rf_bist_start_val),
                    .rf_bist_inc(rf_bist_inc),
                    .rf_bist_up_limit(rf_bist_up_limit),
                    .rf_mux_en(rf_mux_en),
                    .i2si_rtr(i2si_rtr),
                    .i2si_data(i2si_data),
                    .i2si_rts(i2si_rts),
                    .ro_fifo_overrun(ro_fifo_overrun),
                    .trig_fifo_overrun_clr(trig_fifo_overrun_clr),
                    .sync_sck(sync_sck),
        .sync_sck_transition(sync_sck_transition)
          );

          initial begin
                    clk = 0;
                    inp_sck = 0;
        rf_bist_start_val = 12'd1;
// Set starting value to 1
        rf_bist_inc = 12'd1;
// Set increment value to 1
        rf_bist_up_limit = 12'd25;
// Set BIST upper limit to 25
                    rf_mux_en = 0;
// Disable multiplexer select bit, Output deserializer's contents
                    trig_fifo_overrun_clr = 0;


        out = $fopen("i2s_in_test_output.txt");
// Open i2si_in_test2_output.txt




        // Test Data

        test_data [ 0] [0] = 16'hAAAA;
        test_data [ 0] [1] = 16'hFFFF;
        test_data [ 1] [0] = 16'hAAAA;
        test_data [ 1] [1] = 16'hCCCC;
        test_data [ 2] [0] = 16'hCDD7;
        test_data [ 2] [1] = 16'hBABA;
        test_data [ 3] [0] = 16'h4444;
```

```
  test_data [ 3] [1] = 16'hAAAA;
  test_data [ 4] [0] = 16'h7398;
  test_data [ 4] [1] = 16'hFFDD;
  test_data [ 5] [0] = 16'h1111;
  test_data [ 5] [1] = 16'h5982;
  test_data [ 6] [0] = 16'hFFFF;
  test_data [ 6] [1] = 16'hFFFF;
  test_data [ 7] [0] = 16'h1478;
  test_data [ 7] [1] = 16'hA3B9;
  test_data [ 8] [0] = 16'h0000;
  test_data [ 8] [1] = 16'h0000;
  test_data [ 9] [0] = 16'h99C5;
  test_data [ 9] [1] = 16'h7435;
  test_data [10] [0] = 16'h69D9;
  test_data [10] [1] = 16'hABCD;

  /*
  test_data [ 0] [0] = 16'hAAAA;
  test_data [ 0] [1] = 16'hCCCC;
  test_data [ 1] [0] = 16'hFFFF;
  test_data [ 1] [1] = 16'hFFFF;
  test_data [ 2] [0] = 16'hFFFF;
  test_data [ 2] [1] = 16'hFFFF;
  test_data [ 3] [0] = 16'hFFFF;
  test_data [ 3] [1] = 16'hFFFF;
  test_data [ 4] [0] = 16'hFFFF;
  test_data [ 4] [1] = 16'hFFFF;
  test_data [ 5] [0] = 16'hFFFF;
  test_data [ 5] [1] = 16'hFFFF;
  test_data [ 6] [0] = 16'hFFFF;
  test_data [ 6] [1] = 16'hFFFF;
  test_data [ 7] [0] = 16'hFFFF;
  test_data [ 7] [1] = 16'hFFFF;
  test_data [ 8] [0] = 16'hFFFF;
  test_data [ 8] [1] = 16'hFFFF;
  test_data [ 9] [0] = 16'hFFFF;
  test_data [ 9] [1] = 16'hFFFF;
  test_data [10] [0] = 16'hFFFF;
  test_data [10] [1] = 16'hFFFF;*/

  /*
  test_data [ 0] [0] = 16'hAAAA;
  test_data [ 0] [1] = 16'hCCCC;
  test_data [ 1] [0] = 16'h0000;
  test_data [ 1] [1] = 16'h0000;
  test_data [ 2] [0] = 16'h0000;
  test_data [ 2] [1] = 16'h0000;
  test_data [ 3] [0] = 16'h0000;
  test_data [ 3] [1] = 16'h0000;
  test_data [ 4] [0] = 16'h0000;
  test_data [ 4] [1] = 16'h0000;
  test_data [ 5] [0] = 16'h0000;
  test_data [ 5] [1] = 16'h0000;
  test_data [ 6] [0] = 16'h0000;
  test_data [ 6] [1] = 16'h0000;
  test_data [ 7] [0] = 16'h0000;
  test_data [ 7] [1] = 16'h0000;
  test_data [ 8] [0] = 16'h0000;
  test_data [ 8] [1] = 16'h0000;
  test_data [ 9] [0] = 16'h0000;
  test_data [ 9] [1] = 16'h0000;
  test_data [10] [0] = 16'h0000;
  test_data [10] [1] = 16'h0000;*/


// #200000;


  end
```

```
    always
    begin
        count = 0;
        forever
        begin
            #5 clk = ~clk;
// 100 MHz clock rate (100MHz/10^9)/2
            count = count + 1;
        end
    end



    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt     <= 0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            bit_cnt     <= 0;
// count number of bits
            word_cnt    <= 0;
// count the word number
            lr_cnt      <= 0;
// left=0 and right=1
            inp_sck     <= 0;
// serial clock
            sck_d1      <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
                inp_sck <= ~inp_sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1<=inp_sck;
// generate 1 cycle delay of inp_sck
            if(inp_sck & ~sck_d1)
// on a positive transition of sck...

            begin
                if (bit_cnt==bit_tc)
// bit_tc = 15
                begin
                    if (lr_cnt == 1)
// if right
                    begin
                        word_cnt<=word_cnt+1;
// words in the testbench array
                        lr_cnt<=0;
// set to left
                    end
                    else
                        lr_cnt<=1;
// set to right
                    bit_cnt<=0;
// reset bit counter
                end
                else
```

```
                    bit_cnt<=bit_cnt+1;
// increment bit counter
            end
        end
    end


    assign rst_n = !(count < 20);
    assign rf_i2si_en = !(count < 20);
    assign i2si_rtr = i2si_rts;
    assign inp_ws = ((0<=bit_cnt& bit_cnt<=16'd14)&lr_cnt==1) | ((bit_cnt==16'd15)&(lr_cnt==0));
    assign inp_sd = test_data [word_cnt][lr_cnt][bit_tc-bit_cnt];



    //Checks if the data was properly deserialized.
    //First two rtr && rts cycles are skipped because if comparing 32'h00000000
    //the reset value of 0 will match with one of the input words and match_count will be wrong
    always @(posedge clk)
    begin
        if(i2si_rts)
        begin
            if(!skipped_two_cycles)
            begin
                cycle_count = cycle_count + 1;
                if(cycle_count == 2)
                    skipped_two_cycles = 1;
            end
            else
            begin
                //Find the first input word that matches the output word
                if(!match_found)
                begin
                    for(match_count = 0; match_count < `N; match_count = match_count + 1)
                    begin
                        word = {test_data [match_count] [0], test_data [match_count] [1]};
                        if(word == i2si_data && !match_found)
                        begin
                            match_found = 1;
                            begin_comparison = 1;
                            test_failed = 0;
                            compare_count = match_count;
                        end
                    end
                end
                //Stop checking. No input words matched output words
                if(count > 20000 && !begin_comparison && test_failed)
                begin
                    match_found = 1;
                    test_failed = 0;
                    $fdisplay(out, "No matches found. Test failed");
                    #1 $fclose(out);
                end


                //After finding first input word that matches, begin comparing rest of the
inputted words
                if(begin_comparison)
                begin
                    word = {test_data [compare_count] [0], test_data [compare_count] [1]};
                    //If word inputted and outputted match
                    if(word == i2si_data)
                    begin
                        pass_count = pass_count + 1;
                        $fdisplay(out, "Input: %h", word,
                            "        ---       Output: %h",
                            i2si_data, "        ---       Pass");
                    end
                    //End of comparison test. No more words were inputted.
                    else if(i2si_data === 32'hxxxxxxxx)
```

```
                    begin
                        begin_comparison = 0;
                        $fdisplay(out, "\nNumber of Comparisons:                    %d",
pass_count + fail_count,
                            "\nNumber of Successful Comparisons:         %d", pass_count,
                            "\nNumber of Failed Comparisons:           %d", fail_count);
                        #1 $fclose(out);
                    end
                    //If words do not match
                    else
                    begin
                        fail_count = fail_count + 1;
                        $fdisplay(out, "Input: %h", word,
                            "        ---      Output: %h",
                            i2si_data, "        ---      Fail");
                    end
                    compare_count = compare_count + 1;
                end
            end
        end
    end

endmodule
```

i2s_in_test2.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   i2s_in_test2.v
// Create Date:   1/10/2016
// Last Edit:     3/20/2016
// Author:        Kevin Cao
//
// Description      Tests the BIST Generator. Sawtooth wave starting value is 0x800 and upper
limit value is 0x700
//                  Outputs success or failure of test to i2s_in_test2_output.txt
//
//                  Comparison test is currently broken. However BIST has been verified to work
//
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define N 11 // number of test elements

module i2s_in_test2;

          // Inputs
          reg                               clk;
          wire                              rst_n;

          reg                               inp_sck;
          wire                              inp_ws;
          wire                              inp_sd;

          wire                              rf_i2si_en;
          reg     [11:0]                    rf_bist_start_val;
          reg     [ 7:0]                    rf_bist_inc;
          reg     [11:0]                    rf_bist_up_limit;
          reg                               rf_mux_en;

    wire                            i2si_rtr;

    reg                             trig_fifo_overrun_clr;

          // Outputs
          wire    [31:0]                    i2si_data;
          wire                              i2si_rts;
          wire                              ro_fifo_overrun;
          wire                              sync_sck;

    // Internal Variables
    reg     [15:0]                  test_data [`N-1:0] [0:1];
// [Bits Per Word] test_data [# of entities in test] [Left/Right]
    reg                             sck_d1;
// serial clock delay
    integer                         count;
// clock counter
    integer                         sck_cnt;
// serial clock counter
    integer                         bit_cnt;
// bit number counter
    integer                         lr_cnt;
// left right counter
    integer                         word_cnt;
// word counter
    parameter                       cyc_per_half_sck = 40;
// about (100 MHz / 1.44 MHz)/2
    parameter                       bit_tc =  15;
// number of bits in a word

    reg     [31:0]                  word;
// Stores the expected word, to be compared with i2si_data
    reg                             test_complete = 0;
// Boolean to help decide if to continue comparing word and i2si_data
```

```
    reg                             ignore_first_fail = 0;
// Boolean that ignores the first failure if word and i2si_data do not match
    integer                         cycles_complete = 0;
// Tracks the # of cycles the test succeeded


    reg     [15:0]                  ext_start_val;
//16 bit sign extension of rf_bist_start_val

    integer                         out;
// Helps create output text file

          // Instantiate the Unit Under Test (UUT)
          i2s_in uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .inp_sck(inp_sck),
                    .inp_ws(inp_ws),
                    .inp_sd(inp_sd),
                    .rf_i2si_en(rf_i2si_en),
                    .rf_bist_start_val(rf_bist_start_val),
                    .rf_bist_inc(rf_bist_inc),
                    .rf_bist_up_limit(rf_bist_up_limit),
                    .rf_mux_en(rf_mux_en),
                    .i2si_rtr(i2si_rtr),
                    .i2si_data(i2si_data),
                    .i2si_rts(i2si_rts),
                    .trig_fifo_overrun_clr(trig_fifo_overrun_clr),
                    .ro_fifo_overrun(ro_fifo_overrun),
                    .sync_sck(sync_sck),
        .sync_sck_transition(sync_sck_transition)
          );

          initial begin
                    clk = 0;
                    inp_sck = 0;
        rf_bist_start_val = 12'h800;
// Set starting value to 1
        rf_bist_inc = 8'h80;
// Set increment value to 1
        rf_bist_up_limit = 12'h700;
// Set BIST upper limit to 25
                    rf_mux_en = 1;
// Enable multiplexer, Output BIST generator content
                    trig_fifo_overrun_clr = 0;


        ext_start_val = {{4{rf_bist_start_val[11]}}, rf_bist_start_val};

        word = {~ext_start_val, ext_start_val};
// Set expected word to the starting value

        out = $fopen("i2s_in_test2_output.txt");
// Open i2si_in_test2_output.txt



        // Initialize Test Data
        // Not relevant to BIST testing
        test_data [ 0] [0] = 16'hAAAA;
        test_data [ 0] [1] = 16'hFFFF;
        test_data [ 1] [0] = 16'hAAAA;
        test_data [ 1] [1] = 16'hCCCC;
        test_data [ 2] [0] = 16'hCDD7;
        test_data [ 2] [1] = 16'hBABA;
        test_data [ 3] [0] = 16'h4444;
        test_data [ 3] [1] = 16'hAAAA;
        test_data [ 4] [0] = 16'h7398;
        test_data [ 4] [1] = 16'hFFDD;
        test_data [ 5] [0] = 16'h1111;
        test_data [ 5] [1] = 16'h5982;
```

```
        test_data [ 6] [0] = 16'h0001;
        test_data [ 6] [1] = 16'hFFFF;
        test_data [ 7] [0] = 16'h1478;
        test_data [ 7] [1] = 16'hA3B9;
        test_data [ 8] [0] = 16'hF8D5;
        test_data [ 8] [1] = 16'hD55A;
        test_data [ 9] [0] = 16'h99C5;
        test_data [ 9] [1] = 16'h7435;
        test_data [10] [0] = 16'h69D9;
        test_data [10] [1] = 16'hABCD;


          end


    // Defining the master clock
    always
    begin
        count = 0;
        forever
        begin
            #5 clk = ~clk;
// 100 MHz clock
            count = count + 1;
        end
    end



    // Defining the serial clock, and keeps track of which test_data bit to input
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt     <= 0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            bit_cnt     <= 0;
// count number of bits
            word_cnt    <= 0;
// count the word number
            lr_cnt      <= 0;
// left=0 and right=1
            inp_sck     <= 0;
// serial clock
            sck_d1      <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
                inp_sck <= ~inp_sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1<=inp_sck;
// generate 1 cycle delay of inp_sck
            if(inp_sck & ~sck_d1)
// on a positive transition of sck...

            begin
                if (bit_cnt==bit_tc)
// bit_tc = 15
                begin
```

```
                    if (lr_cnt == 1)
// if right
                    begin
                        word_cnt<=word_cnt+1;
// words in the testbench array
                        lr_cnt<=0;
// set to left
                    end
                    else
                        lr_cnt<=1;
// set to right
                    bit_cnt<=0;
// reset bit counter
                end
                else
                    bit_cnt<=bit_cnt+1;
// increment bit counter
            end
        end
    end


    assign rst_n = !(count < 20);
    assign rf_i2si_en = !(count < 20);
    assign i2si_rtr = i2si_rts;
    assign inp_ws = ((0<=bit_cnt& bit_cnt<=16'd14)&lr_cnt==1) | ((bit_cnt==16'd15)&(lr_cnt==0));
    assign inp_sd = test_data [word_cnt][lr_cnt][bit_tc-bit_cnt];


    //Checks if the data was properly deserialized
    always @(posedge clk)
    begin
        if(i2si_rts && i2si_rtr)
        begin
            //Find the first input word that matches the output word
            if(!test_complete)
            begin
                // word and i2si_data match. Print succeeded comparison
                if(word == i2si_data)
                begin
                    $fdisplay (out, "word == i2si_data        word: %d", word,
                    "        i2si_data: %d", i2si_data, "      ---     PASS");
                    word = word + rf_bist_inc;
                    // If upper limit is reached, reset word to beginning value
                    if(word > rf_bist_up_limit)
                    begin
                        cycles_complete = cycles_complete + 1;
                        $fdisplay(out, "\ncycles_complete: %d", cycles_complete, "\n");
                        word = {~ext_start_val, ext_start_val};
                        // If 2 cycles of BIST succeeded end comparison
                        if(cycles_complete == 2)
                        begin
                            test_complete = 1;
                            $fdisplay(out, "TEST SUCCEEDED!!!");
                            #1 $fclose(out);
                        end
                    end
                end
                // Ignores the first failure, as i2si_data begins at value 0 and
rf_bist_start_val may not start at 0
                else if(!ignore_first_fail)
                begin
                    ignore_first_fail = 1;
                end
                // An error was found. Print offending comparison and close text file
                else
                begin
                    $fdisplay (out, "word != i2si_data        word: %d", word,
                    "        i2si_data: %d", i2si_data, "      ---     FAIL");
                    test_complete = 1;
                    $fdisplay (out, "TEST FAILED!!!");
```

```
                    #1 $fclose(out);
                end
            end
        end
    end

endmodule
```

i2s_in_test3.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   i2s_in_test3.v
// Create Date:   01/16/16
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Creates N number random 32 bit words being inputted.
//                  Compares the inputted and outputted words.
//                  Outputs the success and failure of the comparisons in i2s_in_test3_output.txt
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define N 101 // number of test elements

module i2s_in_test3;

         // Inputs
         reg                            clk;
         wire                           rst_n;

         reg                            inp_sck;
         wire                           inp_ws;
         wire                           inp_sd;

         wire                           rf_i2si_en;
         reg     [11:0]                 rf_bist_start_val;
         reg     [ 7:0]                 rf_bist_inc;
         reg     [11:0]                 rf_bist_up_limit;
         reg                            rf_mux_en;

    wire                            i2si_rtr;

    reg                             trig_fifo_overrun_clr;

         // Outputs
         wire    [31:0]                 i2si_data;
         wire                           i2si_rts;
         wire                           ro_fifo_overrun;
         wire                           sync_sck;


    // Internal Variables
    reg                             sck_d1;
// serial clock delay
    integer                         count;
// clock counter
    integer                         sck_cnt;
// serial clock counter
    integer                         bit_cnt;
// bit number counter
    integer                         lr_cnt;
// left right counter
    integer                         word_cnt;
// word counter
    parameter                       cyc_per_half_sck = 40;
// about (100 MHz / 1.44 MHz)/2
    parameter                       bit_tc =  15;
// number of bits in a word


    reg     [15:0]                  test_data [`N-1:0] [0:1];
// [Bits Per Word] test_data [# of entities in test] [Left/Right]
    integer                         index1;
// Counter to help create random 32 bit words
    integer                         index2;
// Counter to help create random 32 bit words
```

```
    integer                         match_count = 0;
// Counter to help find initial matching words for word and i2si_data
    integer                         compare_count = 0;
// Counter to help compare word and i2si_data after intial match is found
    integer                         pass_count = 0;
// Counts number of successful comparisons
    integer                         fail_count = 0;
// Counts number of failed comparisons
    integer                         cycle_count = 0;
// Count to help keep track of how many i2si_rtr && i2si_rts transitions occurred
    reg                             match_found = 0;
// Boolean that determines if intial match was found
    reg                             begin_comparison = 0;
// Boolean that determines if to start comparing words after intial match is found
    reg                             test_failed = 1;
// Boolean that determines if no matches were found and the test failed
    reg                             skipped_two_cycles = 0;
// Boolean that determines if the first two transitions of i2si_rtr and i2si_rts was skipped
    reg     [31:0]                  word;
// Stores the expected word, to be compared with i2si_data

    integer                         out;
// Helps create output text file



          // Instantiate the Unit Under Test (UUT)
          i2s_in uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .inp_sck(inp_sck),
                    .inp_ws(inp_ws),
                    .inp_sd(inp_sd),
                    .rf_i2si_en(rf_i2si_en),
                    .rf_bist_start_val(rf_bist_start_val),
                    .rf_bist_inc(rf_bist_inc),
                    .rf_bist_up_limit(rf_bist_up_limit),
                    .rf_mux_en(rf_mux_en),
                    .i2si_rtr(i2si_rtr),
                    .i2si_data(i2si_data),
                    .i2si_rts(i2si_rts),
                    .ro_fifo_overrun(ro_fifo_overrun),
                    .trig_fifo_overrun_clr(trig_fifo_overrun_clr),
                    .sync_sck(sync_sck),
        .sync_sck_transition(sync_sck_transition)
          );

          initial begin
                    clk = 0;
                    inp_sck = 0;
        rf_bist_start_val = 12'd1;
// Set starting value to 1
        rf_bist_inc = 12'd1;
// Set increment value to 1
        rf_bist_up_limit = 12'd25;
// Set BIST upper limit to 25
                    rf_mux_en = 0;
// Disable multiplexer select bit, Output deserializer's contents
                    trig_fifo_overrun_clr = 0;


        out = $fopen("i2s_in_test3_output.txt");
// Open i2si_in_test2_output.txt



        for(index1 = 0; index1 < `N; index1 = index1 + 1)
        begin
            for(index2 = 0; index2 < 2; index2 = index2 + 1)
            begin
```

```
                test_data [index1] [index2] = $random;
            end
        end


       // #200000;


         end



    always
    begin
        count = 0;
        forever
        begin
            #5 clk = ~clk;
// 100 MHz clock
            count = count + 1;
        end
    end




    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt     <= 0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            bit_cnt     <= 0;
// count number of bits
            word_cnt    <= 0;
// count the word number
            lr_cnt      <= 0;
// left=0 and right=1
            inp_sck     <= 0;
// serial clock
            sck_d1      <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
                inp_sck <= ~inp_sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1<=inp_sck;
// generate 1 cycle delay of inp_sck
            if(inp_sck & ~sck_d1)
// on a positive transition of sck...

            begin
                if (bit_cnt==bit_tc)
// bit_tc = 15
                begin
                    if (lr_cnt == 1)
// if right
                    begin
```

```
                        word_cnt<=word_cnt+1;
// words in the testbench array
                        lr_cnt<=0;
// set to left
                    end
                    else
                    begin
                        lr_cnt<=1;
// set to right

                    end
                    bit_cnt<=0;
// reset bit counter
                end
                else
                    bit_cnt<=bit_cnt+1;
// increment bit counter
            end
        end
    end


    assign rst_n = !(count < 20);
    assign rf_i2si_en = !(count < 20);
    assign i2si_rtr = i2si_rts;
    assign inp_ws = ((0<=bit_cnt& bit_cnt<=16'd14)&lr_cnt==1) | ((bit_cnt==16'd15)&(lr_cnt==0));
    assign inp_sd = test_data [word_cnt][lr_cnt][bit_tc-bit_cnt];



    //Checks if the data was properly deserialized.
    //First two rtr && rts cycles are skipped because if comparing 32'h00000000
    //the reset value of 0 will match with one of the input words and match_count will be wrong
    always @(posedge clk)
    begin
        if(i2si_rts)
        begin
            if(!skipped_two_cycles)
            begin
                cycle_count = cycle_count + 1;
                if(cycle_count == 2)
                    skipped_two_cycles = 1;
            end
            else
            begin
                //Find the first input word that matches the output word
                if(!match_found)
                begin
                    for(match_count = 0; match_count < `N; match_count = match_count + 1)
                    begin
                        word = {test_data [match_count] [0], test_data [match_count] [1]};
                        if(word == i2si_data && !match_found)
                        begin
                            match_found = 1;
                            begin_comparison = 1;
                            test_failed = 0;
                            compare_count = match_count;
                        end
                    end
                end
                //Stop checking. No input words matched output words
                if(count > 20000 && !begin_comparison && test_failed)
                begin
                    match_found = 1;
                    test_failed = 0;
                    $fdisplay(out, "No matches found. Test failed");
                    #1 $fclose(out);
                end
```

```
                //After finding first input word that matches, begin comparing rest of the
inputted words
                if(begin_comparison)
                begin
                    word = {test_data [compare_count] [0], test_data [compare_count] [1]};
                    //If word inputted and outputted match
                    if(word == i2si_data)
                    begin
                        pass_count = pass_count + 1;
                        $fdisplay(out, "Input: %h", word,
                            "        ---       Output: %h",
                            i2si_data, "        ---       Pass");
                    end
                    //End of comparison test. No more words were inputted.
                    else if(i2si_data === 32'hxxxxxxxx)
                    begin
                        begin_comparison = 0;
                        $fdisplay(out, "\nNumber of Comparisons:                    %d",
pass_count + fail_count,
                            "\nNumber of Successful Comparisons:         %d", pass_count,
                            "\nNumber of Failed Comparisons:           %d", fail_count);
                        #1 $fclose(out);
                    end
                    //If words do not match
                    else
                    begin
                        fail_count = fail_count + 1;
                        $fdisplay(out, "Input: %h", word,
                            "        ---       Output: %h",
                            i2si_data, "        ---       Fail");
                    end
                    compare_count = compare_count + 1;
                end
            end
        end
    end


endmodule
```

i2s_in_test4.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   i2s_in_test4.v
// Create Date:   01/16/16
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Testing different clock rates
//                  Creates N number random 32 bit words being inputted.
//                  Compares the inputted and outputted words.
//                  Master Clock Rate:
//                  Serial Clock Rate:
//                  Outputs the success and failure of the comparisons in i2s_in_test3_output.txt
//
//                  IN PROGRESS
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define N 101 // number of test elements

module i2s_in_test4;

          // Inputs
          reg                               clk;
          wire                              rst_n;

          reg                               inp_sck;
          wire                              inp_ws;
          wire                              inp_sd;

          wire                              rf_i2si_en;
          reg      [11:0]                   rf_bist_start_val;
          reg      [ 7:0]                   rf_bist_inc;
          reg      [11:0]                   rf_bist_up_limit;
          reg                               rf_mux_en;

    wire                            i2si_rtr;

    reg                             trig_fifo_overrun_clr;

          // Outputs
          wire     [31:0]                   i2si_data;
          wire                              i2si_rts;
          wire                              ro_fifo_overrun;
          wire                              sync_sck;


    // Internal Variables
    reg                             sck_d1;
// serial clock delay
    integer                         count;
// clock counter
    integer                         sck_cnt;
// serial clock counter
    integer                         bit_cnt;
// bit number counter
    integer                         lr_cnt;
// left right counter
    integer                         word_cnt;
// word counter
    parameter                       cyc_per_half_sck = 1;
// about (100 MHz / 0.256 MHz)/2
    parameter                       bit_tc =  15;
// number of bits in a word


    reg      [15:0]                 test_data [`N-1:0] [0:1];
// [Bits Per Word] test_data [# of entities in test] [Left/Right]
```

```
    integer                         index1;
// Counter to help create random 32 bit words
    integer                         index2;
// Counter to help create random 32 bit words


    integer                         match_count = 0;
// Counter to help find initial matching words for word and i2si_data
    integer                         compare_count = 0;
// Counter to help compare word and i2si_data after intial match is found
    integer                         pass_count = 0;
// Counts number of successful comparisons
    integer                         fail_count = 0;
// Counts number of failed comparisons
    integer                         cycle_count = 0;
// Count to help keep track of how many i2si_rtr && i2si_rts transitions occurred
    reg                             match_found = 0;
// Boolean that determines if intial match was found
    reg                             begin_comparison = 0;
// Boolean that determines if to start comparing words after intial match is found
    reg                             test_failed = 1;
// Boolean that determines if no matches were found and the test failed
    reg                             skipped_two_cycles = 0;
// Boolean that determines if the first two transitions of i2si_rtr and i2si_rts was skipped
    reg     [31:0]                  word;
// Stores the expected word, to be compared with i2si_data

    integer                         out;
// Helps create output text file



          // Instantiate the Unit Under Test (UUT)
          i2s_in uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .inp_sck(inp_sck),
                    .inp_ws(inp_ws),
                    .inp_sd(inp_sd),
                    .rf_i2si_en(rf_i2si_en),
                    .rf_bist_start_val(rf_bist_start_val),
                    .rf_bist_inc(rf_bist_inc),
                    .rf_bist_up_limit(rf_bist_up_limit),
                    .rf_mux_en(rf_mux_en),
                    .i2si_rtr(i2si_rtr),
                    .i2si_data(i2si_data),
                    .i2si_rts(i2si_rts),
                    .ro_fifo_overrun(ro_fifo_overrun),
                    .trig_fifo_overrun_clr(trig_fifo_overrun_clr),
                    .sync_sck(sync_sck),
        .sync_sck_transition(sync_sck_transition)
          );

          initial begin
                    clk = 0;
                    inp_sck = 0;
        rf_bist_start_val = 12'd1;
// Set starting value to 1
        rf_bist_inc = 12'd1;
// Set increment value to 1
        rf_bist_up_limit = 12'd25;
// Set BIST upper limit to 25
                    rf_mux_en = 0;
// Disable multiplexer select bit, Output deserializer's contents
                    trig_fifo_overrun_clr = 0;


        out = $fopen("i2s_in_test4_output.txt");
// Open i2si_in_test4_output.txt
```

```
        for(index1 = 0; index1 < `N; index1 = index1 + 1)
        begin
            for(index2 = 0; index2 < 2; index2 = index2 + 1)
            begin
                test_data [index1] [index2] = $random;
            end
        end



       // #200000;


          end



    always
    begin
        count = 0;
        forever
        begin
            #50 clk = ~clk;
// 100 MHz clock rate (100MHz/10^9)/2
            count = count + 1;
        end
    end




    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt     <= 0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            bit_cnt     <= 0;
// count number of bits
            word_cnt    <= 0;
// count the word number
            lr_cnt      <= 0;
// left=0 and right=1
            inp_sck     <= 0;
// serial clock
            sck_d1      <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
                inp_sck <= ~inp_sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1<=inp_sck;
// generate 1 cycle delay of inp_sck
            if(inp_sck & ~sck_d1)
// on a positive transition of sck...

            begin
```

```
                if (bit_cnt==bit_tc)
// bit_tc = 15
                begin
                    if (lr_cnt == 1)
// if right
                    begin
                        word_cnt<=word_cnt+1;
// words in the testbench array
                        lr_cnt<=0;
// set to left
                    end
                    else
                    begin
                        lr_cnt<=1;
// set to right

                    end
                    bit_cnt<=0;
// reset bit counter
                end
                else
                    bit_cnt<=bit_cnt+1;
// increment bit counter
            end
        end
    end


    assign rst_n = !(count < 20);
    assign rf_i2si_en = !(count < 20);
    assign i2si_rtr = i2si_rts;
    assign inp_ws = ((0<=bit_cnt& bit_cnt<=16'd14)&lr_cnt==1) | ((bit_cnt==16'd15)&(lr_cnt==0));
    assign inp_sd = test_data [word_cnt][lr_cnt][bit_tc-bit_cnt];



    //Checks if the data was properly deserialized.
    //First two rtr && rts cycles are skipped because if comparing 32'h00000000
    //the reset value of 0 will match with one of the input words and match_count will be wrong
    always @(posedge clk)
    begin
        if(i2si_rts)
        begin
            if(!skipped_two_cycles)
            begin
                cycle_count = cycle_count + 1;
                if(cycle_count == 2)
                    skipped_two_cycles = 1;
            end
            else
            begin
                //Find the first input word that matches the output word
                if(!match_found)
                begin
                    for(match_count = 0; match_count < `N; match_count = match_count + 1)
                    begin
                        word = {test_data [match_count] [0], test_data [match_count] [1]};
                        if(word == i2si_data && !match_found)
                        begin
                            match_found = 1;
                            begin_comparison = 1;
                            test_failed = 0;
                            compare_count = match_count;
                        end
                    end
                end
                //Stop checking. No input words matched output words
                if(count > 20000 && !begin_comparison && test_failed)
                begin
                    match_found = 1;
                    test_failed = 0;
```

```
                    $fdisplay(out, "No matches found. Test failed");
                    #1 $fclose(out);
                end


                //After finding first input word that matches, begin comparing rest of the
inputted words
                if(begin_comparison)
                begin
                    word = {test_data [compare_count] [0], test_data [compare_count] [1]};
                    //If word inputted and outputted match
                    if(word == i2si_data)
                    begin
                        pass_count = pass_count + 1;
                        $fdisplay(out, "Input: %h", word,
                            "        ---       Output: %h",
                            i2si_data, "        ---       Pass");
                    end
                    //End of comparison test. No more words were inputted.
                    else if(i2si_data === 32'hxxxxxxxx)
                    begin
                        begin_comparison = 0;
                        $fdisplay(out, "\nNumber of Comparisons:                    %d",
pass_count + fail_count,
                            "\nNumber of Successful Comparisons:         %d", pass_count,
                            "\nNumber of Failed Comparisons:           %d", fail_count);
                        #1 $fclose(out);
                    end
                    //If words do not match
                    else
                    begin
                        fail_count = fail_count + 1;
                        $fdisplay(out, "Input: %h", word,
                            "        ---       Output: %h",
                            i2si_data, "        ---       Fail");
                    end
                    compare_count = compare_count + 1;
                end
            end
        end
    end


endmodule
```

mux_test.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   mux_test.v
// Create Date:   10/13/2015
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Testing output of multiplexer
//                  Ensuring multiplexer outputs correct inputs based on select bit
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module mux_test;

          // Inputs
          reg         [31:0]                          in_0_data;
          reg                                         in_0_xfc;
          reg         [31:0]                          in_1_data;
          reg                                         in_1_xfc;
          reg                                         sel;

          // Outputs
          wire        [31:0]                          mux_data;
          wire                                        mux_xfc;

          // Instantiate the Unit Under Test (UUT)
          i2si_mux uut (
                    .in_0_data(in_0_data),
                    .in_0_xfc(in_0_xfc),
                    .in_1_data(in_1_data),
                    .in_1_xfc(in_1_xfc),
                    .sel(sel),
                    .mux_data(mux_data),
                    .mux_xfc(mux_xfc)
          );

          initial
    begin
                    in_0_data = 0;
                    in_0_xfc = 0;
                    in_1_data = 0;
                    in_1_xfc = 0;
                    sel = 0;
                    #10;

        in_0_data = 32'd25;
        in_0_xfc = 1;
        in_1_data = 32'd50;
        in_1_xfc = 1;
        sel = 1;
        #10;

        in_0_data = 32'd100;
        in_0_xfc = 1;
        in_1_data = 32'd1000;
        in_1_xfc = 0;
        sel = 0;
        #10;

        in_0_data = 32'd2048;
        in_0_xfc = 1;
        in_1_data = 32'd4096;
        in_1_xfc = 0;
        sel = 1;
        #10;

        in_0_data = 0;
                    in_0_xfc = 0;
```

```
                    in_1_data = 0;
                    in_1_xfc = 0;
                    sel = 0;

          end

endmodule
```

synchronizer_testbench.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   synchronizer_testbench.v
// Create Date:   10/13/2015
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Test to ensure input signals are delayed by x number of clock cycles
//                      sck: 2 clk cycles
//                      sd : 4 clk cycles
//                      ws : 4 clk cycles
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define N 11                                                            // number of test
elements

module synchronizer_testbench;

          // Inputs
          reg                     clk;
          wire                    rst_n;
          reg                     _sck;
          wire                    _sd;
          wire                    _ws;

          // Outputs
          wire                    sck;
          wire                    sck_transition;
          wire                    sd;
          wire                    ws;

    // Internal Variables
    reg         [15:0]      test_data [`N-1:0] [0:1];                   // [Bits Per
Word] test_data [# of entities in test] [Left/Right]
    reg                     sck_d1;                                     // serial clock
delay
    reg         [31:0]      count;                                      // clock counter
    reg         [31:0]      sck_cnt;                                    // serial clock
counter
    reg         [31:0]      bit_cnt;                                    // bit number
counter
    reg                     lr_cnt;                                     // left right
counter
    reg         [31:0]      word_cnt;                                   // word counter
    reg         [31:0]      cyc_per_half_sck = 40;                      // about (100 MHz
/ 1.44 MHz)/2
    reg         [31:0]      bit_tc =  15;                               // number of bits
in a word

          // Instantiate the Unit Under Test (UUT)
          synchronizer uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    ._sck(_sck),
                    .sck(sck),
                    .sck_transition(sck_transition),
                    ._sd(_sd),
                    .sd(sd),
                    ._ws(_ws),
                    .ws(ws)
          );

          initial
    begin
                    // Initialize Inputs
                    clk = 0;
```

```
        // Test Data
        test_data [0] [0] = 16'hAAAA;
        test_data [0] [1] = 16'hFFFF;
        test_data [1] [0] = 16'h1478;
        test_data [1] [1] = 16'hA3B9;
        test_data [2] [0] = 16'hCDD7;
        test_data [2] [1] = 16'hBABA;
        test_data [3] [0] = 16'h4444;
        test_data [3] [1] = 16'hAAAA;
        test_data [4] [0] = 16'h7398;
        test_data [4] [1] = 16'hFFDD;
        test_data [5] [0] = 16'h1111;
        test_data [5] [1] = 16'h5982;
        test_data [6] [0] = 16'h0001;
        test_data [6] [1] = 16'hFFFF;
        test_data [7] [0] = 16'h1478;
        test_data [7] [1] = 16'hA3B9;
        test_data [8] [0] = 16'hF8D5;
        test_data [8] [1] = 16'hD55A;
        test_data [9] [0] = 16'h99C5;
        test_data [9] [1] = 16'h7435;
        test_data [10] [0] = 16'h69D9;
        test_data [10] [1] = 16'hABCD;

                  // Wait 100 ns for global reset to finish
                  #100;
         end

    // Generate master clock
    always
    begin
        count = 0;                                                      // set clock counter to
zero
    forever
        begin
            #5 clk = ~clk;                                              // 100 MHz clock
            count = count + 1;                                          // increment clock
counter
        end
    end


    assign rst_n = !(count < 20);
    assign _ws = ((0<=bit_cnt& bit_cnt<=16'd14)&lr_cnt==1)|((bit_cnt==16'd15)&(lr_cnt==0));
    assign _sd = test_data [word_cnt][lr_cnt][bit_tc-bit_cnt];          // assign serial data
from the test_data


    //Generates sck signal
    //Help assign sd and ws values
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt<=0;                                                 // counts master clock
cycles, causes sck to toggle each time it hits cyc_per_half_sck
            bit_cnt<=0;                                                 // count number of bits
            word_cnt<=0;                                                // count the word number
            lr_cnt <= 0;                                                // left=0 and right=1
            _sck<=0;                                                    // serial clock
            sck_d1<=0;                                                  // serial clock delayed
by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)                          // cyc_per_half_sck ~
(100 MHz/1.44 MHz)/2
            begin
```

```
                sck_cnt <= 0;                                       // reset serial clock
counter
                _sck <= ~_sck;                                      // toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;                             // increment serial clock
counter

            sck_d1<=_sck;                                           // generate 1 cycle delay
of _sck
            if(_sck & ~sck_d1)                                      // on a positive
transition of sck...

            begin
                if (bit_cnt==bit_tc)                                // bit_tc = 15
                begin
                    if (lr_cnt == 1)                                // if right
                    begin
                        word_cnt<=word_cnt+1;                       // words in the testbench
array
                        lr_cnt<=0;                                  // set to left
                    end
                    else
                        lr_cnt<=1;                                  // set to right
                    bit_cnt<=0;                                     // reset bit counter
                end
                else
                    bit_cnt<=bit_cnt+1;                             // increment bit counter
            end

        end

    end

endmodule
```

i2s_out_test.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   i2s_out_test2.v
// Create Date:   11/27/15
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Creates NUM_ELEMENTS 32 bits words specified by the user that are fed
//                  into the FIFO and outputted in serial form. Words are fed into the buffer
//                  wihtout causing it to become full and triggering the FIFO full message.
//                  Creates i2s_out_test_output.txt with the results of the comparison test
//                  of the words queued into the FIFO and the words outputted by i2s_out.
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define NUM_ELEMENTS 16

module i2s_out_test;

          // Inputs
          reg                         clk;
          wire                        rst_n;

    wire                        i2so_sync_sck;
          wire                        i2so_sck_transition;

          reg                         filt_rts;
          reg     [31:0]              filt_data;

          reg                         trig_fifo_underrun;


          // Outputs
          wire                        filt_rtr;
          wire                        i2so_ws;
          wire                        i2so_sd;
          wire                        ro_fifo_underrun;


    // Internal Variables
    parameter                   cyc_per_half_sck = 40;
// about (100 MHz / 1.44 MHz)/2
    integer                     count;
// clock counter
    integer                     sck_cnt;
// serial clock counter
    reg                         sck;
// serial clock
    reg                         sck_d1;
// serial clock delay
    reg                         ws_d1;
// word select delay
    reg                         ws_d2;
// word select delay by 2 clock cycles
    wire                        ws_transition;
// level to pulse converter when ws_dl goes from low to high

    integer                     out;
// Helps create output text file of comparison test
    integer                     word_count;
// word counter
    integer                     match_count;
// counter that helps find the starting point for comparison
    integer                     filt_data_count = 0;
// keeps track of which data word to output
    integer                     pass_count =  0;
// counts number of comparisons that succeed
    integer                     fail_count = 0;
// counts number of comparisons that fail
```

```
    reg     [31:0]              test_data       [0:`NUM_ELEMENTS-1];
// test_data [# of entities in test]
    reg     [31:0]              filt_data_list  [0:`NUM_ELEMENTS-1];
// stores values of filt_data
    reg     [31:0]              word;
// serial data rebuilt in parallel, for comparison
    reg                         match_found = 0;
// boolean value used to find the starting point of comparison
    reg                         begin_comparison = 0;
// boolean value used to start the comparison loop
    reg                         comparison_failed = 1;
// boolean that determines if no matches were found and the comparison test failed




          // Instantiate the Unit Under Test (UUT)
          i2s_out uut (
                    .clk(clk),
                    .rst_n(rst_n),
        .i2so_sync_sck(i2so_sync_sck),
                    .i2so_sck_transition(i2so_sck_transition),
                    .filt_rts(filt_rts),
                    .filt_data(filt_data),
                    .filt_rtr(filt_rtr),
        .i2so_sck(i2so_sck),
                    .i2so_ws(i2so_ws),
                    .i2so_sd(i2so_sd),
                    .ro_fifo_underrun(ro_fifo_underrun),
                    .trig_fifo_underrun(trig_fifo_underrun)
          );

          initial
    begin
                    // Initialize Inputs
                    clk = 0;
                    filt_rts = 0;
                    filt_data = 0;
                    trig_fifo_underrun = 0;


        out = $fopen("i2s_out_test_output.txt");
// Open i2si_in_test_output.txt



        //Internal variables
        test_data [ 0] = 32'hFFFFFFFF;
        test_data [ 1] = 32'hAAAAAAAA;
        test_data [ 2] = 32'hFFFF0000;
        test_data [ 3] = 32'h0000FFFF;
        test_data [ 4] = 32'hCCCCCCCC;
        test_data [ 5] = 32'h33333333;
        test_data [ 6] = 32'h11111111;
        test_data [ 7] = 32'h22222222;
        test_data [ 8] = 32'hEEEEEEEE;
        test_data [ 9] = 32'h88888888;
        test_data [10] = 32'h00000000;
        test_data [11] = 32'hFFFFFFFF;
        test_data [12] = 32'hED32DE66;
        test_data [13] = 32'h0456CB22;
        test_data [14] = 32'h5256AE55;
        test_data [15] = 32'hE3FC48B4;


          end
```

```
    // Creates master clock signal
    always
    begin
        count = 0;
    forever
        begin
            #5 clk = ~clk;
            count = count + 1;
// increment clock counter
        end
    end


    // Creates the serial clock signal
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt <= 0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            sck <= 0;
// serial clock
            sck_d1 <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
                sck <= ~sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1 <= sck;
// generate 1 cycle delay of sck
        end
    end


    // Creates a delay of word select signal, used to help in comparison test
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            ws_d1 <= 0;
            ws_d2 <= 0;
        end
        else if(i2so_sck_transition)
        begin
            ws_d1 <= i2so_ws;
// generate 1 cycle delay of i2so_ws
            ws_d2 <= ws_d1;
// generate 2nd cycle delay of i2so_ws
        end
    end

    // Queue words into FIFO and stores word into an array for comparison
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            word_count <= 0;
        end
        // If word count is greater than the number of elements being queued, stop queueing data
```

```
        else if(word_count > `NUM_ELEMENTS-1)
        begin
            word_count <= word_count + 1;
        end
        // count % 2551 was determined by guess and check. Wanted to queue data slow enough so
buffer doesn't
        // become full. Also wanted to ensure every time ws_transition is high the variable word
is the next queued word
        // if using a number larger then 2551, word may not change its value to the next queued
word when ws_transition is high
        else if(count % 2551 == 0)
        begin
            queue(test_data [word_count]);
//indicates when to queue more data
            word_count <= word_count + 1;
//indicates which data to queue
            filt_data_list [word_count] <= filt_data;
//stores the word that is outputted by the fifo for comparison
        end
    end


    assign rst_n = !(count < 20);
// turn on reset after 10 clock cycles
    assign i2so_sync_sck = sck;
// assign serial clock input value
    assign i2so_sck_transition = sck & ~sck_d1;
// level to pulse converter when sck goes from low to high
    assign ws_transition = ~ws_d1 & ws_d2;
// level to pulse converter when ws goes from high to low


    // Creates 32 bit words from the serial data being outputted to be compared with the words in
filt_data_list
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            word <= 32'b0;
        end
        else if(i2so_sck_transition)
        begin
            word[31:1] <= word[30:0];
            word[0] <= i2so_sd;
        end
    end


    // Checking if data is properly serialized
    always @(posedge clk)
    begin
        if(ws_transition && i2so_sck_transition)
        begin
            // Find first input word that matches output word
            if(!match_found)
            begin
                for(match_count = 0; match_count < `NUM_ELEMENTS; match_count = match_count + 1)
                begin
                    if(word == filt_data_list[match_count])
                    begin
                        match_found = 1;
// ends loop to find matching words
                        begin_comparison = 1;
// booolean to begin comparison test
                        comparison_failed = 0;
// ensures failed message does not output
                        filt_data_count = match_count;
// sets which index in the array the comparison test should begin
                    end
                end
            end
```

```
            // No matches were found. End trying to find a match
            if(count > 30000 && !begin_comparison && comparison_failed)
            begin
                match_found = 1;
                comparison_failed = 0;
                $fdisplay (out, "No matches found. Comparison test failed");
                #1;
                $fclose(out);
            end

            // Initial match found. Begin comparison test
            if(begin_comparison)
            begin
                // If words inputted and outputted match
                if(word == filt_data_list[filt_data_count])
                begin
                    pass_count = pass_count + 1;
                    $fdisplay (out, "Input: %h", word, "     ---      ",
                        "Output: %h", filt_data_list [filt_data_count],
                        "      ---      Pass");
                end
                // End comparison test after 15 words
                else if(filt_data_count > (`NUM_ELEMENTS - 1))
                begin
                    begin_comparison = 0;
                    $fdisplay(out, "\nNumber of Comparisons:                    %d", pass_count +
fail_count,
                        "\nNumber of Successful Comparisons:         %d", pass_count,
                        "\nNumber of Failed Comparisons:           %d", fail_count);
                    $fclose(out);
                end
                // If words inputted and outputted do not match
                else
                begin
                    fail_count = fail_count + 1;
                    $fdisplay (out, "Input: %h", word, "     ---      ",
                        "Output: %h", filt_data_list [filt_data_count],
                        "      ---      Fail");
                end
                filt_data_count = filt_data_count + 1;
            end
        end
    end



    // Defining queue task for FIFO
    task queue;
// define the queue task
    input[31:0] data;
// the data to be queueed
       if(!filt_rtr)
// if buffer is full display warning
                $display("---Cannot queue: Buffer Full---");
            else
// if buffer is not full
            begin
               $display("Queued %h",data );
// display that the data was queueed
               filt_data = data;
// the input to the buffer is set as the data
               filt_rts = 1;
// write is enabled
               @(posedge clk);
// checks if clock is at postive edge
               #1 filt_rts = 0;
// set write enabled equal to zero then
            end
    endtask

endmodule
```

i2s_out_test2.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   i2s_out_test2.v
// Create Date:   1/19/16
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Creates NUM_ELEMENTS random 32 bits words that are fed into the FIFO
//                  and outputted in serial form. Words are fed into the buffer
//                  wihtout causing it to become full and triggering the FIFO full message.
//                  Creates i2s_out_test2_output.txt with the results of the comparison test
//                  of the words queued into the FIFO and the words outputted by i2s_out.
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define NUM_ELEMENTS 100 // no. of test elements

module i2s_out_test2;

          // Inputs
          reg                         clk;
          wire                        rst_n;

    wire                        i2so_sync_sck;
          wire                        i2so_sck_transition;

          reg                         filt_rts;
          reg     [31:0]              filt_data;

          reg                         trig_fifo_underrun;


          // Outputs
          wire                        filt_rtr;
          wire                        i2so_ws;
          wire                        i2so_sd;
          wire                        ro_fifo_underrun;


    // Internal Variables
    parameter                   cyc_per_half_sck = 40;
// about (100 MHz / 1.44 MHz)/2
    integer                     count = 0;
// clock counter
    integer                     sck_cnt;
// serial clock counter
    reg                         sck;
// serial clock
    reg                         sck_d1;
// serial clock delay
    reg                         ws_d1;
// word select delay
    reg                         ws_d2;
// word select delay by 2 clock cycles
    wire                        ws_transition;
// level to pulse converter when ws_dl goes from low to high

    integer                     out;
// Helps create output text file
    integer                     word_count;
// word counter. Counts number of words queued
    integer                     test_count;
// index counter for test_data
    integer                     match_count;
// counter that helps find the starting point for comparison
    integer                     filt_data_count = 0;
// keeps track of which data word to output
    integer                     pass_count =  0;
// counts number of comparisons that succeed
```

```
    integer                     fail_count = 0;
// counts number of comparisons that fail
    reg     [31:0]              test_data       [0:`NUM_ELEMENTS-1];
// test_data [# of entities in test]
    reg     [31:0]              filt_data_list  [0:`NUM_ELEMENTS-1];
// stores values of filt_data
    reg     [31:0]              word;
// serial data rebuilt in parallel, for comparison
    reg                         match_found = 0;
// boolean value used to find the starting point of comparison
    reg                         begin_comparison = 0;
// boolean value used to start the comparison loop
    reg                         comparison_failed = 1;
// boolean that determines if no matches were found and the comparison test failed




          // Instantiate the Unit Under Test (UUT)
          i2s_out uut (
                    .clk(clk),
                    .rst_n(rst_n),
        .i2so_sync_sck(i2so_sync_sck),
                    .i2so_sck_transition(i2so_sck_transition),
                    .filt_rts(filt_rts),
                    .filt_data(filt_data),
                    .filt_rtr(filt_rtr),
        .i2so_sck(i2so_sck),
                    .i2so_ws(i2so_ws),
                    .i2so_sd(i2so_sd),
                    .ro_fifo_underrun(ro_fifo_underrun),
                    .trig_fifo_underrun(trig_fifo_underrun)
          );

          initial
    begin
                    // Initialize Inputs
                    clk = 0;
                    filt_rts = 0;
                    filt_data = 0;
                    trig_fifo_underrun = 0;


        out = $fopen("i2s_out_test2_output.txt");
// Open i2s_out_test2_output.txt



        //Internal variables
        for(test_count = 0; test_count < `NUM_ELEMENTS; test_count = test_count + 1)
        begin
            test_data [test_count] = $random;
        end

          end


    // Creates master clock signal
    always
    begin
    forever
        begin
            #5 clk = ~clk;
            count = count + 1;
// increment clock counter
        end
    end
```

```
    // Creates the serial clock signal
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt <= 0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            sck <= 0;
// serial clock
            sck_d1 <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
                sck <= ~sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1 <= sck;
// generate 1 cycle delay of sck
        end
    end


    // Creates a delay of word select signal, used to help in comparison test
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            ws_d1 <= 0;
            ws_d2 <= 0;
        end
        else if(i2so_sck_transition)
        begin
            ws_d1 <= i2so_ws;
// generate 1 cycle delay of i2so_ws
            ws_d2 <= ws_d1;
// generate 2nd cycle delay of i2so_ws
        end
    end

    // Queue words into FIFO and stores word into an array for comparison
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            word_count <= 0;
        end
        // If word count is greater than the number of elements being queued, stop queueing data
        else if(word_count > `NUM_ELEMENTS-1)
        begin
            word_count <= word_count + 1;
        end
        // count % 2551 was determined by guess and check. Wanted to queue data slow enough so
buffer doesn't
        // become full. Also wanted to ensure every time ws_transition is high the variable word
is the next queued word
        // if using a number larger then 2551, word may not change its value to the next queued
word when ws_transition is high
        else if(count % 2551 == 0)
        begin
```

```
            queue(test_data [word_count]);
//indicates when to queue more data
            word_count <= word_count + 1;
//indicates which data to queue
            filt_data_list [word_count] <= filt_data;
//stores the word that is outputted by the fifo for comparison
        end
    end


    assign rst_n = !(count < 20);
// turn on reset after 10 clock cycles
    assign i2so_sync_sck = sck;
// assign serial clock input value
    assign i2so_sck_transition = sck & ~sck_d1;
// level to pulse converter when sck goes from low to high
    assign ws_transition = ~ws_d1 & ws_d2;
// level to pulse converter when ws goes from high to low


    // Creates 32 bit words from the serial data being outputted to be compared with the words in
filt_data_list
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            word <= 32'b0;
        end
        else if(i2so_sck_transition)
        begin
            word[31:1] <= word[30:0];
            word[0] <= i2so_sd;
        end
    end


    // Checking if data is properly serialized
    always @(posedge clk)
    begin
        if(ws_transition && i2so_sck_transition)
        begin
            // Find first input word that matches output word
            if(!match_found)
            begin
                for(match_count = 0; match_count < `NUM_ELEMENTS; match_count = match_count + 1)
                begin
                    if(word == filt_data_list[match_count])
                    begin
                        match_found = 1;
// ends loop to find matching words
                        begin_comparison = 1;
// booolean to begin comparison test
                        comparison_failed = 0;
// ensures failed message does not output
                        filt_data_count = match_count;
// sets which test_count in the array the comparison test should begin
                    end
                end
            end
            // No matches were found. End trying to find a match
            if(count > 30000 && !begin_comparison && comparison_failed)
            begin
                match_found = 1;
                comparison_failed = 0;
                $fdisplay (out, "No matches found. Comparison test failed");
                #1 $fclose(out);
            end

            // Initial match found. Begin comparison test
            if(begin_comparison)
            begin
```

```
                // If words inputted and outputted match
                if(word == filt_data_list[filt_data_count])
                begin
                    pass_count = pass_count + 1;
                    $fdisplay (out, "Input: %h", word, "    ---     ",
                        "Output: %h", filt_data_list [filt_data_count],
                        "     ---     Pass");
                end
                // End comparison test after comparing all the test elements
                else if(filt_data_count > (`NUM_ELEMENTS-1))
                begin
                    begin_comparison = 0;
                    $fdisplay(out, "\nNumber of Comparisons:                    %d", pass_count +
fail_count,
                        "\nNumber of Successful Comparisons:         %d", pass_count,
                        "\nNumber of Failed Comparisons:           %d", fail_count);
                    $fclose(out);
                end
                // If words inputted and outputted do not match
                else
                begin
                    fail_count = fail_count + 1;
                    $fdisplay (out, "Input: %h", word, "    ---     ",
                        "Output: %h", filt_data_list [filt_data_count],
                        "     ---     Fail");
                end
                filt_data_count = filt_data_count + 1;
            end
        end
    end




    // Defining queue task for FIFO
    task queue;
// define the queue task
    input[31:0] data;
// the data to be queueed
        if(!filt_rtr)
// if buffer is full display warning
                $display("---Cannot queue: Buffer Full---");
            else
// if buffer is not full
            begin
               $display("Queued %h",data );
// display that the data was queueed
               filt_data = data;
// the input to the buffer is set as the data
               filt_rts = 1;
// write is enabled
               @(posedge clk);
// checks if clock is at postive edge
               #1 filt_rts = 0;
// set write enabled equal to zero then
            end
    endtask

endmodule
```

i2s_out_test3.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   i2s_out_test3.v
// Create Date:   1/25/16
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Creates NUM_ELEMENTS random 32 bits words that are fed into the FIFO
//                  and outputted in serial form. Words are fed into the buffer
//                  and causes it to become full and triggers the FIFO full message.
//                  Creates i2s_out_test3_output.txt with the results of the comparison test
//                  of the words queued into the FIFO and the words outputted by i2s_out.
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
`define NUM_ELEMENTS 500 // no. of test elements

module i2s_out_test3;

          // Inputs
          reg                         clk;
          wire                        rst_n;

    wire                        i2so_sync_sck;
          wire                        i2so_sck_transition;

          reg                         filt_rts;
          reg     [31:0]              filt_data;

          reg                         trig_fifo_underrun;


          // Outputs
          wire                        filt_rtr;
          wire                        i2so_ws;
          wire                        i2so_sd;
          wire                        ro_fifo_underrun;


    // Internal Variables
    parameter                   cyc_per_half_sck = 40;
// about (100 MHz / 1.44 MHz)/2
    integer                     count = 0;
// clock counter
    integer                     sck_cnt;
// serial clock counter
    reg                         sck;
// serial clock
    reg                         sck_d1;
// serial clock delay
    reg                         ws_d1;
// word select delay
    reg                         ws_d2;
// word select delay by 2 clock cycles
    wire                        ws_transition;
// level to pulse converter when ws_dl goes from low to high

    integer                     out;
// Helps create output text file
    integer                     word_count;
// word counter. Counts number of attempts of words being queued
    integer                     test_count;
// index counter for test_data
    integer                     match_count;
// counter that helps find the starting point for comparison
    integer                     filt_data_count = 0;
// keeps track of which data word to output
    integer                     words_queued = 0;
// counter that keeps track # of words queued in filt_data_list
```

```
    integer                     pass_count =  0;
// counts number of comparisons that succeed
    integer                     fail_count = 0;
// counts number of comparisons that fail
    reg     [31:0]              test_data       [0:`NUM_ELEMENTS-1];
// test_data [# of entities in test]
    reg     [31:0]              filt_data_list  [0:`NUM_ELEMENTS-1];
// stores values of filt_data
    reg     [31:0]              word;
// serial data rebuilt in parallel, for comparison
    reg                         match_found = 0;
// boolean value used to find the starting point of comparison
    reg                         begin_comparison = 0;
// boolean value used to start the comparison loop
    reg                         comparison_failed = 1;
// boolean that determines if no matches were found and the comparison test failed




          // Instantiate the Unit Under Test (UUT)
          i2s_out uut (
                    .clk(clk),
                    .rst_n(rst_n),
        .i2so_sync_sck(i2so_sync_sck),
                    .i2so_sck_transition(i2so_sck_transition),
                    .filt_rts(filt_rts),
                    .filt_data(filt_data),
                    .filt_rtr(filt_rtr),
        .i2so_sck(i2so_sck),
                    .i2so_ws(i2so_ws),
                    .i2so_sd(i2so_sd),
                    .ro_fifo_underrun(ro_fifo_underrun),
                    .trig_fifo_underrun(trig_fifo_underrun)
          );

          initial
    begin
                    // Initialize Inputs
                    clk = 0;
                    filt_rts = 0;
                    filt_data = 0;
                    trig_fifo_underrun = 0;


        out = $fopen("i2s_out_test3_output.txt");
// Open i2s_out_test2_output.txt



        //Internal variables
        for(test_count = 0; test_count < `NUM_ELEMENTS; test_count = test_count + 1)
        begin
            test_data [test_count] = $random;
        end

          end


    // Creates master clock signal
    always
    begin
    forever
        begin
            #5 clk = ~clk;
            count = count + 1;
// increment clock counter
        end
    end
```

```
    // Creates the serial clock signal
    always @ (posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            sck_cnt <= 0;
// counts master clock cycles, causes sck to toggle each time it hits cyc_per_half_sck
            sck <= 0;
// serial clock
            sck_d1 <= 0;
// serial clock delayed by one clock cycle
        end
        else
        begin

            if (sck_cnt == cyc_per_half_sck-1)
// cyc_per_half_sck ~ (100 MHz/1.44 MHz)/2
            begin
                sck_cnt <= 0;
// reset serial clock counter
                sck <= ~sck;
// toggle serial clock
            end
            else
                sck_cnt <= sck_cnt + 1;
// increment serial clock counter

            sck_d1 <= sck;
// generate 1 cycle delay of sck
        end
    end


    // Creates a delay of word select signal, used to help in comparison test
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            ws_d1 <= 0;
            ws_d2 <= 0;
        end
        else if(i2so_sck_transition)
        begin
            ws_d1 <= i2so_ws;
// generate 1 cycle delay of i2so_ws
            ws_d2 <= ws_d1;
// generate 2nd cycle delay of i2so_ws
        end
    end

    // Queue words into FIFO and stores word into an array for comparison
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            word_count <= 0;
        end
        // If word count is greater than the number of elements being queued, stop attempting to
queue data
        else if(word_count > `NUM_ELEMENTS-1)
        begin
            word_count <= word_count + 1;
        end
        // count % 2551 was determined by guess and check. Wanted to queue data slow enough so
buffer doesn't
        // become full. Also wanted to ensure every time ws_transition is high the variable word
is the next queued word
        // if using a number larger then 2551, word may not change its value to the next queued
word when ws_transition is high
```

```
        else if(count % 1001 == 0)
        begin
            queue(test_data [word_count]);
//indicates when to queue more data
        end
    end


    assign rst_n = !(count < 20);
// turn on reset after 10 clock cycles
    assign i2so_sync_sck = sck;
// assign serial clock input value
    assign i2so_sck_transition = sck & ~sck_d1;
// level to pulse converter when sck goes from low to high
    assign ws_transition = ~ws_d1 & ws_d2;
// level to pulse converter when ws goes from high to low


    // Creates 32 bit words from the serial data being outputted to be compared with the words in
filt_data_list
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            word <= 32'b0;
        end
        else if(i2so_sck_transition)
        begin
            word[31:1] <= word[30:0];
            word[0] <= i2so_sd;
        end
    end


    // Checking if data is properly serialized
    always @(posedge clk)
    begin
        if(ws_transition && i2so_sck_transition)
        begin
            // Find first input word that matches output word
            if(!match_found)
            begin
                for(match_count = 0; match_count < `NUM_ELEMENTS; match_count = match_count + 1)
                begin
                    if(word == filt_data_list[match_count])
                    begin
                        match_found = 1;
// ends loop to find matching words
                        begin_comparison = 1;
// booolean to begin comparison test
                        comparison_failed = 0;
// ensures failed message does not output
                        filt_data_count = match_count;
// sets which test_count in the array the comparison test should begin
                    end
                end
            end
            // No matches were found. End trying to find a match
            if(count > 30000 && !begin_comparison && comparison_failed)
            begin
                match_found = 1;
                comparison_failed = 0;
                $fdisplay (out, "No matches found. Comparison test failed");
                #1 $fclose(out);
            end

            // Initial match found. Begin comparison test
            if(begin_comparison)
            begin
                // If words inputted and outputted match
                if(word == filt_data_list[filt_data_count])
```

```
                begin
                    pass_count = pass_count + 1;
                    $fdisplay (out, "Input: %h", word, "     ---      ",
                        "Output: %h", filt_data_list [filt_data_count],
                        "      ---      Pass");
                end
                // End comparison test after 15 words
                else if(filt_data_count > (`NUM_ELEMENTS-1))
                begin
                    begin_comparison = 0;
                    $fdisplay(out, "\nNumber of Comparisons:                    %d", pass_count +
fail_count,
                        "\nNumber of Successful Comparisons:         %d", pass_count,
                        "\nNumber of Failed Comparisons:           %d", fail_count);
                    $fclose(out);
                end
                // If words inputted and outputted do not match
                else
                begin
                    fail_count = fail_count + 1;
                    $fdisplay (out, "Input: %h", word, "     ---      ",
                        "Output: %h", filt_data_list [filt_data_count],
                        "      ---      Fail");
                end
                filt_data_count = filt_data_count + 1;
            end
        end
    end




    // Defining queue task for FIFO
    task queue;
// define the queue task
    input[31:0] data;
// the data to be queueed
        if(!filt_rtr)
// if buffer is full display warning
                $display("---Cannot queue: Buffer Full---");
            else
// if buffer is not full
            begin
                $display("Queued %h",data );
// display that the data was queueed
                filt_data = data;
// the input to the buffer is set as the data
                filt_rts = 1;
// write is enabled
                word_count <= word_count + 1;
//indicates which data to queue
                filt_data_list [word_count] <= filt_data;
//stores the word that is outputted by the fifo for comparison
                words_queued <= words_queued + 1;
                @(posedge clk);
// checks if clock is at postive edge
                #1 filt_rts = 0;
// set write enabled equal to zero then
            end
    endtask

endmodule
```

serializer_testbench.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:   serializer_testbench.v
// Create Date:   2/14/2016
// Last Edit:     3/20/16
// Author:        Kevin Cao
//
// Description:     Creates N number 32 bit words specified by the programmer to be inputted.
//                  Compares the inputted and outputted words.
//                  Outputs the success and failure of the comparisons in i2s_in_test_output.txt
//
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module serializer_testbench;

          // Inputs
          reg clk;
          reg rst_n;
          reg filt_i2so_rts;
          reg [15:0] filt_i2so_lft;
          reg [15:0] filt_i2so_rgt;
          reg sck_transition;

          // Outputs
          wire i2so_sd;
          wire i2so_ws;
          wire filt_i2so_rtr;

          // Instantiate the Unit Under Test (UUT)
          i2so_serializer uut (
                    .clk(clk),
                    .rst_n(rst_n),
                    .filt_i2so_rts(filt_i2so_rts),
                    .i2so_sd(i2so_sd),
                    .i2so_ws(i2so_ws),
                    .filt_i2so_lft(filt_i2so_lft),
                    .filt_i2so_rgt(filt_i2so_rgt),
                    .filt_i2so_rtr(filt_i2so_rtr),
                    .sck_transition(sck_transition)
          );

          initial begin
                    // Initialize Inputs
                    clk = 0;
                    rst_n = 0;
                    filt_i2so_rts = 0;
                    filt_i2so_lft = 0;
                    filt_i2so_rgt = 0;
                    sck_transition = 0;

                    // Wait 100 ns for global reset to finish
                    #100;

                    // Add stimulus here

          end

endmodule
```

reg_testbench.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             reg_testbench.v
// Create Date:             2/1/2016
// Last Modification:       3/16/2016
// Author:                  Julie Swift
// Description: ???????????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module reg_testbench;

          // Inputs
          reg rst_n;
          reg clk;
          reg [10:0] addr;
          reg [7:0] wdata;
          reg w_enable;
          reg wxfc;
          reg ro_fifo_underrun;
          reg ro_fifo_overrun;
          reg ro_filter_ovf_flag;

          // Outputs
          wire rxfc;
          wire [7:0] rdata;
          wire rf_i2si_bist_en;
          wire [2:0] rf_filter_shift;
          wire rf_filter_clip_en;
          wire trig_fifo_overrun;
          wire trig_fifo_underrun;
          wire trig_filter_ovf_flag_clear;
          wire [7:0] rf_i2si_bist_start_val_a;
          wire [3:0] rf_i2si_bist_start_val_b;
          wire [7:0] rf_i2si_bist_incr;
          wire [7:0] rf_i2si_bist_upper_limit_a;
          wire [3:0] rf_i2si_bist_upper_limit_b;
          wire [7:0] rf_filter_coeff0_a;
          wire [7:0] rf_filter_coeff0_b;
          wire [7:0] rf_filter_coeff1_a;
          wire [7:0] rf_filter_coeff1_b;
          wire [7:0] rf_filter_coeff2_a;
          wire [7:0] rf_filter_coeff2_b;
          wire [7:0] rf_filter_coeff3_a;
          wire [7:0] rf_filter_coeff3_b;
          wire [7:0] rf_filter_coeff4_a;
          wire [7:0] rf_filter_coeff4_b;
          wire [7:0] rf_filter_coeff5_a;
          wire [7:0] rf_filter_coeff5_b;
          wire [7:0] rf_filter_coeff6_a;
          wire [7:0] rf_filter_coeff6_b;
          wire [7:0] rf_filter_coeff7_a;
          wire [7:0] rf_filter_coeff7_b;
          wire [7:0] rf_filter_coeff8_a;
          wire [7:0] rf_filter_coeff8_b;
          wire [7:0] rf_filter_coeff9_a;
          wire [7:0] rf_filter_coeff9_b;
          wire [7:0] rf_filter_coeff10_a;
          wire [7:0] rf_filter_coeff10_b;
          wire [7:0] rf_filter_coeff11_a;
          wire [7:0] rf_filter_coeff11_b;
          wire [7:0] rf_filter_coeff12_a;
          wire [7:0] rf_filter_coeff12_b;
          wire [7:0] rf_filter_coeff13_a;
          wire [7:0] rf_filter_coeff13_b;
          wire [7:0] rf_filter_coeff14_a;
          wire [7:0] rf_filter_coeff14_b;
          wire [7:0] rf_filter_coeff15_a;
          wire [7:0] rf_filter_coeff15_b;
```

```
wire [7:0] rf_filter_coeff16_a;
wire [7:0] rf_filter_coeff16_b;
wire [7:0] rf_filter_coeff17_a;
wire [7:0] rf_filter_coeff17_b;
wire [7:0] rf_filter_coeff18_a;
wire [7:0] rf_filter_coeff18_b;
wire [7:0] rf_filter_coeff19_a;
wire [7:0] rf_filter_coeff19_b;
wire [7:0] rf_filter_coeff20_a;
wire [7:0] rf_filter_coeff20_b;
wire [7:0] rf_filter_coeff21_a;
wire [7:0] rf_filter_coeff21_b;
wire [7:0] rf_filter_coeff22_a;
wire [7:0] rf_filter_coeff22_b;
wire [7:0] rf_filter_coeff23_a;
wire [7:0] rf_filter_coeff23_b;
wire [7:0] rf_filter_coeff24_a;
wire [7:0] rf_filter_coeff24_b;
wire [7:0] rf_filter_coeff25_a;
wire [7:0] rf_filter_coeff25_b;
wire [7:0] rf_filter_coeff26_a;
wire [7:0] rf_filter_coeff26_b;
wire [7:0] rf_filter_coeff27_a;
wire [7:0] rf_filter_coeff27_b;
wire [7:0] rf_filter_coeff28_a;
wire [7:0] rf_filter_coeff28_b;
wire [7:0] rf_filter_coeff29_a;
wire [7:0] rf_filter_coeff29_b;
wire [7:0] rf_filter_coeff30_a;
wire [7:0] rf_filter_coeff30_b;
wire [7:0] rf_filter_coeff31_a;
wire [7:0] rf_filter_coeff31_b;
wire [7:0] rf_filter_coeff32_a;
wire [7:0] rf_filter_coeff32_b;
wire [7:0] rf_filter_coeff33_a;
wire [7:0] rf_filter_coeff33_b;
wire [7:0] rf_filter_coeff34_a;
wire [7:0] rf_filter_coeff34_b;
wire [7:0] rf_filter_coeff35_a;
wire [7:0] rf_filter_coeff35_b;
wire [7:0] rf_filter_coeff36_a;
wire [7:0] rf_filter_coeff36_b;
wire [7:0] rf_filter_coeff37_a;
wire [7:0] rf_filter_coeff37_b;
wire [7:0] rf_filter_coeff38_a;
wire [7:0] rf_filter_coeff38_b;
wire [7:0] rf_filter_coeff39_a;
wire [7:0] rf_filter_coeff39_b;
wire [7:0] rf_filter_coeff40_a;
wire [7:0] rf_filter_coeff40_b;
wire [7:0] rf_filter_coeff41_a;
wire [7:0] rf_filter_coeff41_b;
wire [7:0] rf_filter_coeff42_a;
wire [7:0] rf_filter_coeff42_b;
wire [7:0] rf_filter_coeff43_a;
wire [7:0] rf_filter_coeff43_b;
wire [7:0] rf_filter_coeff44_a;
wire [7:0] rf_filter_coeff44_b;
wire [7:0] rf_filter_coeff45_a;
wire [7:0] rf_filter_coeff45_b;
wire [7:0] rf_filter_coeff46_a;
wire [7:0] rf_filter_coeff46_b;
wire [7:0] rf_filter_coeff47_a;
wire [7:0] rf_filter_coeff47_b;
wire [7:0] rf_filter_coeff48_a;
wire [7:0] rf_filter_coeff48_b;
wire [7:0] rf_filter_coeff49_a;
wire [7:0] rf_filter_coeff49_b;
wire [7:0] rf_filter_coeff50_a;
wire [7:0] rf_filter_coeff50_b;
wire [7:0] rf_filter_coeff51_a;
```

```
wire [7:0] rf_filter_coeff51_b;
wire [7:0] rf_filter_coeff52_a;
wire [7:0] rf_filter_coeff52_b;
wire [7:0] rf_filter_coeff53_a;
wire [7:0] rf_filter_coeff53_b;
wire [7:0] rf_filter_coeff54_a;
wire [7:0] rf_filter_coeff54_b;
wire [7:0] rf_filter_coeff55_a;
wire [7:0] rf_filter_coeff55_b;
wire [7:0] rf_filter_coeff56_a;
wire [7:0] rf_filter_coeff56_b;
wire [7:0] rf_filter_coeff57_a;
wire [7:0] rf_filter_coeff57_b;
wire [7:0] rf_filter_coeff58_a;
wire [7:0] rf_filter_coeff58_b;
wire [7:0] rf_filter_coeff59_a;
wire [7:0] rf_filter_coeff59_b;
wire [7:0] rf_filter_coeff60_a;
wire [7:0] rf_filter_coeff60_b;
wire [7:0] rf_filter_coeff61_a;
wire [7:0] rf_filter_coeff61_b;
wire [7:0] rf_filter_coeff62_a;
wire [7:0] rf_filter_coeff62_b;
wire [7:0] rf_filter_coeff63_a;
wire [7:0] rf_filter_coeff63_b;
wire [7:0] rf_filter_coeff64_a;
wire [7:0] rf_filter_coeff64_b;
wire [7:0] rf_filter_coeff65_a;
wire [7:0] rf_filter_coeff65_b;
wire [7:0] rf_filter_coeff66_a;
wire [7:0] rf_filter_coeff66_b;
wire [7:0] rf_filter_coeff67_a;
wire [7:0] rf_filter_coeff67_b;
wire [7:0] rf_filter_coeff68_a;
wire [7:0] rf_filter_coeff68_b;
wire [7:0] rf_filter_coeff69_a;
wire [7:0] rf_filter_coeff69_b;
wire [7:0] rf_filter_coeff70_a;
wire [7:0] rf_filter_coeff70_b;
wire [7:0] rf_filter_coeff71_a;
wire [7:0] rf_filter_coeff71_b;
wire [7:0] rf_filter_coeff72_a;
wire [7:0] rf_filter_coeff72_b;
wire [7:0] rf_filter_coeff73_a;
wire [7:0] rf_filter_coeff73_b;
wire [7:0] rf_filter_coeff74_a;
wire [7:0] rf_filter_coeff74_b;
wire [7:0] rf_filter_coeff75_a;
wire [7:0] rf_filter_coeff75_b;
wire [7:0] rf_filter_coeff76_a;
wire [7:0] rf_filter_coeff76_b;
wire [7:0] rf_filter_coeff77_a;
wire [7:0] rf_filter_coeff77_b;
wire [7:0] rf_filter_coeff78_a;
wire [7:0] rf_filter_coeff78_b;
wire [7:0] rf_filter_coeff79_a;
wire [7:0] rf_filter_coeff79_b;
wire [7:0] rf_filter_coeff80_a;
wire [7:0] rf_filter_coeff80_b;
wire [7:0] rf_filter_coeff81_a;
wire [7:0] rf_filter_coeff81_b;
wire [7:0] rf_filter_coeff82_a;
wire [7:0] rf_filter_coeff82_b;
wire [7:0] rf_filter_coeff83_a;
wire [7:0] rf_filter_coeff83_b;
wire [7:0] rf_filter_coeff84_a;
wire [7:0] rf_filter_coeff84_b;
wire [7:0] rf_filter_coeff85_a;
wire [7:0] rf_filter_coeff85_b;
wire [7:0] rf_filter_coeff86_a;
wire [7:0] rf_filter_coeff86_b;
```

```
wire [7:0] rf_filter_coeff87_a;
wire [7:0] rf_filter_coeff87_b;
wire [7:0] rf_filter_coeff88_a;
wire [7:0] rf_filter_coeff88_b;
wire [7:0] rf_filter_coeff89_a;
wire [7:0] rf_filter_coeff89_b;
wire [7:0] rf_filter_coeff90_a;
wire [7:0] rf_filter_coeff90_b;
wire [7:0] rf_filter_coeff91_a;
wire [7:0] rf_filter_coeff91_b;
wire [7:0] rf_filter_coeff92_a;
wire [7:0] rf_filter_coeff92_b;
wire [7:0] rf_filter_coeff93_a;
wire [7:0] rf_filter_coeff93_b;
wire [7:0] rf_filter_coeff94_a;
wire [7:0] rf_filter_coeff94_b;
wire [7:0] rf_filter_coeff95_a;
wire [7:0] rf_filter_coeff95_b;
wire [7:0] rf_filter_coeff96_a;
wire [7:0] rf_filter_coeff96_b;
wire [7:0] rf_filter_coeff97_a;
wire [7:0] rf_filter_coeff97_b;
wire [7:0] rf_filter_coeff98_a;
wire [7:0] rf_filter_coeff98_b;
wire [7:0] rf_filter_coeff99_a;
wire [7:0] rf_filter_coeff99_b;
wire [7:0] rf_filter_coeff100_a;
wire [7:0] rf_filter_coeff100_b;
wire [7:0] rf_filter_coeff101_a;
wire [7:0] rf_filter_coeff101_b;
wire [7:0] rf_filter_coeff102_a;
wire [7:0] rf_filter_coeff102_b;
wire [7:0] rf_filter_coeff103_a;
wire [7:0] rf_filter_coeff103_b;
wire [7:0] rf_filter_coeff104_a;
wire [7:0] rf_filter_coeff104_b;
wire [7:0] rf_filter_coeff105_a;
wire [7:0] rf_filter_coeff105_b;
wire [7:0] rf_filter_coeff106_a;
wire [7:0] rf_filter_coeff106_b;
wire [7:0] rf_filter_coeff107_a;
wire [7:0] rf_filter_coeff107_b;
wire [7:0] rf_filter_coeff108_a;
wire [7:0] rf_filter_coeff108_b;
wire [7:0] rf_filter_coeff109_a;
wire [7:0] rf_filter_coeff109_b;
wire [7:0] rf_filter_coeff110_a;
wire [7:0] rf_filter_coeff110_b;
wire [7:0] rf_filter_coeff111_a;
wire [7:0] rf_filter_coeff111_b;
wire [7:0] rf_filter_coeff112_a;
wire [7:0] rf_filter_coeff112_b;
wire [7:0] rf_filter_coeff113_a;
wire [7:0] rf_filter_coeff113_b;
wire [7:0] rf_filter_coeff114_a;
wire [7:0] rf_filter_coeff114_b;
wire [7:0] rf_filter_coeff115_a;
wire [7:0] rf_filter_coeff115_b;
wire [7:0] rf_filter_coeff116_a;
wire [7:0] rf_filter_coeff116_b;
wire [7:0] rf_filter_coeff117_a;
wire [7:0] rf_filter_coeff117_b;
wire [7:0] rf_filter_coeff118_a;
wire [7:0] rf_filter_coeff118_b;
wire [7:0] rf_filter_coeff119_a;
wire [7:0] rf_filter_coeff119_b;
wire [7:0] rf_filter_coeff120_a;
wire [7:0] rf_filter_coeff120_b;
wire [7:0] rf_filter_coeff121_a;
wire [7:0] rf_filter_coeff121_b;
wire [7:0] rf_filter_coeff122_a;
```

```
wire [7:0] rf_filter_coeff122_b;
wire [7:0] rf_filter_coeff123_a;
wire [7:0] rf_filter_coeff123_b;
wire [7:0] rf_filter_coeff124_a;
wire [7:0] rf_filter_coeff124_b;
wire [7:0] rf_filter_coeff125_a;
wire [7:0] rf_filter_coeff125_b;
wire [7:0] rf_filter_coeff126_a;
wire [7:0] rf_filter_coeff126_b;
wire [7:0] rf_filter_coeff127_a;
wire [7:0] rf_filter_coeff127_b;
wire [7:0] rf_filter_coeff128_a;
wire [7:0] rf_filter_coeff128_b;
wire [7:0] rf_filter_coeff129_a;
wire [7:0] rf_filter_coeff129_b;
wire [7:0] rf_filter_coeff130_a;
wire [7:0] rf_filter_coeff130_b;
wire [7:0] rf_filter_coeff131_a;
wire [7:0] rf_filter_coeff131_b;
wire [7:0] rf_filter_coeff132_a;
wire [7:0] rf_filter_coeff132_b;
wire [7:0] rf_filter_coeff133_a;
wire [7:0] rf_filter_coeff133_b;
wire [7:0] rf_filter_coeff134_a;
wire [7:0] rf_filter_coeff134_b;
wire [7:0] rf_filter_coeff135_a;
wire [7:0] rf_filter_coeff135_b;
wire [7:0] rf_filter_coeff136_a;
wire [7:0] rf_filter_coeff136_b;
wire [7:0] rf_filter_coeff137_a;
wire [7:0] rf_filter_coeff137_b;
wire [7:0] rf_filter_coeff138_a;
wire [7:0] rf_filter_coeff138_b;
wire [7:0] rf_filter_coeff139_a;
wire [7:0] rf_filter_coeff139_b;
wire [7:0] rf_filter_coeff140_a;
wire [7:0] rf_filter_coeff140_b;
wire [7:0] rf_filter_coeff141_a;
wire [7:0] rf_filter_coeff141_b;
wire [7:0] rf_filter_coeff142_a;
wire [7:0] rf_filter_coeff142_b;
wire [7:0] rf_filter_coeff143_a;
wire [7:0] rf_filter_coeff143_b;
wire [7:0] rf_filter_coeff144_a;
wire [7:0] rf_filter_coeff144_b;
wire [7:0] rf_filter_coeff145_a;
wire [7:0] rf_filter_coeff145_b;
wire [7:0] rf_filter_coeff146_a;
wire [7:0] rf_filter_coeff146_b;
wire [7:0] rf_filter_coeff147_a;
wire [7:0] rf_filter_coeff147_b;
wire [7:0] rf_filter_coeff148_a;
wire [7:0] rf_filter_coeff148_b;
wire [7:0] rf_filter_coeff149_a;
wire [7:0] rf_filter_coeff149_b;
wire [7:0] rf_filter_coeff150_a;
wire [7:0] rf_filter_coeff150_b;
wire [7:0] rf_filter_coeff151_a;
wire [7:0] rf_filter_coeff151_b;
wire [7:0] rf_filter_coeff152_a;
wire [7:0] rf_filter_coeff152_b;
wire [7:0] rf_filter_coeff153_a;
wire [7:0] rf_filter_coeff153_b;
wire [7:0] rf_filter_coeff154_a;
wire [7:0] rf_filter_coeff154_b;
wire [7:0] rf_filter_coeff155_a;
wire [7:0] rf_filter_coeff155_b;
wire [7:0] rf_filter_coeff156_a;
wire [7:0] rf_filter_coeff156_b;
wire [7:0] rf_filter_coeff157_a;
wire [7:0] rf_filter_coeff157_b;
```

```
wire [7:0] rf_filter_coeff158_a;
wire [7:0] rf_filter_coeff158_b;
wire [7:0] rf_filter_coeff159_a;
wire [7:0] rf_filter_coeff159_b;
wire [7:0] rf_filter_coeff160_a;
wire [7:0] rf_filter_coeff160_b;
wire [7:0] rf_filter_coeff161_a;
wire [7:0] rf_filter_coeff161_b;
wire [7:0] rf_filter_coeff162_a;
wire [7:0] rf_filter_coeff162_b;
wire [7:0] rf_filter_coeff163_a;
wire [7:0] rf_filter_coeff163_b;
wire [7:0] rf_filter_coeff164_a;
wire [7:0] rf_filter_coeff164_b;
wire [7:0] rf_filter_coeff165_a;
wire [7:0] rf_filter_coeff165_b;
wire [7:0] rf_filter_coeff166_a;
wire [7:0] rf_filter_coeff166_b;
wire [7:0] rf_filter_coeff167_a;
wire [7:0] rf_filter_coeff167_b;
wire [7:0] rf_filter_coeff168_a;
wire [7:0] rf_filter_coeff168_b;
wire [7:0] rf_filter_coeff169_a;
wire [7:0] rf_filter_coeff169_b;
wire [7:0] rf_filter_coeff170_a;
wire [7:0] rf_filter_coeff170_b;
wire [7:0] rf_filter_coeff171_a;
wire [7:0] rf_filter_coeff171_b;
wire [7:0] rf_filter_coeff172_a;
wire [7:0] rf_filter_coeff172_b;
wire [7:0] rf_filter_coeff173_a;
wire [7:0] rf_filter_coeff173_b;
wire [7:0] rf_filter_coeff174_a;
wire [7:0] rf_filter_coeff174_b;
wire [7:0] rf_filter_coeff175_a;
wire [7:0] rf_filter_coeff175_b;
wire [7:0] rf_filter_coeff176_a;
wire [7:0] rf_filter_coeff176_b;
wire [7:0] rf_filter_coeff177_a;
wire [7:0] rf_filter_coeff177_b;
wire [7:0] rf_filter_coeff178_a;
wire [7:0] rf_filter_coeff178_b;
wire [7:0] rf_filter_coeff179_a;
wire [7:0] rf_filter_coeff179_b;
wire [7:0] rf_filter_coeff180_a;
wire [7:0] rf_filter_coeff180_b;
wire [7:0] rf_filter_coeff181_a;
wire [7:0] rf_filter_coeff181_b;
wire [7:0] rf_filter_coeff182_a;
wire [7:0] rf_filter_coeff182_b;
wire [7:0] rf_filter_coeff183_a;
wire [7:0] rf_filter_coeff183_b;
wire [7:0] rf_filter_coeff184_a;
wire [7:0] rf_filter_coeff184_b;
wire [7:0] rf_filter_coeff185_a;
wire [7:0] rf_filter_coeff185_b;
wire [7:0] rf_filter_coeff186_a;
wire [7:0] rf_filter_coeff186_b;
wire [7:0] rf_filter_coeff187_a;
wire [7:0] rf_filter_coeff187_b;
wire [7:0] rf_filter_coeff188_a;
wire [7:0] rf_filter_coeff188_b;
wire [7:0] rf_filter_coeff189_a;
wire [7:0] rf_filter_coeff189_b;
wire [7:0] rf_filter_coeff190_a;
wire [7:0] rf_filter_coeff190_b;
wire [7:0] rf_filter_coeff191_a;
wire [7:0] rf_filter_coeff191_b;
wire [7:0] rf_filter_coeff192_a;
wire [7:0] rf_filter_coeff192_b;
wire [7:0] rf_filter_coeff193_a;
```

```
wire [7:0] rf_filter_coeff193_b;
wire [7:0] rf_filter_coeff194_a;
wire [7:0] rf_filter_coeff194_b;
wire [7:0] rf_filter_coeff195_a;
wire [7:0] rf_filter_coeff195_b;
wire [7:0] rf_filter_coeff196_a;
wire [7:0] rf_filter_coeff196_b;
wire [7:0] rf_filter_coeff197_a;
wire [7:0] rf_filter_coeff197_b;
wire [7:0] rf_filter_coeff198_a;
wire [7:0] rf_filter_coeff198_b;
wire [7:0] rf_filter_coeff199_a;
wire [7:0] rf_filter_coeff199_b;
wire [7:0] rf_filter_coeff200_a;
wire [7:0] rf_filter_coeff200_b;
wire [7:0] rf_filter_coeff201_a;
wire [7:0] rf_filter_coeff201_b;
wire [7:0] rf_filter_coeff202_a;
wire [7:0] rf_filter_coeff202_b;
wire [7:0] rf_filter_coeff203_a;
wire [7:0] rf_filter_coeff203_b;
wire [7:0] rf_filter_coeff204_a;
wire [7:0] rf_filter_coeff204_b;
wire [7:0] rf_filter_coeff205_a;
wire [7:0] rf_filter_coeff205_b;
wire [7:0] rf_filter_coeff206_a;
wire [7:0] rf_filter_coeff206_b;
wire [7:0] rf_filter_coeff207_a;
wire [7:0] rf_filter_coeff207_b;
wire [7:0] rf_filter_coeff208_a;
wire [7:0] rf_filter_coeff208_b;
wire [7:0] rf_filter_coeff209_a;
wire [7:0] rf_filter_coeff209_b;
wire [7:0] rf_filter_coeff210_a;
wire [7:0] rf_filter_coeff210_b;
wire [7:0] rf_filter_coeff211_a;
wire [7:0] rf_filter_coeff211_b;
wire [7:0] rf_filter_coeff212_a;
wire [7:0] rf_filter_coeff212_b;
wire [7:0] rf_filter_coeff213_a;
wire [7:0] rf_filter_coeff213_b;
wire [7:0] rf_filter_coeff214_a;
wire [7:0] rf_filter_coeff214_b;
wire [7:0] rf_filter_coeff215_a;
wire [7:0] rf_filter_coeff215_b;
wire [7:0] rf_filter_coeff216_a;
wire [7:0] rf_filter_coeff216_b;
wire [7:0] rf_filter_coeff217_a;
wire [7:0] rf_filter_coeff217_b;
wire [7:0] rf_filter_coeff218_a;
wire [7:0] rf_filter_coeff218_b;
wire [7:0] rf_filter_coeff219_a;
wire [7:0] rf_filter_coeff219_b;
wire [7:0] rf_filter_coeff220_a;
wire [7:0] rf_filter_coeff220_b;
wire [7:0] rf_filter_coeff221_a;
wire [7:0] rf_filter_coeff221_b;
wire [7:0] rf_filter_coeff222_a;
wire [7:0] rf_filter_coeff222_b;
wire [7:0] rf_filter_coeff223_a;
wire [7:0] rf_filter_coeff223_b;
wire [7:0] rf_filter_coeff224_a;
wire [7:0] rf_filter_coeff224_b;
wire [7:0] rf_filter_coeff225_a;
wire [7:0] rf_filter_coeff225_b;
wire [7:0] rf_filter_coeff226_a;
wire [7:0] rf_filter_coeff226_b;
wire [7:0] rf_filter_coeff227_a;
wire [7:0] rf_filter_coeff227_b;
wire [7:0] rf_filter_coeff228_a;
wire [7:0] rf_filter_coeff228_b;
```

```
wire [7:0] rf_filter_coeff229_a;
wire [7:0] rf_filter_coeff229_b;
wire [7:0] rf_filter_coeff230_a;
wire [7:0] rf_filter_coeff230_b;
wire [7:0] rf_filter_coeff231_a;
wire [7:0] rf_filter_coeff231_b;
wire [7:0] rf_filter_coeff232_a;
wire [7:0] rf_filter_coeff232_b;
wire [7:0] rf_filter_coeff233_a;
wire [7:0] rf_filter_coeff233_b;
wire [7:0] rf_filter_coeff234_a;
wire [7:0] rf_filter_coeff234_b;
wire [7:0] rf_filter_coeff235_a;
wire [7:0] rf_filter_coeff235_b;
wire [7:0] rf_filter_coeff236_a;
wire [7:0] rf_filter_coeff236_b;
wire [7:0] rf_filter_coeff237_a;
wire [7:0] rf_filter_coeff237_b;
wire [7:0] rf_filter_coeff238_a;
wire [7:0] rf_filter_coeff238_b;
wire [7:0] rf_filter_coeff239_a;
wire [7:0] rf_filter_coeff239_b;
wire [7:0] rf_filter_coeff240_a;
wire [7:0] rf_filter_coeff240_b;
wire [7:0] rf_filter_coeff241_a;
wire [7:0] rf_filter_coeff241_b;
wire [7:0] rf_filter_coeff242_a;
wire [7:0] rf_filter_coeff242_b;
wire [7:0] rf_filter_coeff243_a;
wire [7:0] rf_filter_coeff243_b;
wire [7:0] rf_filter_coeff244_a;
wire [7:0] rf_filter_coeff244_b;
wire [7:0] rf_filter_coeff245_a;
wire [7:0] rf_filter_coeff245_b;
wire [7:0] rf_filter_coeff246_a;
wire [7:0] rf_filter_coeff246_b;
wire [7:0] rf_filter_coeff247_a;
wire [7:0] rf_filter_coeff247_b;
wire [7:0] rf_filter_coeff248_a;
wire [7:0] rf_filter_coeff248_b;
wire [7:0] rf_filter_coeff249_a;
wire [7:0] rf_filter_coeff249_b;
wire [7:0] rf_filter_coeff250_a;
wire [7:0] rf_filter_coeff250_b;
wire [7:0] rf_filter_coeff251_a;
wire [7:0] rf_filter_coeff251_b;
wire [7:0] rf_filter_coeff252_a;
wire [7:0] rf_filter_coeff252_b;
wire [7:0] rf_filter_coeff253_a;
wire [7:0] rf_filter_coeff253_b;
wire [7:0] rf_filter_coeff254_a;
wire [7:0] rf_filter_coeff254_b;
wire [7:0] rf_filter_coeff255_a;
wire [7:0] rf_filter_coeff255_b;
wire [7:0] rf_filter_coeff256_a;
wire [7:0] rf_filter_coeff256_b;
wire [7:0] rf_filter_coeff257_a;
wire [7:0] rf_filter_coeff257_b;
wire [7:0] rf_filter_coeff258_a;
wire [7:0] rf_filter_coeff258_b;
wire [7:0] rf_filter_coeff259_a;
wire [7:0] rf_filter_coeff259_b;
wire [7:0] rf_filter_coeff260_a;
wire [7:0] rf_filter_coeff260_b;
wire [7:0] rf_filter_coeff261_a;
wire [7:0] rf_filter_coeff261_b;
wire [7:0] rf_filter_coeff262_a;
wire [7:0] rf_filter_coeff262_b;
wire [7:0] rf_filter_coeff263_a;
wire [7:0] rf_filter_coeff263_b;
wire [7:0] rf_filter_coeff264_a;
```

```
wire [7:0] rf_filter_coeff264_b;
wire [7:0] rf_filter_coeff265_a;
wire [7:0] rf_filter_coeff265_b;
wire [7:0] rf_filter_coeff266_a;
wire [7:0] rf_filter_coeff266_b;
wire [7:0] rf_filter_coeff267_a;
wire [7:0] rf_filter_coeff267_b;
wire [7:0] rf_filter_coeff268_a;
wire [7:0] rf_filter_coeff268_b;
wire [7:0] rf_filter_coeff269_a;
wire [7:0] rf_filter_coeff269_b;
wire [7:0] rf_filter_coeff270_a;
wire [7:0] rf_filter_coeff270_b;
wire [7:0] rf_filter_coeff271_a;
wire [7:0] rf_filter_coeff271_b;
wire [7:0] rf_filter_coeff272_a;
wire [7:0] rf_filter_coeff272_b;
wire [7:0] rf_filter_coeff273_a;
wire [7:0] rf_filter_coeff273_b;
wire [7:0] rf_filter_coeff274_a;
wire [7:0] rf_filter_coeff274_b;
wire [7:0] rf_filter_coeff275_a;
wire [7:0] rf_filter_coeff275_b;
wire [7:0] rf_filter_coeff276_a;
wire [7:0] rf_filter_coeff276_b;
wire [7:0] rf_filter_coeff277_a;
wire [7:0] rf_filter_coeff277_b;
wire [7:0] rf_filter_coeff278_a;
wire [7:0] rf_filter_coeff278_b;
wire [7:0] rf_filter_coeff279_a;
wire [7:0] rf_filter_coeff279_b;
wire [7:0] rf_filter_coeff280_a;
wire [7:0] rf_filter_coeff280_b;
wire [7:0] rf_filter_coeff281_a;
wire [7:0] rf_filter_coeff281_b;
wire [7:0] rf_filter_coeff282_a;
wire [7:0] rf_filter_coeff282_b;
wire [7:0] rf_filter_coeff283_a;
wire [7:0] rf_filter_coeff283_b;
wire [7:0] rf_filter_coeff284_a;
wire [7:0] rf_filter_coeff284_b;
wire [7:0] rf_filter_coeff285_a;
wire [7:0] rf_filter_coeff285_b;
wire [7:0] rf_filter_coeff286_a;
wire [7:0] rf_filter_coeff286_b;
wire [7:0] rf_filter_coeff287_a;
wire [7:0] rf_filter_coeff287_b;
wire [7:0] rf_filter_coeff288_a;
wire [7:0] rf_filter_coeff288_b;
wire [7:0] rf_filter_coeff289_a;
wire [7:0] rf_filter_coeff289_b;
wire [7:0] rf_filter_coeff290_a;
wire [7:0] rf_filter_coeff290_b;
wire [7:0] rf_filter_coeff291_a;
wire [7:0] rf_filter_coeff291_b;
wire [7:0] rf_filter_coeff292_a;
wire [7:0] rf_filter_coeff292_b;
wire [7:0] rf_filter_coeff293_a;
wire [7:0] rf_filter_coeff293_b;
wire [7:0] rf_filter_coeff294_a;
wire [7:0] rf_filter_coeff294_b;
wire [7:0] rf_filter_coeff295_a;
wire [7:0] rf_filter_coeff295_b;
wire [7:0] rf_filter_coeff296_a;
wire [7:0] rf_filter_coeff296_b;
wire [7:0] rf_filter_coeff297_a;
wire [7:0] rf_filter_coeff297_b;
wire [7:0] rf_filter_coeff298_a;
wire [7:0] rf_filter_coeff298_b;
wire [7:0] rf_filter_coeff299_a;
wire [7:0] rf_filter_coeff299_b;
```

```
wire [7:0] rf_filter_coeff300_a;
wire [7:0] rf_filter_coeff300_b;
wire [7:0] rf_filter_coeff301_a;
wire [7:0] rf_filter_coeff301_b;
wire [7:0] rf_filter_coeff302_a;
wire [7:0] rf_filter_coeff302_b;
wire [7:0] rf_filter_coeff303_a;
wire [7:0] rf_filter_coeff303_b;
wire [7:0] rf_filter_coeff304_a;
wire [7:0] rf_filter_coeff304_b;
wire [7:0] rf_filter_coeff305_a;
wire [7:0] rf_filter_coeff305_b;
wire [7:0] rf_filter_coeff306_a;
wire [7:0] rf_filter_coeff306_b;
wire [7:0] rf_filter_coeff307_a;
wire [7:0] rf_filter_coeff307_b;
wire [7:0] rf_filter_coeff308_a;
wire [7:0] rf_filter_coeff308_b;
wire [7:0] rf_filter_coeff309_a;
wire [7:0] rf_filter_coeff309_b;
wire [7:0] rf_filter_coeff310_a;
wire [7:0] rf_filter_coeff310_b;
wire [7:0] rf_filter_coeff311_a;
wire [7:0] rf_filter_coeff311_b;
wire [7:0] rf_filter_coeff312_a;
wire [7:0] rf_filter_coeff312_b;
wire [7:0] rf_filter_coeff313_a;
wire [7:0] rf_filter_coeff313_b;
wire [7:0] rf_filter_coeff314_a;
wire [7:0] rf_filter_coeff314_b;
wire [7:0] rf_filter_coeff315_a;
wire [7:0] rf_filter_coeff315_b;
wire [7:0] rf_filter_coeff316_a;
wire [7:0] rf_filter_coeff316_b;
wire [7:0] rf_filter_coeff317_a;
wire [7:0] rf_filter_coeff317_b;
wire [7:0] rf_filter_coeff318_a;
wire [7:0] rf_filter_coeff318_b;
wire [7:0] rf_filter_coeff319_a;
wire [7:0] rf_filter_coeff319_b;
wire [7:0] rf_filter_coeff320_a;
wire [7:0] rf_filter_coeff320_b;
wire [7:0] rf_filter_coeff321_a;
wire [7:0] rf_filter_coeff321_b;
wire [7:0] rf_filter_coeff322_a;
wire [7:0] rf_filter_coeff322_b;
wire [7:0] rf_filter_coeff323_a;
wire [7:0] rf_filter_coeff323_b;
wire [7:0] rf_filter_coeff324_a;
wire [7:0] rf_filter_coeff324_b;
wire [7:0] rf_filter_coeff325_a;
wire [7:0] rf_filter_coeff325_b;
wire [7:0] rf_filter_coeff326_a;
wire [7:0] rf_filter_coeff326_b;
wire [7:0] rf_filter_coeff327_a;
wire [7:0] rf_filter_coeff327_b;
wire [7:0] rf_filter_coeff328_a;
wire [7:0] rf_filter_coeff328_b;
wire [7:0] rf_filter_coeff329_a;
wire [7:0] rf_filter_coeff329_b;
wire [7:0] rf_filter_coeff330_a;
wire [7:0] rf_filter_coeff330_b;
wire [7:0] rf_filter_coeff331_a;
wire [7:0] rf_filter_coeff331_b;
wire [7:0] rf_filter_coeff332_a;
wire [7:0] rf_filter_coeff332_b;
wire [7:0] rf_filter_coeff333_a;
wire [7:0] rf_filter_coeff333_b;
wire [7:0] rf_filter_coeff334_a;
wire [7:0] rf_filter_coeff334_b;
wire [7:0] rf_filter_coeff335_a;
```

```
wire [7:0] rf_filter_coeff335_b;
wire [7:0] rf_filter_coeff336_a;
wire [7:0] rf_filter_coeff336_b;
wire [7:0] rf_filter_coeff337_a;
wire [7:0] rf_filter_coeff337_b;
wire [7:0] rf_filter_coeff338_a;
wire [7:0] rf_filter_coeff338_b;
wire [7:0] rf_filter_coeff339_a;
wire [7:0] rf_filter_coeff339_b;
wire [7:0] rf_filter_coeff340_a;
wire [7:0] rf_filter_coeff340_b;
wire [7:0] rf_filter_coeff341_a;
wire [7:0] rf_filter_coeff341_b;
wire [7:0] rf_filter_coeff342_a;
wire [7:0] rf_filter_coeff342_b;
wire [7:0] rf_filter_coeff343_a;
wire [7:0] rf_filter_coeff343_b;
wire [7:0] rf_filter_coeff344_a;
wire [7:0] rf_filter_coeff344_b;
wire [7:0] rf_filter_coeff345_a;
wire [7:0] rf_filter_coeff345_b;
wire [7:0] rf_filter_coeff346_a;
wire [7:0] rf_filter_coeff346_b;
wire [7:0] rf_filter_coeff347_a;
wire [7:0] rf_filter_coeff347_b;
wire [7:0] rf_filter_coeff348_a;
wire [7:0] rf_filter_coeff348_b;
wire [7:0] rf_filter_coeff349_a;
wire [7:0] rf_filter_coeff349_b;
wire [7:0] rf_filter_coeff350_a;
wire [7:0] rf_filter_coeff350_b;
wire [7:0] rf_filter_coeff351_a;
wire [7:0] rf_filter_coeff351_b;
wire [7:0] rf_filter_coeff352_a;
wire [7:0] rf_filter_coeff352_b;
wire [7:0] rf_filter_coeff353_a;
wire [7:0] rf_filter_coeff353_b;
wire [7:0] rf_filter_coeff354_a;
wire [7:0] rf_filter_coeff354_b;
wire [7:0] rf_filter_coeff355_a;
wire [7:0] rf_filter_coeff355_b;
wire [7:0] rf_filter_coeff356_a;
wire [7:0] rf_filter_coeff356_b;
wire [7:0] rf_filter_coeff357_a;
wire [7:0] rf_filter_coeff357_b;
wire [7:0] rf_filter_coeff358_a;
wire [7:0] rf_filter_coeff358_b;
wire [7:0] rf_filter_coeff359_a;
wire [7:0] rf_filter_coeff359_b;
wire [7:0] rf_filter_coeff360_a;
wire [7:0] rf_filter_coeff360_b;
wire [7:0] rf_filter_coeff361_a;
wire [7:0] rf_filter_coeff361_b;
wire [7:0] rf_filter_coeff362_a;
wire [7:0] rf_filter_coeff362_b;
wire [7:0] rf_filter_coeff363_a;
wire [7:0] rf_filter_coeff363_b;
wire [7:0] rf_filter_coeff364_a;
wire [7:0] rf_filter_coeff364_b;
wire [7:0] rf_filter_coeff365_a;
wire [7:0] rf_filter_coeff365_b;
wire [7:0] rf_filter_coeff366_a;
wire [7:0] rf_filter_coeff366_b;
wire [7:0] rf_filter_coeff367_a;
wire [7:0] rf_filter_coeff367_b;
wire [7:0] rf_filter_coeff368_a;
wire [7:0] rf_filter_coeff368_b;
wire [7:0] rf_filter_coeff369_a;
wire [7:0] rf_filter_coeff369_b;
wire [7:0] rf_filter_coeff370_a;
wire [7:0] rf_filter_coeff370_b;
```

```
wire [7:0] rf_filter_coeff371_a;
wire [7:0] rf_filter_coeff371_b;
wire [7:0] rf_filter_coeff372_a;
wire [7:0] rf_filter_coeff372_b;
wire [7:0] rf_filter_coeff373_a;
wire [7:0] rf_filter_coeff373_b;
wire [7:0] rf_filter_coeff374_a;
wire [7:0] rf_filter_coeff374_b;
wire [7:0] rf_filter_coeff375_a;
wire [7:0] rf_filter_coeff375_b;
wire [7:0] rf_filter_coeff376_a;
wire [7:0] rf_filter_coeff376_b;
wire [7:0] rf_filter_coeff377_a;
wire [7:0] rf_filter_coeff377_b;
wire [7:0] rf_filter_coeff378_a;
wire [7:0] rf_filter_coeff378_b;
wire [7:0] rf_filter_coeff379_a;
wire [7:0] rf_filter_coeff379_b;
wire [7:0] rf_filter_coeff380_a;
wire [7:0] rf_filter_coeff380_b;
wire [7:0] rf_filter_coeff381_a;
wire [7:0] rf_filter_coeff381_b;
wire [7:0] rf_filter_coeff382_a;
wire [7:0] rf_filter_coeff382_b;
wire [7:0] rf_filter_coeff383_a;
wire [7:0] rf_filter_coeff383_b;
wire [7:0] rf_filter_coeff384_a;
wire [7:0] rf_filter_coeff384_b;
wire [7:0] rf_filter_coeff385_a;
wire [7:0] rf_filter_coeff385_b;
wire [7:0] rf_filter_coeff386_a;
wire [7:0] rf_filter_coeff386_b;
wire [7:0] rf_filter_coeff387_a;
wire [7:0] rf_filter_coeff387_b;
wire [7:0] rf_filter_coeff388_a;
wire [7:0] rf_filter_coeff388_b;
wire [7:0] rf_filter_coeff389_a;
wire [7:0] rf_filter_coeff389_b;
wire [7:0] rf_filter_coeff390_a;
wire [7:0] rf_filter_coeff390_b;
wire [7:0] rf_filter_coeff391_a;
wire [7:0] rf_filter_coeff391_b;
wire [7:0] rf_filter_coeff392_a;
wire [7:0] rf_filter_coeff392_b;
wire [7:0] rf_filter_coeff393_a;
wire [7:0] rf_filter_coeff393_b;
wire [7:0] rf_filter_coeff394_a;
wire [7:0] rf_filter_coeff394_b;
wire [7:0] rf_filter_coeff395_a;
wire [7:0] rf_filter_coeff395_b;
wire [7:0] rf_filter_coeff396_a;
wire [7:0] rf_filter_coeff396_b;
wire [7:0] rf_filter_coeff397_a;
wire [7:0] rf_filter_coeff397_b;
wire [7:0] rf_filter_coeff398_a;
wire [7:0] rf_filter_coeff398_b;
wire [7:0] rf_filter_coeff399_a;
wire [7:0] rf_filter_coeff399_b;
wire [7:0] rf_filter_coeff400_a;
wire [7:0] rf_filter_coeff400_b;
wire [7:0] rf_filter_coeff401_a;
wire [7:0] rf_filter_coeff401_b;
wire [7:0] rf_filter_coeff402_a;
wire [7:0] rf_filter_coeff402_b;
wire [7:0] rf_filter_coeff403_a;
wire [7:0] rf_filter_coeff403_b;
wire [7:0] rf_filter_coeff404_a;
wire [7:0] rf_filter_coeff404_b;
wire [7:0] rf_filter_coeff405_a;
wire [7:0] rf_filter_coeff405_b;
wire [7:0] rf_filter_coeff406_a;
```

```
wire [7:0] rf_filter_coeff406_b;
wire [7:0] rf_filter_coeff407_a;
wire [7:0] rf_filter_coeff407_b;
wire [7:0] rf_filter_coeff408_a;
wire [7:0] rf_filter_coeff408_b;
wire [7:0] rf_filter_coeff409_a;
wire [7:0] rf_filter_coeff409_b;
wire [7:0] rf_filter_coeff410_a;
wire [7:0] rf_filter_coeff410_b;
wire [7:0] rf_filter_coeff411_a;
wire [7:0] rf_filter_coeff411_b;
wire [7:0] rf_filter_coeff412_a;
wire [7:0] rf_filter_coeff412_b;
wire [7:0] rf_filter_coeff413_a;
wire [7:0] rf_filter_coeff413_b;
wire [7:0] rf_filter_coeff414_a;
wire [7:0] rf_filter_coeff414_b;
wire [7:0] rf_filter_coeff415_a;
wire [7:0] rf_filter_coeff415_b;
wire [7:0] rf_filter_coeff416_a;
wire [7:0] rf_filter_coeff416_b;
wire [7:0] rf_filter_coeff417_a;
wire [7:0] rf_filter_coeff417_b;
wire [7:0] rf_filter_coeff418_a;
wire [7:0] rf_filter_coeff418_b;
wire [7:0] rf_filter_coeff419_a;
wire [7:0] rf_filter_coeff419_b;
wire [7:0] rf_filter_coeff420_a;
wire [7:0] rf_filter_coeff420_b;
wire [7:0] rf_filter_coeff421_a;
wire [7:0] rf_filter_coeff421_b;
wire [7:0] rf_filter_coeff422_a;
wire [7:0] rf_filter_coeff422_b;
wire [7:0] rf_filter_coeff423_a;
wire [7:0] rf_filter_coeff423_b;
wire [7:0] rf_filter_coeff424_a;
wire [7:0] rf_filter_coeff424_b;
wire [7:0] rf_filter_coeff425_a;
wire [7:0] rf_filter_coeff425_b;
wire [7:0] rf_filter_coeff426_a;
wire [7:0] rf_filter_coeff426_b;
wire [7:0] rf_filter_coeff427_a;
wire [7:0] rf_filter_coeff427_b;
wire [7:0] rf_filter_coeff428_a;
wire [7:0] rf_filter_coeff428_b;
wire [7:0] rf_filter_coeff429_a;
wire [7:0] rf_filter_coeff429_b;
wire [7:0] rf_filter_coeff430_a;
wire [7:0] rf_filter_coeff430_b;
wire [7:0] rf_filter_coeff431_a;
wire [7:0] rf_filter_coeff431_b;
wire [7:0] rf_filter_coeff432_a;
wire [7:0] rf_filter_coeff432_b;
wire [7:0] rf_filter_coeff433_a;
wire [7:0] rf_filter_coeff433_b;
wire [7:0] rf_filter_coeff434_a;
wire [7:0] rf_filter_coeff434_b;
wire [7:0] rf_filter_coeff435_a;
wire [7:0] rf_filter_coeff435_b;
wire [7:0] rf_filter_coeff436_a;
wire [7:0] rf_filter_coeff436_b;
wire [7:0] rf_filter_coeff437_a;
wire [7:0] rf_filter_coeff437_b;
wire [7:0] rf_filter_coeff438_a;
wire [7:0] rf_filter_coeff438_b;
wire [7:0] rf_filter_coeff439_a;
wire [7:0] rf_filter_coeff439_b;
wire [7:0] rf_filter_coeff440_a;
wire [7:0] rf_filter_coeff440_b;
wire [7:0] rf_filter_coeff441_a;
wire [7:0] rf_filter_coeff441_b;
```

```
wire [7:0] rf_filter_coeff442_a;
wire [7:0] rf_filter_coeff442_b;
wire [7:0] rf_filter_coeff443_a;
wire [7:0] rf_filter_coeff443_b;
wire [7:0] rf_filter_coeff444_a;
wire [7:0] rf_filter_coeff444_b;
wire [7:0] rf_filter_coeff445_a;
wire [7:0] rf_filter_coeff445_b;
wire [7:0] rf_filter_coeff446_a;
wire [7:0] rf_filter_coeff446_b;
wire [7:0] rf_filter_coeff447_a;
wire [7:0] rf_filter_coeff447_b;
wire [7:0] rf_filter_coeff448_a;
wire [7:0] rf_filter_coeff448_b;
wire [7:0] rf_filter_coeff449_a;
wire [7:0] rf_filter_coeff449_b;
wire [7:0] rf_filter_coeff450_a;
wire [7:0] rf_filter_coeff450_b;
wire [7:0] rf_filter_coeff451_a;
wire [7:0] rf_filter_coeff451_b;
wire [7:0] rf_filter_coeff452_a;
wire [7:0] rf_filter_coeff452_b;
wire [7:0] rf_filter_coeff453_a;
wire [7:0] rf_filter_coeff453_b;
wire [7:0] rf_filter_coeff454_a;
wire [7:0] rf_filter_coeff454_b;
wire [7:0] rf_filter_coeff455_a;
wire [7:0] rf_filter_coeff455_b;
wire [7:0] rf_filter_coeff456_a;
wire [7:0] rf_filter_coeff456_b;
wire [7:0] rf_filter_coeff457_a;
wire [7:0] rf_filter_coeff457_b;
wire [7:0] rf_filter_coeff458_a;
wire [7:0] rf_filter_coeff458_b;
wire [7:0] rf_filter_coeff459_a;
wire [7:0] rf_filter_coeff459_b;
wire [7:0] rf_filter_coeff460_a;
wire [7:0] rf_filter_coeff460_b;
wire [7:0] rf_filter_coeff461_a;
wire [7:0] rf_filter_coeff461_b;
wire [7:0] rf_filter_coeff462_a;
wire [7:0] rf_filter_coeff462_b;
wire [7:0] rf_filter_coeff463_a;
wire [7:0] rf_filter_coeff463_b;
wire [7:0] rf_filter_coeff464_a;
wire [7:0] rf_filter_coeff464_b;
wire [7:0] rf_filter_coeff465_a;
wire [7:0] rf_filter_coeff465_b;
wire [7:0] rf_filter_coeff466_a;
wire [7:0] rf_filter_coeff466_b;
wire [7:0] rf_filter_coeff467_a;
wire [7:0] rf_filter_coeff467_b;
wire [7:0] rf_filter_coeff468_a;
wire [7:0] rf_filter_coeff468_b;
wire [7:0] rf_filter_coeff469_a;
wire [7:0] rf_filter_coeff469_b;
wire [7:0] rf_filter_coeff470_a;
wire [7:0] rf_filter_coeff470_b;
wire [7:0] rf_filter_coeff471_a;
wire [7:0] rf_filter_coeff471_b;
wire [7:0] rf_filter_coeff472_a;
wire [7:0] rf_filter_coeff472_b;
wire [7:0] rf_filter_coeff473_a;
wire [7:0] rf_filter_coeff473_b;
wire [7:0] rf_filter_coeff474_a;
wire [7:0] rf_filter_coeff474_b;
wire [7:0] rf_filter_coeff475_a;
wire [7:0] rf_filter_coeff475_b;
wire [7:0] rf_filter_coeff476_a;
wire [7:0] rf_filter_coeff476_b;
wire [7:0] rf_filter_coeff477_a;
```

```
wire [7:0] rf_filter_coeff477_b;
wire [7:0] rf_filter_coeff478_a;
wire [7:0] rf_filter_coeff478_b;
wire [7:0] rf_filter_coeff479_a;
wire [7:0] rf_filter_coeff479_b;
wire [7:0] rf_filter_coeff480_a;
wire [7:0] rf_filter_coeff480_b;
wire [7:0] rf_filter_coeff481_a;
wire [7:0] rf_filter_coeff481_b;
wire [7:0] rf_filter_coeff482_a;
wire [7:0] rf_filter_coeff482_b;
wire [7:0] rf_filter_coeff483_a;
wire [7:0] rf_filter_coeff483_b;
wire [7:0] rf_filter_coeff484_a;
wire [7:0] rf_filter_coeff484_b;
wire [7:0] rf_filter_coeff485_a;
wire [7:0] rf_filter_coeff485_b;
wire [7:0] rf_filter_coeff486_a;
wire [7:0] rf_filter_coeff486_b;
wire [7:0] rf_filter_coeff487_a;
wire [7:0] rf_filter_coeff487_b;
wire [7:0] rf_filter_coeff488_a;
wire [7:0] rf_filter_coeff488_b;
wire [7:0] rf_filter_coeff489_a;
wire [7:0] rf_filter_coeff489_b;
wire [7:0] rf_filter_coeff490_a;
wire [7:0] rf_filter_coeff490_b;
wire [7:0] rf_filter_coeff491_a;
wire [7:0] rf_filter_coeff491_b;
wire [7:0] rf_filter_coeff492_a;
wire [7:0] rf_filter_coeff492_b;
wire [7:0] rf_filter_coeff493_a;
wire [7:0] rf_filter_coeff493_b;
wire [7:0] rf_filter_coeff494_a;
wire [7:0] rf_filter_coeff494_b;
wire [7:0] rf_filter_coeff495_a;
wire [7:0] rf_filter_coeff495_b;
wire [7:0] rf_filter_coeff496_a;
wire [7:0] rf_filter_coeff496_b;
wire [7:0] rf_filter_coeff497_a;
wire [7:0] rf_filter_coeff497_b;
wire [7:0] rf_filter_coeff498_a;
wire [7:0] rf_filter_coeff498_b;
wire [7:0] rf_filter_coeff499_a;
wire [7:0] rf_filter_coeff499_b;
wire [7:0] rf_filter_coeff500_a;
wire [7:0] rf_filter_coeff500_b;
wire [7:0] rf_filter_coeff501_a;
wire [7:0] rf_filter_coeff501_b;
wire [7:0] rf_filter_coeff502_a;
wire [7:0] rf_filter_coeff502_b;
wire [7:0] rf_filter_coeff503_a;
wire [7:0] rf_filter_coeff503_b;
wire [7:0] rf_filter_coeff504_a;
wire [7:0] rf_filter_coeff504_b;
wire [7:0] rf_filter_coeff505_a;
wire [7:0] rf_filter_coeff505_b;
wire [7:0] rf_filter_coeff506_a;
wire [7:0] rf_filter_coeff506_b;
wire [7:0] rf_filter_coeff507_a;
wire [7:0] rf_filter_coeff507_b;
wire [7:0] rf_filter_coeff508_a;
wire [7:0] rf_filter_coeff508_b;
wire [7:0] rf_filter_coeff509_a;
wire [7:0] rf_filter_coeff509_b;
wire [7:0] rf_filter_coeff510_a;
wire [7:0] rf_filter_coeff510_b;
wire [7:0] rf_filter_coeff511_a;
wire [7:0] rf_filter_coeff511_b;

// Instantiate the Unit Under Test (UUT)
```

```
register uut (
          .rst_n(rst_n),
          .clk(clk),
          .addr(addr),
          .wdata(wdata),
          .w_enable(w_enable),
          .wxfc(wxfc),
          .rxfc(rxfc),
          .ro_fifo_underrun(ro_fifo_underrun),
          .ro_fifo_overrun(ro_fifo_overrun),
          .rdata(rdata),
          .rf_i2si_bist_en(rf_i2si_bist_en),
          .rf_filter_shift(rf_filter_shift),
          .rf_filter_clip_en(rf_filter_clip_en),
          .ro_filter_ovf_flag(ro_filter_ovf_flag),
          .trig_fifo_overrun(trig_fifo_overrun),
          .trig_fifo_underrun(trig_fifo_underrun),
          .trig_filter_ovf_flag_clear(trig_filter_ovf_flag_clear),
          .rf_i2si_bist_start_val_a(rf_i2si_bist_start_val_a),
          .rf_i2si_bist_start_val_b(rf_i2si_bist_start_val_b),
          .rf_i2si_bist_incr(rf_i2si_bist_incr),
          .rf_i2si_bist_upper_limit_a(rf_i2si_bist_upper_limit_a),
          .rf_i2si_bist_upper_limit_b(rf_i2si_bist_upper_limit_b),
          .rf_filter_coeff0_a(rf_filter_coeff0_a),
          .rf_filter_coeff0_b(rf_filter_coeff0_b),
          .rf_filter_coeff1_a(rf_filter_coeff1_a),
          .rf_filter_coeff1_b(rf_filter_coeff1_b),
          .rf_filter_coeff2_a(rf_filter_coeff2_a),
          .rf_filter_coeff2_b(rf_filter_coeff2_b),
          .rf_filter_coeff3_a(rf_filter_coeff3_a),
          .rf_filter_coeff3_b(rf_filter_coeff3_b),
          .rf_filter_coeff4_a(rf_filter_coeff4_a),
          .rf_filter_coeff4_b(rf_filter_coeff4_b),
          .rf_filter_coeff5_a(rf_filter_coeff5_a),
          .rf_filter_coeff5_b(rf_filter_coeff5_b),
          .rf_filter_coeff6_a(rf_filter_coeff6_a),
          .rf_filter_coeff6_b(rf_filter_coeff6_b),
          .rf_filter_coeff7_a(rf_filter_coeff7_a),
          .rf_filter_coeff7_b(rf_filter_coeff7_b),
          .rf_filter_coeff8_a(rf_filter_coeff8_a),
          .rf_filter_coeff8_b(rf_filter_coeff8_b),
          .rf_filter_coeff9_a(rf_filter_coeff9_a),
          .rf_filter_coeff9_b(rf_filter_coeff9_b),
          .rf_filter_coeff10_a(rf_filter_coeff10_a),
          .rf_filter_coeff10_b(rf_filter_coeff10_b),
          .rf_filter_coeff11_a(rf_filter_coeff11_a),
          .rf_filter_coeff11_b(rf_filter_coeff11_b),
          .rf_filter_coeff12_a(rf_filter_coeff12_a),
          .rf_filter_coeff12_b(rf_filter_coeff12_b),
          .rf_filter_coeff13_a(rf_filter_coeff13_a),
          .rf_filter_coeff13_b(rf_filter_coeff13_b),
          .rf_filter_coeff14_a(rf_filter_coeff14_a),
          .rf_filter_coeff14_b(rf_filter_coeff14_b),
          .rf_filter_coeff15_a(rf_filter_coeff15_a),
          .rf_filter_coeff15_b(rf_filter_coeff15_b),
          .rf_filter_coeff16_a(rf_filter_coeff16_a),
          .rf_filter_coeff16_b(rf_filter_coeff16_b),
          .rf_filter_coeff17_a(rf_filter_coeff17_a),
          .rf_filter_coeff17_b(rf_filter_coeff17_b),
          .rf_filter_coeff18_a(rf_filter_coeff18_a),
          .rf_filter_coeff18_b(rf_filter_coeff18_b),
          .rf_filter_coeff19_a(rf_filter_coeff19_a),
          .rf_filter_coeff19_b(rf_filter_coeff19_b),
          .rf_filter_coeff20_a(rf_filter_coeff20_a),
          .rf_filter_coeff20_b(rf_filter_coeff20_b),
          .rf_filter_coeff21_a(rf_filter_coeff21_a),
          .rf_filter_coeff21_b(rf_filter_coeff21_b),
          .rf_filter_coeff22_a(rf_filter_coeff22_a),
          .rf_filter_coeff22_b(rf_filter_coeff22_b),
          .rf_filter_coeff23_a(rf_filter_coeff23_a),
          .rf_filter_coeff23_b(rf_filter_coeff23_b),
```

```
.rf_filter_coeff24_a(rf_filter_coeff24_a),
.rf_filter_coeff24_b(rf_filter_coeff24_b),
.rf_filter_coeff25_a(rf_filter_coeff25_a),
.rf_filter_coeff25_b(rf_filter_coeff25_b),
.rf_filter_coeff26_a(rf_filter_coeff26_a),
.rf_filter_coeff26_b(rf_filter_coeff26_b),
.rf_filter_coeff27_a(rf_filter_coeff27_a),
.rf_filter_coeff27_b(rf_filter_coeff27_b),
.rf_filter_coeff28_a(rf_filter_coeff28_a),
.rf_filter_coeff28_b(rf_filter_coeff28_b),
.rf_filter_coeff29_a(rf_filter_coeff29_a),
.rf_filter_coeff29_b(rf_filter_coeff29_b),
.rf_filter_coeff30_a(rf_filter_coeff30_a),
.rf_filter_coeff30_b(rf_filter_coeff30_b),
.rf_filter_coeff31_a(rf_filter_coeff31_a),
.rf_filter_coeff31_b(rf_filter_coeff31_b),
.rf_filter_coeff32_a(rf_filter_coeff32_a),
.rf_filter_coeff32_b(rf_filter_coeff32_b),
.rf_filter_coeff33_a(rf_filter_coeff33_a),
.rf_filter_coeff33_b(rf_filter_coeff33_b),
.rf_filter_coeff34_a(rf_filter_coeff34_a),
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
```

```
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
```

```
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
```

```
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
```

```
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
```

```
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
```

```
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
```

```
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
```

```
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
```

```
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
```

```
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
```

```
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
```

```
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
.rf_filter_coeff479_a(rf_filter_coeff479_a),
.rf_filter_coeff479_b(rf_filter_coeff479_b),
.rf_filter_coeff480_a(rf_filter_coeff480_a),
.rf_filter_coeff480_b(rf_filter_coeff480_b),
.rf_filter_coeff481_a(rf_filter_coeff481_a),
.rf_filter_coeff481_b(rf_filter_coeff481_b),
.rf_filter_coeff482_a(rf_filter_coeff482_a),
.rf_filter_coeff482_b(rf_filter_coeff482_b),
.rf_filter_coeff483_a(rf_filter_coeff483_a),
.rf_filter_coeff483_b(rf_filter_coeff483_b),
.rf_filter_coeff484_a(rf_filter_coeff484_a),
.rf_filter_coeff484_b(rf_filter_coeff484_b),
.rf_filter_coeff485_a(rf_filter_coeff485_a),
```

```
            .rf_filter_coeff485_b(rf_filter_coeff485_b),
            .rf_filter_coeff486_a(rf_filter_coeff486_a),
            .rf_filter_coeff486_b(rf_filter_coeff486_b),
            .rf_filter_coeff487_a(rf_filter_coeff487_a),
            .rf_filter_coeff487_b(rf_filter_coeff487_b),
            .rf_filter_coeff488_a(rf_filter_coeff488_a),
            .rf_filter_coeff488_b(rf_filter_coeff488_b),
            .rf_filter_coeff489_a(rf_filter_coeff489_a),
            .rf_filter_coeff489_b(rf_filter_coeff489_b),
            .rf_filter_coeff490_a(rf_filter_coeff490_a),
            .rf_filter_coeff490_b(rf_filter_coeff490_b),
            .rf_filter_coeff491_a(rf_filter_coeff491_a),
            .rf_filter_coeff491_b(rf_filter_coeff491_b),
            .rf_filter_coeff492_a(rf_filter_coeff492_a),
            .rf_filter_coeff492_b(rf_filter_coeff492_b),
            .rf_filter_coeff493_a(rf_filter_coeff493_a),
            .rf_filter_coeff493_b(rf_filter_coeff493_b),
            .rf_filter_coeff494_a(rf_filter_coeff494_a),
            .rf_filter_coeff494_b(rf_filter_coeff494_b),
            .rf_filter_coeff495_a(rf_filter_coeff495_a),
            .rf_filter_coeff495_b(rf_filter_coeff495_b),
            .rf_filter_coeff496_a(rf_filter_coeff496_a),
            .rf_filter_coeff496_b(rf_filter_coeff496_b),
            .rf_filter_coeff497_a(rf_filter_coeff497_a),
            .rf_filter_coeff497_b(rf_filter_coeff497_b),
            .rf_filter_coeff498_a(rf_filter_coeff498_a),
            .rf_filter_coeff498_b(rf_filter_coeff498_b),
            .rf_filter_coeff499_a(rf_filter_coeff499_a),
            .rf_filter_coeff499_b(rf_filter_coeff499_b),
            .rf_filter_coeff500_a(rf_filter_coeff500_a),
            .rf_filter_coeff500_b(rf_filter_coeff500_b),
            .rf_filter_coeff501_a(rf_filter_coeff501_a),
            .rf_filter_coeff501_b(rf_filter_coeff501_b),
            .rf_filter_coeff502_a(rf_filter_coeff502_a),
            .rf_filter_coeff502_b(rf_filter_coeff502_b),
            .rf_filter_coeff503_a(rf_filter_coeff503_a),
            .rf_filter_coeff503_b(rf_filter_coeff503_b),
            .rf_filter_coeff504_a(rf_filter_coeff504_a),
            .rf_filter_coeff504_b(rf_filter_coeff504_b),
            .rf_filter_coeff505_a(rf_filter_coeff505_a),
            .rf_filter_coeff505_b(rf_filter_coeff505_b),
            .rf_filter_coeff506_a(rf_filter_coeff506_a),
            .rf_filter_coeff506_b(rf_filter_coeff506_b),
            .rf_filter_coeff507_a(rf_filter_coeff507_a),
            .rf_filter_coeff507_b(rf_filter_coeff507_b),
            .rf_filter_coeff508_a(rf_filter_coeff508_a),
            .rf_filter_coeff508_b(rf_filter_coeff508_b),
            .rf_filter_coeff509_a(rf_filter_coeff509_a),
            .rf_filter_coeff509_b(rf_filter_coeff509_b),
            .rf_filter_coeff510_a(rf_filter_coeff510_a),
            .rf_filter_coeff510_b(rf_filter_coeff510_b),
            .rf_filter_coeff511_a(rf_filter_coeff511_a),
            .rf_filter_coeff511_b(rf_filter_coeff511_b)
);

initial begin
            // Initialize Inputs
            rst_n = 0;
            clk = 0;
            addr = 0;
            wdata = 0;
            w_enable = 0;
            wxfc = 0;
            ro_fifo_underrun = 0;
            ro_fifo_overrun = 0;
            ro_filter_ovf_flag = 0;

            // Wait 100 ns for global reset to finish
            #100;
```

```
        end

endmodule
```

register_testbench.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             register_testbench.v
// Create Date:             1/19/2016
// Last Modification:       3/16/2016
// Author:                  Julie Swift
// Description: ???????????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module register_testbench;

          // Inputs
          reg rst_n;
          reg clk;
          reg [10:0] addr;
          reg [7:0] wdata;
          reg w_enable;
          reg wxfc;
          reg rxfc;
          reg ro_i2c_reg_indir_data;
          reg ro_fifo_underrun;
          reg ro_fifo_overrun;
          reg ro_filter_ovf_flag;
          reg ro_filter_ovf_flag_clear;
          reg rf_i2so_clk2sck_div_a;
          reg rf_i2so_clk2sck_div_b;

          // Outputs
          wire [7:0] rdata;
          wire rf_soft_reset;
          wire rf_i2si_bist_en;
          wire rf_filter_shift;
          wire rf_filter_clip_en;
          wire trig_fifo_overrun;
          wire trig_fifo_underrun;
          wire rf_i2si_bist_start_val_a;
          wire rf_i2si_bist_start_val_b;
          wire rf_i2si_bist_incr;
          wire rf_i2si_bist_upper_limit_a;
          wire rf_i2si_bist_upper_limit_b;
          wire rf_filter_coeff0_a;
          wire rf_filter_coeff0_b;
          wire rf_filter_coeff1_a;
          wire rf_filter_coeff1_b;
          wire rf_filter_coeff2_a;
          wire rf_filter_coeff2_b;
          wire rf_filter_coeff3_a;
          wire rf_filter_coeff3_b;
          wire rf_filter_coeff4_a;
          wire rf_filter_coeff4_b;
          wire rf_filter_coeff5_a;
          wire rf_filter_coeff5_b;
          wire rf_filter_coeff6_a;
          wire rf_filter_coeff6_b;
          wire rf_filter_coeff7_a;
          wire rf_filter_coeff7_b;
          wire rf_filter_coeff8_a;
          wire rf_filter_coeff8_b;
          wire rf_filter_coeff9_a;
          wire rf_filter_coeff9_b;
          wire rf_filter_coeff10_a;
          wire rf_filter_coeff10_b;
          wire rf_filter_coeff11_a;
          wire rf_filter_coeff11_b;
          wire rf_filter_coeff12_a;
          wire rf_filter_coeff12_b;
          wire rf_filter_coeff13_a;
          wire rf_filter_coeff13_b;
```

```
wire rf_filter_coeff14_a;
wire rf_filter_coeff14_b;
wire rf_filter_coeff15_a;
wire rf_filter_coeff15_b;
wire rf_filter_coeff16_a;
wire rf_filter_coeff16_b;
wire rf_filter_coeff17_a;
wire rf_filter_coeff17_b;
wire rf_filter_coeff18_a;
wire rf_filter_coeff18_b;
wire rf_filter_coeff19_a;
wire rf_filter_coeff19_b;
wire rf_filter_coeff20_a;
wire rf_filter_coeff20_b;
wire rf_filter_coeff21_a;
wire rf_filter_coeff21_b;
wire rf_filter_coeff22_a;
wire rf_filter_coeff22_b;
wire rf_filter_coeff23_a;
wire rf_filter_coeff23_b;
wire rf_filter_coeff24_a;
wire rf_filter_coeff24_b;
wire rf_filter_coeff25_a;
wire rf_filter_coeff25_b;
wire rf_filter_coeff26_a;
wire rf_filter_coeff26_b;
wire rf_filter_coeff27_a;
wire rf_filter_coeff27_b;
wire rf_filter_coeff28_a;
wire rf_filter_coeff28_b;
wire rf_filter_coeff29_a;
wire rf_filter_coeff29_b;
wire rf_filter_coeff30_a;
wire rf_filter_coeff30_b;
wire rf_filter_coeff31_a;
wire rf_filter_coeff31_b;
wire rf_filter_coeff32_a;
wire rf_filter_coeff32_b;
wire rf_filter_coeff33_a;
wire rf_filter_coeff33_b;
wire rf_filter_coeff34_a;
wire rf_filter_coeff34_b;
wire rf_filter_coeff35_a;
wire rf_filter_coeff35_b;
wire rf_filter_coeff36_a;
wire rf_filter_coeff36_b;
wire rf_filter_coeff37_a;
wire rf_filter_coeff37_b;
wire rf_filter_coeff38_a;
wire rf_filter_coeff38_b;
wire rf_filter_coeff39_a;
wire rf_filter_coeff39_b;
wire rf_filter_coeff40_a;
wire rf_filter_coeff40_b;
wire rf_filter_coeff41_a;
wire rf_filter_coeff41_b;
wire rf_filter_coeff42_a;
wire rf_filter_coeff42_b;
wire rf_filter_coeff43_a;
wire rf_filter_coeff43_b;
wire rf_filter_coeff44_a;
wire rf_filter_coeff44_b;
wire rf_filter_coeff45_a;
wire rf_filter_coeff45_b;
wire rf_filter_coeff46_a;
wire rf_filter_coeff46_b;
wire rf_filter_coeff47_a;
wire rf_filter_coeff47_b;
wire rf_filter_coeff48_a;
wire rf_filter_coeff48_b;
wire rf_filter_coeff49_a;
```

```
wire rf_filter_coeff49_b;
wire rf_filter_coeff50_a;
wire rf_filter_coeff50_b;
wire rf_filter_coeff51_a;
wire rf_filter_coeff51_b;
wire rf_filter_coeff52_a;
wire rf_filter_coeff52_b;
wire rf_filter_coeff53_a;
wire rf_filter_coeff53_b;
wire rf_filter_coeff54_a;
wire rf_filter_coeff54_b;
wire rf_filter_coeff55_a;
wire rf_filter_coeff55_b;
wire rf_filter_coeff56_a;
wire rf_filter_coeff56_b;
wire rf_filter_coeff57_a;
wire rf_filter_coeff57_b;
wire rf_filter_coeff58_a;
wire rf_filter_coeff58_b;
wire rf_filter_coeff59_a;
wire rf_filter_coeff59_b;
wire rf_filter_coeff60_a;
wire rf_filter_coeff60_b;
wire rf_filter_coeff61_a;
wire rf_filter_coeff61_b;
wire rf_filter_coeff62_a;
wire rf_filter_coeff62_b;
wire rf_filter_coeff63_a;
wire rf_filter_coeff63_b;
wire rf_filter_coeff64_a;
wire rf_filter_coeff64_b;
wire rf_filter_coeff65_a;
wire rf_filter_coeff65_b;
wire rf_filter_coeff66_a;
wire rf_filter_coeff66_b;
wire rf_filter_coeff67_a;
wire rf_filter_coeff67_b;
wire rf_filter_coeff68_a;
wire rf_filter_coeff68_b;
wire rf_filter_coeff69_a;
wire rf_filter_coeff69_b;
wire rf_filter_coeff70_a;
wire rf_filter_coeff70_b;
wire rf_filter_coeff71_a;
wire rf_filter_coeff71_b;
wire rf_filter_coeff72_a;
wire rf_filter_coeff72_b;
wire rf_filter_coeff73_a;
wire rf_filter_coeff73_b;
wire rf_filter_coeff74_a;
wire rf_filter_coeff74_b;
wire rf_filter_coeff75_a;
wire rf_filter_coeff75_b;
wire rf_filter_coeff76_a;
wire rf_filter_coeff76_b;
wire rf_filter_coeff77_a;
wire rf_filter_coeff77_b;
wire rf_filter_coeff78_a;
wire rf_filter_coeff78_b;
wire rf_filter_coeff79_a;
wire rf_filter_coeff79_b;
wire rf_filter_coeff80_a;
wire rf_filter_coeff80_b;
wire rf_filter_coeff81_a;
wire rf_filter_coeff81_b;
wire rf_filter_coeff82_a;
wire rf_filter_coeff82_b;
wire rf_filter_coeff83_a;
wire rf_filter_coeff83_b;
wire rf_filter_coeff84_a;
wire rf_filter_coeff84_b;
```

```
wire rf_filter_coeff85_a;
wire rf_filter_coeff85_b;
wire rf_filter_coeff86_a;
wire rf_filter_coeff86_b;
wire rf_filter_coeff87_a;
wire rf_filter_coeff87_b;
wire rf_filter_coeff88_a;
wire rf_filter_coeff88_b;
wire rf_filter_coeff89_a;
wire rf_filter_coeff89_b;
wire rf_filter_coeff90_a;
wire rf_filter_coeff90_b;
wire rf_filter_coeff91_a;
wire rf_filter_coeff91_b;
wire rf_filter_coeff92_a;
wire rf_filter_coeff92_b;
wire rf_filter_coeff93_a;
wire rf_filter_coeff93_b;
wire rf_filter_coeff94_a;
wire rf_filter_coeff94_b;
wire rf_filter_coeff95_a;
wire rf_filter_coeff95_b;
wire rf_filter_coeff96_a;
wire rf_filter_coeff96_b;
wire rf_filter_coeff97_a;
wire rf_filter_coeff97_b;
wire rf_filter_coeff98_a;
wire rf_filter_coeff98_b;
wire rf_filter_coeff99_a;
wire rf_filter_coeff99_b;
wire rf_filter_coeff100_a;
wire rf_filter_coeff100_b;
wire rf_filter_coeff101_a;
wire rf_filter_coeff101_b;
wire rf_filter_coeff102_a;
wire rf_filter_coeff102_b;
wire rf_filter_coeff103_a;
wire rf_filter_coeff103_b;
wire rf_filter_coeff104_a;
wire rf_filter_coeff104_b;
wire rf_filter_coeff105_a;
wire rf_filter_coeff105_b;
wire rf_filter_coeff106_a;
wire rf_filter_coeff106_b;
wire rf_filter_coeff107_a;
wire rf_filter_coeff107_b;
wire rf_filter_coeff108_a;
wire rf_filter_coeff108_b;
wire rf_filter_coeff109_a;
wire rf_filter_coeff109_b;
wire rf_filter_coeff110_a;
wire rf_filter_coeff110_b;
wire rf_filter_coeff111_a;
wire rf_filter_coeff111_b;
wire rf_filter_coeff112_a;
wire rf_filter_coeff112_b;
wire rf_filter_coeff113_a;
wire rf_filter_coeff113_b;
wire rf_filter_coeff114_a;
wire rf_filter_coeff114_b;
wire rf_filter_coeff115_a;
wire rf_filter_coeff115_b;
wire rf_filter_coeff116_a;
wire rf_filter_coeff116_b;
wire rf_filter_coeff117_a;
wire rf_filter_coeff117_b;
wire rf_filter_coeff118_a;
wire rf_filter_coeff118_b;
wire rf_filter_coeff119_a;
wire rf_filter_coeff119_b;
wire rf_filter_coeff120_a;
```

```
wire rf_filter_coeff120_b;
wire rf_filter_coeff121_a;
wire rf_filter_coeff121_b;
wire rf_filter_coeff122_a;
wire rf_filter_coeff122_b;
wire rf_filter_coeff123_a;
wire rf_filter_coeff123_b;
wire rf_filter_coeff124_a;
wire rf_filter_coeff124_b;
wire rf_filter_coeff125_a;
wire rf_filter_coeff125_b;
wire rf_filter_coeff126_a;
wire rf_filter_coeff126_b;
wire rf_filter_coeff127_a;
wire rf_filter_coeff127_b;
wire rf_filter_coeff128_a;
wire rf_filter_coeff128_b;
wire rf_filter_coeff129_a;
wire rf_filter_coeff129_b;
wire rf_filter_coeff130_a;
wire rf_filter_coeff130_b;
wire rf_filter_coeff131_a;
wire rf_filter_coeff131_b;
wire rf_filter_coeff132_a;
wire rf_filter_coeff132_b;
wire rf_filter_coeff133_a;
wire rf_filter_coeff133_b;
wire rf_filter_coeff134_a;
wire rf_filter_coeff134_b;
wire rf_filter_coeff135_a;
wire rf_filter_coeff135_b;
wire rf_filter_coeff136_a;
wire rf_filter_coeff136_b;
wire rf_filter_coeff137_a;
wire rf_filter_coeff137_b;
wire rf_filter_coeff138_a;
wire rf_filter_coeff138_b;
wire rf_filter_coeff139_a;
wire rf_filter_coeff139_b;
wire rf_filter_coeff140_a;
wire rf_filter_coeff140_b;
wire rf_filter_coeff141_a;
wire rf_filter_coeff141_b;
wire rf_filter_coeff142_a;
wire rf_filter_coeff142_b;
wire rf_filter_coeff143_a;
wire rf_filter_coeff143_b;
wire rf_filter_coeff144_a;
wire rf_filter_coeff144_b;
wire rf_filter_coeff145_a;
wire rf_filter_coeff145_b;
wire rf_filter_coeff146_a;
wire rf_filter_coeff146_b;
wire rf_filter_coeff147_a;
wire rf_filter_coeff147_b;
wire rf_filter_coeff148_a;
wire rf_filter_coeff148_b;
wire rf_filter_coeff149_a;
wire rf_filter_coeff149_b;
wire rf_filter_coeff150_a;
wire rf_filter_coeff150_b;
wire rf_filter_coeff151_a;
wire rf_filter_coeff151_b;
wire rf_filter_coeff152_a;
wire rf_filter_coeff152_b;
wire rf_filter_coeff153_a;
wire rf_filter_coeff153_b;
wire rf_filter_coeff154_a;
wire rf_filter_coeff154_b;
wire rf_filter_coeff155_a;
wire rf_filter_coeff155_b;
```

```
wire rf_filter_coeff156_a;
wire rf_filter_coeff156_b;
wire rf_filter_coeff157_a;
wire rf_filter_coeff157_b;
wire rf_filter_coeff158_a;
wire rf_filter_coeff158_b;
wire rf_filter_coeff159_a;
wire rf_filter_coeff159_b;
wire rf_filter_coeff160_a;
wire rf_filter_coeff160_b;
wire rf_filter_coeff161_a;
wire rf_filter_coeff161_b;
wire rf_filter_coeff162_a;
wire rf_filter_coeff162_b;
wire rf_filter_coeff163_a;
wire rf_filter_coeff163_b;
wire rf_filter_coeff164_a;
wire rf_filter_coeff164_b;
wire rf_filter_coeff165_a;
wire rf_filter_coeff165_b;
wire rf_filter_coeff166_a;
wire rf_filter_coeff166_b;
wire rf_filter_coeff167_a;
wire rf_filter_coeff167_b;
wire rf_filter_coeff168_a;
wire rf_filter_coeff168_b;
wire rf_filter_coeff169_a;
wire rf_filter_coeff169_b;
wire rf_filter_coeff170_a;
wire rf_filter_coeff170_b;
wire rf_filter_coeff171_a;
wire rf_filter_coeff171_b;
wire rf_filter_coeff172_a;
wire rf_filter_coeff172_b;
wire rf_filter_coeff173_a;
wire rf_filter_coeff173_b;
wire rf_filter_coeff174_a;
wire rf_filter_coeff174_b;
wire rf_filter_coeff175_a;
wire rf_filter_coeff175_b;
wire rf_filter_coeff176_a;
wire rf_filter_coeff176_b;
wire rf_filter_coeff177_a;
wire rf_filter_coeff177_b;
wire rf_filter_coeff178_a;
wire rf_filter_coeff178_b;
wire rf_filter_coeff179_a;
wire rf_filter_coeff179_b;
wire rf_filter_coeff180_a;
wire rf_filter_coeff180_b;
wire rf_filter_coeff181_a;
wire rf_filter_coeff181_b;
wire rf_filter_coeff182_a;
wire rf_filter_coeff182_b;
wire rf_filter_coeff183_a;
wire rf_filter_coeff183_b;
wire rf_filter_coeff184_a;
wire rf_filter_coeff184_b;
wire rf_filter_coeff185_a;
wire rf_filter_coeff185_b;
wire rf_filter_coeff186_a;
wire rf_filter_coeff186_b;
wire rf_filter_coeff187_a;
wire rf_filter_coeff187_b;
wire rf_filter_coeff188_a;
wire rf_filter_coeff188_b;
wire rf_filter_coeff189_a;
wire rf_filter_coeff189_b;
wire rf_filter_coeff190_a;
wire rf_filter_coeff190_b;
wire rf_filter_coeff191_a;
```

```
wire rf_filter_coeff191_b;
wire rf_filter_coeff192_a;
wire rf_filter_coeff192_b;
wire rf_filter_coeff193_a;
wire rf_filter_coeff193_b;
wire rf_filter_coeff194_a;
wire rf_filter_coeff194_b;
wire rf_filter_coeff195_a;
wire rf_filter_coeff195_b;
wire rf_filter_coeff196_a;
wire rf_filter_coeff196_b;
wire rf_filter_coeff197_a;
wire rf_filter_coeff197_b;
wire rf_filter_coeff198_a;
wire rf_filter_coeff198_b;
wire rf_filter_coeff199_a;
wire rf_filter_coeff199_b;
wire rf_filter_coeff200_a;
wire rf_filter_coeff200_b;
wire rf_filter_coeff201_a;
wire rf_filter_coeff201_b;
wire rf_filter_coeff202_a;
wire rf_filter_coeff202_b;
wire rf_filter_coeff203_a;
wire rf_filter_coeff203_b;
wire rf_filter_coeff204_a;
wire rf_filter_coeff204_b;
wire rf_filter_coeff205_a;
wire rf_filter_coeff205_b;
wire rf_filter_coeff206_a;
wire rf_filter_coeff206_b;
wire rf_filter_coeff207_a;
wire rf_filter_coeff207_b;
wire rf_filter_coeff208_a;
wire rf_filter_coeff208_b;
wire rf_filter_coeff209_a;
wire rf_filter_coeff209_b;
wire rf_filter_coeff210_a;
wire rf_filter_coeff210_b;
wire rf_filter_coeff211_a;
wire rf_filter_coeff211_b;
wire rf_filter_coeff212_a;
wire rf_filter_coeff212_b;
wire rf_filter_coeff213_a;
wire rf_filter_coeff213_b;
wire rf_filter_coeff214_a;
wire rf_filter_coeff214_b;
wire rf_filter_coeff215_a;
wire rf_filter_coeff215_b;
wire rf_filter_coeff216_a;
wire rf_filter_coeff216_b;
wire rf_filter_coeff217_a;
wire rf_filter_coeff217_b;
wire rf_filter_coeff218_a;
wire rf_filter_coeff218_b;
wire rf_filter_coeff219_a;
wire rf_filter_coeff219_b;
wire rf_filter_coeff220_a;
wire rf_filter_coeff220_b;
wire rf_filter_coeff221_a;
wire rf_filter_coeff221_b;
wire rf_filter_coeff222_a;
wire rf_filter_coeff222_b;
wire rf_filter_coeff223_a;
wire rf_filter_coeff223_b;
wire rf_filter_coeff224_a;
wire rf_filter_coeff224_b;
wire rf_filter_coeff225_a;
wire rf_filter_coeff225_b;
wire rf_filter_coeff226_a;
wire rf_filter_coeff226_b;
```

```
wire rf_filter_coeff227_a;
wire rf_filter_coeff227_b;
wire rf_filter_coeff228_a;
wire rf_filter_coeff228_b;
wire rf_filter_coeff229_a;
wire rf_filter_coeff229_b;
wire rf_filter_coeff230_a;
wire rf_filter_coeff230_b;
wire rf_filter_coeff231_a;
wire rf_filter_coeff231_b;
wire rf_filter_coeff232_a;
wire rf_filter_coeff232_b;
wire rf_filter_coeff233_a;
wire rf_filter_coeff233_b;
wire rf_filter_coeff234_a;
wire rf_filter_coeff234_b;
wire rf_filter_coeff235_a;
wire rf_filter_coeff235_b;
wire rf_filter_coeff236_a;
wire rf_filter_coeff236_b;
wire rf_filter_coeff237_a;
wire rf_filter_coeff237_b;
wire rf_filter_coeff238_a;
wire rf_filter_coeff238_b;
wire rf_filter_coeff239_a;
wire rf_filter_coeff239_b;
wire rf_filter_coeff240_a;
wire rf_filter_coeff240_b;
wire rf_filter_coeff241_a;
wire rf_filter_coeff241_b;
wire rf_filter_coeff242_a;
wire rf_filter_coeff242_b;
wire rf_filter_coeff243_a;
wire rf_filter_coeff243_b;
wire rf_filter_coeff244_a;
wire rf_filter_coeff244_b;
wire rf_filter_coeff245_a;
wire rf_filter_coeff245_b;
wire rf_filter_coeff246_a;
wire rf_filter_coeff246_b;
wire rf_filter_coeff247_a;
wire rf_filter_coeff247_b;
wire rf_filter_coeff248_a;
wire rf_filter_coeff248_b;
wire rf_filter_coeff249_a;
wire rf_filter_coeff249_b;
wire rf_filter_coeff250_a;
wire rf_filter_coeff250_b;
wire rf_filter_coeff251_a;
wire rf_filter_coeff251_b;
wire rf_filter_coeff252_a;
wire rf_filter_coeff252_b;
wire rf_filter_coeff253_a;
wire rf_filter_coeff253_b;
wire rf_filter_coeff254_a;
wire rf_filter_coeff254_b;
wire rf_filter_coeff255_a;
wire rf_filter_coeff255_b;
wire rf_filter_coeff256_a;
wire rf_filter_coeff256_b;
wire rf_filter_coeff257_a;
wire rf_filter_coeff257_b;
wire rf_filter_coeff258_a;
wire rf_filter_coeff258_b;
wire rf_filter_coeff259_a;
wire rf_filter_coeff259_b;
wire rf_filter_coeff260_a;
wire rf_filter_coeff260_b;
wire rf_filter_coeff261_a;
wire rf_filter_coeff261_b;
wire rf_filter_coeff262_a;
```

```
wire rf_filter_coeff262_b;
wire rf_filter_coeff263_a;
wire rf_filter_coeff263_b;
wire rf_filter_coeff264_a;
wire rf_filter_coeff264_b;
wire rf_filter_coeff265_a;
wire rf_filter_coeff265_b;
wire rf_filter_coeff266_a;
wire rf_filter_coeff266_b;
wire rf_filter_coeff267_a;
wire rf_filter_coeff267_b;
wire rf_filter_coeff268_a;
wire rf_filter_coeff268_b;
wire rf_filter_coeff269_a;
wire rf_filter_coeff269_b;
wire rf_filter_coeff270_a;
wire rf_filter_coeff270_b;
wire rf_filter_coeff271_a;
wire rf_filter_coeff271_b;
wire rf_filter_coeff272_a;
wire rf_filter_coeff272_b;
wire rf_filter_coeff273_a;
wire rf_filter_coeff273_b;
wire rf_filter_coeff274_a;
wire rf_filter_coeff274_b;
wire rf_filter_coeff275_a;
wire rf_filter_coeff275_b;
wire rf_filter_coeff276_a;
wire rf_filter_coeff276_b;
wire rf_filter_coeff277_a;
wire rf_filter_coeff277_b;
wire rf_filter_coeff278_a;
wire rf_filter_coeff278_b;
wire rf_filter_coeff279_a;
wire rf_filter_coeff279_b;
wire rf_filter_coeff280_a;
wire rf_filter_coeff280_b;
wire rf_filter_coeff281_a;
wire rf_filter_coeff281_b;
wire rf_filter_coeff282_a;
wire rf_filter_coeff282_b;
wire rf_filter_coeff283_a;
wire rf_filter_coeff283_b;
wire rf_filter_coeff284_a;
wire rf_filter_coeff284_b;
wire rf_filter_coeff285_a;
wire rf_filter_coeff285_b;
wire rf_filter_coeff286_a;
wire rf_filter_coeff286_b;
wire rf_filter_coeff287_a;
wire rf_filter_coeff287_b;
wire rf_filter_coeff288_a;
wire rf_filter_coeff288_b;
wire rf_filter_coeff289_a;
wire rf_filter_coeff289_b;
wire rf_filter_coeff290_a;
wire rf_filter_coeff290_b;
wire rf_filter_coeff291_a;
wire rf_filter_coeff291_b;
wire rf_filter_coeff292_a;
wire rf_filter_coeff292_b;
wire rf_filter_coeff293_a;
wire rf_filter_coeff293_b;
wire rf_filter_coeff294_a;
wire rf_filter_coeff294_b;
wire rf_filter_coeff295_a;
wire rf_filter_coeff295_b;
wire rf_filter_coeff296_a;
wire rf_filter_coeff296_b;
wire rf_filter_coeff297_a;
wire rf_filter_coeff297_b;
```

```
wire rf_filter_coeff298_a;
wire rf_filter_coeff298_b;
wire rf_filter_coeff299_a;
wire rf_filter_coeff299_b;
wire rf_filter_coeff300_a;
wire rf_filter_coeff300_b;
wire rf_filter_coeff301_a;
wire rf_filter_coeff301_b;
wire rf_filter_coeff302_a;
wire rf_filter_coeff302_b;
wire rf_filter_coeff303_a;
wire rf_filter_coeff303_b;
wire rf_filter_coeff304_a;
wire rf_filter_coeff304_b;
wire rf_filter_coeff305_a;
wire rf_filter_coeff305_b;
wire rf_filter_coeff306_a;
wire rf_filter_coeff306_b;
wire rf_filter_coeff307_a;
wire rf_filter_coeff307_b;
wire rf_filter_coeff308_a;
wire rf_filter_coeff308_b;
wire rf_filter_coeff309_a;
wire rf_filter_coeff309_b;
wire rf_filter_coeff310_a;
wire rf_filter_coeff310_b;
wire rf_filter_coeff311_a;
wire rf_filter_coeff311_b;
wire rf_filter_coeff312_a;
wire rf_filter_coeff312_b;
wire rf_filter_coeff313_a;
wire rf_filter_coeff313_b;
wire rf_filter_coeff314_a;
wire rf_filter_coeff314_b;
wire rf_filter_coeff315_a;
wire rf_filter_coeff315_b;
wire rf_filter_coeff316_a;
wire rf_filter_coeff316_b;
wire rf_filter_coeff317_a;
wire rf_filter_coeff317_b;
wire rf_filter_coeff318_a;
wire rf_filter_coeff318_b;
wire rf_filter_coeff319_a;
wire rf_filter_coeff319_b;
wire rf_filter_coeff320_a;
wire rf_filter_coeff320_b;
wire rf_filter_coeff321_a;
wire rf_filter_coeff321_b;
wire rf_filter_coeff322_a;
wire rf_filter_coeff322_b;
wire rf_filter_coeff323_a;
wire rf_filter_coeff323_b;
wire rf_filter_coeff324_a;
wire rf_filter_coeff324_b;
wire rf_filter_coeff325_a;
wire rf_filter_coeff325_b;
wire rf_filter_coeff326_a;
wire rf_filter_coeff326_b;
wire rf_filter_coeff327_a;
wire rf_filter_coeff327_b;
wire rf_filter_coeff328_a;
wire rf_filter_coeff328_b;
wire rf_filter_coeff329_a;
wire rf_filter_coeff329_b;
wire rf_filter_coeff330_a;
wire rf_filter_coeff330_b;
wire rf_filter_coeff331_a;
wire rf_filter_coeff331_b;
wire rf_filter_coeff332_a;
wire rf_filter_coeff332_b;
wire rf_filter_coeff333_a;
```

```
wire rf_filter_coeff333_b;
wire rf_filter_coeff334_a;
wire rf_filter_coeff334_b;
wire rf_filter_coeff335_a;
wire rf_filter_coeff335_b;
wire rf_filter_coeff336_a;
wire rf_filter_coeff336_b;
wire rf_filter_coeff337_a;
wire rf_filter_coeff337_b;
wire rf_filter_coeff338_a;
wire rf_filter_coeff338_b;
wire rf_filter_coeff339_a;
wire rf_filter_coeff339_b;
wire rf_filter_coeff340_a;
wire rf_filter_coeff340_b;
wire rf_filter_coeff341_a;
wire rf_filter_coeff341_b;
wire rf_filter_coeff342_a;
wire rf_filter_coeff342_b;
wire rf_filter_coeff343_a;
wire rf_filter_coeff343_b;
wire rf_filter_coeff344_a;
wire rf_filter_coeff344_b;
wire rf_filter_coeff345_a;
wire rf_filter_coeff345_b;
wire rf_filter_coeff346_a;
wire rf_filter_coeff346_b;
wire rf_filter_coeff347_a;
wire rf_filter_coeff347_b;
wire rf_filter_coeff348_a;
wire rf_filter_coeff348_b;
wire rf_filter_coeff349_a;
wire rf_filter_coeff349_b;
wire rf_filter_coeff350_a;
wire rf_filter_coeff350_b;
wire rf_filter_coeff351_a;
wire rf_filter_coeff351_b;
wire rf_filter_coeff352_a;
wire rf_filter_coeff352_b;
wire rf_filter_coeff353_a;
wire rf_filter_coeff353_b;
wire rf_filter_coeff354_a;
wire rf_filter_coeff354_b;
wire rf_filter_coeff355_a;
wire rf_filter_coeff355_b;
wire rf_filter_coeff356_a;
wire rf_filter_coeff356_b;
wire rf_filter_coeff357_a;
wire rf_filter_coeff357_b;
wire rf_filter_coeff358_a;
wire rf_filter_coeff358_b;
wire rf_filter_coeff359_a;
wire rf_filter_coeff359_b;
wire rf_filter_coeff360_a;
wire rf_filter_coeff360_b;
wire rf_filter_coeff361_a;
wire rf_filter_coeff361_b;
wire rf_filter_coeff362_a;
wire rf_filter_coeff362_b;
wire rf_filter_coeff363_a;
wire rf_filter_coeff363_b;
wire rf_filter_coeff364_a;
wire rf_filter_coeff364_b;
wire rf_filter_coeff365_a;
wire rf_filter_coeff365_b;
wire rf_filter_coeff366_a;
wire rf_filter_coeff366_b;
wire rf_filter_coeff367_a;
wire rf_filter_coeff367_b;
wire rf_filter_coeff368_a;
wire rf_filter_coeff368_b;
```

```
wire rf_filter_coeff369_a;
wire rf_filter_coeff369_b;
wire rf_filter_coeff370_a;
wire rf_filter_coeff370_b;
wire rf_filter_coeff371_a;
wire rf_filter_coeff371_b;
wire rf_filter_coeff372_a;
wire rf_filter_coeff372_b;
wire rf_filter_coeff373_a;
wire rf_filter_coeff373_b;
wire rf_filter_coeff374_a;
wire rf_filter_coeff374_b;
wire rf_filter_coeff375_a;
wire rf_filter_coeff375_b;
wire rf_filter_coeff376_a;
wire rf_filter_coeff376_b;
wire rf_filter_coeff377_a;
wire rf_filter_coeff377_b;
wire rf_filter_coeff378_a;
wire rf_filter_coeff378_b;
wire rf_filter_coeff379_a;
wire rf_filter_coeff379_b;
wire rf_filter_coeff380_a;
wire rf_filter_coeff380_b;
wire rf_filter_coeff381_a;
wire rf_filter_coeff381_b;
wire rf_filter_coeff382_a;
wire rf_filter_coeff382_b;
wire rf_filter_coeff383_a;
wire rf_filter_coeff383_b;
wire rf_filter_coeff384_a;
wire rf_filter_coeff384_b;
wire rf_filter_coeff385_a;
wire rf_filter_coeff385_b;
wire rf_filter_coeff386_a;
wire rf_filter_coeff386_b;
wire rf_filter_coeff387_a;
wire rf_filter_coeff387_b;
wire rf_filter_coeff388_a;
wire rf_filter_coeff388_b;
wire rf_filter_coeff389_a;
wire rf_filter_coeff389_b;
wire rf_filter_coeff390_a;
wire rf_filter_coeff390_b;
wire rf_filter_coeff391_a;
wire rf_filter_coeff391_b;
wire rf_filter_coeff392_a;
wire rf_filter_coeff392_b;
wire rf_filter_coeff393_a;
wire rf_filter_coeff393_b;
wire rf_filter_coeff394_a;
wire rf_filter_coeff394_b;
wire rf_filter_coeff395_a;
wire rf_filter_coeff395_b;
wire rf_filter_coeff396_a;
wire rf_filter_coeff396_b;
wire rf_filter_coeff397_a;
wire rf_filter_coeff397_b;
wire rf_filter_coeff398_a;
wire rf_filter_coeff398_b;
wire rf_filter_coeff399_a;
wire rf_filter_coeff399_b;
wire rf_filter_coeff400_a;
wire rf_filter_coeff400_b;
wire rf_filter_coeff401_a;
wire rf_filter_coeff401_b;
wire rf_filter_coeff402_a;
wire rf_filter_coeff402_b;
wire rf_filter_coeff403_a;
wire rf_filter_coeff403_b;
wire rf_filter_coeff404_a;
```

```
wire rf_filter_coeff404_b;
wire rf_filter_coeff405_a;
wire rf_filter_coeff405_b;
wire rf_filter_coeff406_a;
wire rf_filter_coeff406_b;
wire rf_filter_coeff407_a;
wire rf_filter_coeff407_b;
wire rf_filter_coeff408_a;
wire rf_filter_coeff408_b;
wire rf_filter_coeff409_a;
wire rf_filter_coeff409_b;
wire rf_filter_coeff410_a;
wire rf_filter_coeff410_b;
wire rf_filter_coeff411_a;
wire rf_filter_coeff411_b;
wire rf_filter_coeff412_a;
wire rf_filter_coeff412_b;
wire rf_filter_coeff413_a;
wire rf_filter_coeff413_b;
wire rf_filter_coeff414_a;
wire rf_filter_coeff414_b;
wire rf_filter_coeff415_a;
wire rf_filter_coeff415_b;
wire rf_filter_coeff416_a;
wire rf_filter_coeff416_b;
wire rf_filter_coeff417_a;
wire rf_filter_coeff417_b;
wire rf_filter_coeff418_a;
wire rf_filter_coeff418_b;
wire rf_filter_coeff419_a;
wire rf_filter_coeff419_b;
wire rf_filter_coeff420_a;
wire rf_filter_coeff420_b;
wire rf_filter_coeff421_a;
wire rf_filter_coeff421_b;
wire rf_filter_coeff422_a;
wire rf_filter_coeff422_b;
wire rf_filter_coeff423_a;
wire rf_filter_coeff423_b;
wire rf_filter_coeff424_a;
wire rf_filter_coeff424_b;
wire rf_filter_coeff425_a;
wire rf_filter_coeff425_b;
wire rf_filter_coeff426_a;
wire rf_filter_coeff426_b;
wire rf_filter_coeff427_a;
wire rf_filter_coeff427_b;
wire rf_filter_coeff428_a;
wire rf_filter_coeff428_b;
wire rf_filter_coeff429_a;
wire rf_filter_coeff429_b;
wire rf_filter_coeff430_a;
wire rf_filter_coeff430_b;
wire rf_filter_coeff431_a;
wire rf_filter_coeff431_b;
wire rf_filter_coeff432_a;
wire rf_filter_coeff432_b;
wire rf_filter_coeff433_a;
wire rf_filter_coeff433_b;
wire rf_filter_coeff434_a;
wire rf_filter_coeff434_b;
wire rf_filter_coeff435_a;
wire rf_filter_coeff435_b;
wire rf_filter_coeff436_a;
wire rf_filter_coeff436_b;
wire rf_filter_coeff437_a;
wire rf_filter_coeff437_b;
wire rf_filter_coeff438_a;
wire rf_filter_coeff438_b;
wire rf_filter_coeff439_a;
wire rf_filter_coeff439_b;
```

```
wire rf_filter_coeff440_a;
wire rf_filter_coeff440_b;
wire rf_filter_coeff441_a;
wire rf_filter_coeff441_b;
wire rf_filter_coeff442_a;
wire rf_filter_coeff442_b;
wire rf_filter_coeff443_a;
wire rf_filter_coeff443_b;
wire rf_filter_coeff444_a;
wire rf_filter_coeff444_b;
wire rf_filter_coeff445_a;
wire rf_filter_coeff445_b;
wire rf_filter_coeff446_a;
wire rf_filter_coeff446_b;
wire rf_filter_coeff447_a;
wire rf_filter_coeff447_b;
wire rf_filter_coeff448_a;
wire rf_filter_coeff448_b;
wire rf_filter_coeff449_a;
wire rf_filter_coeff449_b;
wire rf_filter_coeff450_a;
wire rf_filter_coeff450_b;
wire rf_filter_coeff451_a;
wire rf_filter_coeff451_b;
wire rf_filter_coeff452_a;
wire rf_filter_coeff452_b;
wire rf_filter_coeff453_a;
wire rf_filter_coeff453_b;
wire rf_filter_coeff454_a;
wire rf_filter_coeff454_b;
wire rf_filter_coeff455_a;
wire rf_filter_coeff455_b;
wire rf_filter_coeff456_a;
wire rf_filter_coeff456_b;
wire rf_filter_coeff457_a;
wire rf_filter_coeff457_b;
wire rf_filter_coeff458_a;
wire rf_filter_coeff458_b;
wire rf_filter_coeff459_a;
wire rf_filter_coeff459_b;
wire rf_filter_coeff460_a;
wire rf_filter_coeff460_b;
wire rf_filter_coeff461_a;
wire rf_filter_coeff461_b;
wire rf_filter_coeff462_a;
wire rf_filter_coeff462_b;
wire rf_filter_coeff463_a;
wire rf_filter_coeff463_b;
wire rf_filter_coeff464_a;
wire rf_filter_coeff464_b;
wire rf_filter_coeff465_a;
wire rf_filter_coeff465_b;
wire rf_filter_coeff466_a;
wire rf_filter_coeff466_b;
wire rf_filter_coeff467_a;
wire rf_filter_coeff467_b;
wire rf_filter_coeff468_a;
wire rf_filter_coeff468_b;
wire rf_filter_coeff469_a;
wire rf_filter_coeff469_b;
wire rf_filter_coeff470_a;
wire rf_filter_coeff470_b;
wire rf_filter_coeff471_a;
wire rf_filter_coeff471_b;
wire rf_filter_coeff472_a;
wire rf_filter_coeff472_b;
wire rf_filter_coeff473_a;
wire rf_filter_coeff473_b;
wire rf_filter_coeff474_a;
wire rf_filter_coeff474_b;
wire rf_filter_coeff475_a;
```

```
wire rf_filter_coeff475_b;
wire rf_filter_coeff476_a;
wire rf_filter_coeff476_b;
wire rf_filter_coeff477_a;
wire rf_filter_coeff477_b;
wire rf_filter_coeff478_a;
wire rf_filter_coeff478_b;
wire rf_filter_coeff479_a;
wire rf_filter_coeff479_b;
wire rf_filter_coeff480_a;
wire rf_filter_coeff480_b;
wire rf_filter_coeff481_a;
wire rf_filter_coeff481_b;
wire rf_filter_coeff482_a;
wire rf_filter_coeff482_b;
wire rf_filter_coeff483_a;
wire rf_filter_coeff483_b;
wire rf_filter_coeff484_a;
wire rf_filter_coeff484_b;
wire rf_filter_coeff485_a;
wire rf_filter_coeff485_b;
wire rf_filter_coeff486_a;
wire rf_filter_coeff486_b;
wire rf_filter_coeff487_a;
wire rf_filter_coeff487_b;
wire rf_filter_coeff488_a;
wire rf_filter_coeff488_b;
wire rf_filter_coeff489_a;
wire rf_filter_coeff489_b;
wire rf_filter_coeff490_a;
wire rf_filter_coeff490_b;
wire rf_filter_coeff491_a;
wire rf_filter_coeff491_b;
wire rf_filter_coeff492_a;
wire rf_filter_coeff492_b;
wire rf_filter_coeff493_a;
wire rf_filter_coeff493_b;
wire rf_filter_coeff494_a;
wire rf_filter_coeff494_b;
wire rf_filter_coeff495_a;
wire rf_filter_coeff495_b;
wire rf_filter_coeff496_a;
wire rf_filter_coeff496_b;
wire rf_filter_coeff497_a;
wire rf_filter_coeff497_b;
wire rf_filter_coeff498_a;
wire rf_filter_coeff498_b;
wire rf_filter_coeff499_a;
wire rf_filter_coeff499_b;
wire rf_filter_coeff500_a;
wire rf_filter_coeff500_b;
wire rf_filter_coeff501_a;
wire rf_filter_coeff501_b;
wire rf_filter_coeff502_a;
wire rf_filter_coeff502_b;
wire rf_filter_coeff503_a;
wire rf_filter_coeff503_b;
wire rf_filter_coeff504_a;
wire rf_filter_coeff504_b;
wire rf_filter_coeff505_a;
wire rf_filter_coeff505_b;
wire rf_filter_coeff506_a;
wire rf_filter_coeff506_b;
wire rf_filter_coeff507_a;
wire rf_filter_coeff507_b;
wire rf_filter_coeff508_a;
wire rf_filter_coeff508_b;
wire rf_filter_coeff509_a;
wire rf_filter_coeff509_b;
wire rf_filter_coeff510_a;
wire rf_filter_coeff510_b;
```

```
wire rf_filter_coeff511_a;
wire rf_filter_coeff511_b;

// Instantiate the Unit Under Test (UUT)
register uut (
          .rst_n(rst_n),
          .clk(clk),
          .addr(addr),
          .wdata(wdata),
          .w_enable(w_enable),
          .wxfc(wxfc),
          .rxfc(rxfc),
          .ro_i2c_reg_indir_data(ro_i2c_reg_indir_data),
          .ro_fifo_underrun(ro_fifo_underrun),
          .ro_fifo_overrun(ro_fifo_overrun),
          .rdata(rdata),
          .rf_soft_reset(rf_soft_reset),
          .rf_i2si_bist_en(rf_i2si_bist_en),
          .rf_filter_shift(rf_filter_shift),
          .rf_filter_clip_en(rf_filter_clip_en),
          .rf_i2si_dec_factor(rf_i2si_dec_factor),
          .rf_i2so_dec_factor(rf_i2so_dec_factor),
          .rf_i2so_clk2sck_div_a(rf_i2so_clk2sck_div_a),
          .rf_i2so_clk2sck_div_b(rf_i2so_clk2sck_div_b),
          .trig_fifo_overrun(trig_fifo_overrun),
          .trig_fifo_underrun(trig_fifo_underrun),
          .rf_i2si_bist_start_val_a(rf_i2si_bist_start_val_a),
          .rf_i2si_bist_start_val_b(rf_i2si_bist_start_val_b),
          .rf_i2si_bist_incr(rf_i2si_bist_incr),
          .rf_i2si_bist_upper_limit_a(rf_i2si_bist_upper_limit_a),
          .rf_i2si_bist_upper_limit_b(rf_i2si_bist_upper_limit_b),
          .rf_i2c_reg_indir_addr_a(rf_i2c_reg_indir_addr_a),
          .rf_i2c_reg_indir_addr_b(rf_i2c_reg_indir_addr_b),
          .rf_filter_coeff0_a(rf_filter_coeff0_a),
          .rf_filter_coeff0_b(rf_filter_coeff0_b),
          .rf_filter_coeff1_a(rf_filter_coeff1_a),
          .rf_filter_coeff1_b(rf_filter_coeff1_b),
          .rf_filter_coeff2_a(rf_filter_coeff2_a),
          .rf_filter_coeff2_b(rf_filter_coeff2_b),
          .rf_filter_coeff3_a(rf_filter_coeff3_a),
          .rf_filter_coeff3_b(rf_filter_coeff3_b),
          .rf_filter_coeff4_a(rf_filter_coeff4_a),
          .rf_filter_coeff4_b(rf_filter_coeff4_b),
          .rf_filter_coeff5_a(rf_filter_coeff5_a),
          .rf_filter_coeff5_b(rf_filter_coeff5_b),
          .rf_filter_coeff6_a(rf_filter_coeff6_a),
          .rf_filter_coeff6_b(rf_filter_coeff6_b),
          .rf_filter_coeff7_a(rf_filter_coeff7_a),
          .rf_filter_coeff7_b(rf_filter_coeff7_b),
          .rf_filter_coeff8_a(rf_filter_coeff8_a),
          .rf_filter_coeff8_b(rf_filter_coeff8_b),
          .rf_filter_coeff9_a(rf_filter_coeff9_a),
          .rf_filter_coeff9_b(rf_filter_coeff9_b),
          .rf_filter_coeff10_a(rf_filter_coeff10_a),
          .rf_filter_coeff10_b(rf_filter_coeff10_b),
          .rf_filter_coeff11_a(rf_filter_coeff11_a),
          .rf_filter_coeff11_b(rf_filter_coeff11_b),
          .rf_filter_coeff12_a(rf_filter_coeff12_a),
          .rf_filter_coeff12_b(rf_filter_coeff12_b),
          .rf_filter_coeff13_a(rf_filter_coeff13_a),
          .rf_filter_coeff13_b(rf_filter_coeff13_b),
          .rf_filter_coeff14_a(rf_filter_coeff14_a),
          .rf_filter_coeff14_b(rf_filter_coeff14_b),
          .rf_filter_coeff15_a(rf_filter_coeff15_a),
          .rf_filter_coeff15_b(rf_filter_coeff15_b),
          .rf_filter_coeff16_a(rf_filter_coeff16_a),
          .rf_filter_coeff16_b(rf_filter_coeff16_b),
          .rf_filter_coeff17_a(rf_filter_coeff17_a),
          .rf_filter_coeff17_b(rf_filter_coeff17_b),
          .rf_filter_coeff18_a(rf_filter_coeff18_a),
          .rf_filter_coeff18_b(rf_filter_coeff18_b),
```

```
.rf_filter_coeff19_a(rf_filter_coeff19_a),
.rf_filter_coeff19_b(rf_filter_coeff19_b),
.rf_filter_coeff20_a(rf_filter_coeff20_a),
.rf_filter_coeff20_b(rf_filter_coeff20_b),
.rf_filter_coeff21_a(rf_filter_coeff21_a),
.rf_filter_coeff21_b(rf_filter_coeff21_b),
.rf_filter_coeff22_a(rf_filter_coeff22_a),
.rf_filter_coeff22_b(rf_filter_coeff22_b),
.rf_filter_coeff23_a(rf_filter_coeff23_a),
.rf_filter_coeff23_b(rf_filter_coeff23_b),
.rf_filter_coeff24_a(rf_filter_coeff24_a),
.rf_filter_coeff24_b(rf_filter_coeff24_b),
.rf_filter_coeff25_a(rf_filter_coeff25_a),
.rf_filter_coeff25_b(rf_filter_coeff25_b),
.rf_filter_coeff26_a(rf_filter_coeff26_a),
.rf_filter_coeff26_b(rf_filter_coeff26_b),
.rf_filter_coeff27_a(rf_filter_coeff27_a),
.rf_filter_coeff27_b(rf_filter_coeff27_b),
.rf_filter_coeff28_a(rf_filter_coeff28_a),
.rf_filter_coeff28_b(rf_filter_coeff28_b),
.rf_filter_coeff29_a(rf_filter_coeff29_a),
.rf_filter_coeff29_b(rf_filter_coeff29_b),
.rf_filter_coeff30_a(rf_filter_coeff30_a),
.rf_filter_coeff30_b(rf_filter_coeff30_b),
.rf_filter_coeff31_a(rf_filter_coeff31_a),
.rf_filter_coeff31_b(rf_filter_coeff31_b),
.rf_filter_coeff32_a(rf_filter_coeff32_a),
.rf_filter_coeff32_b(rf_filter_coeff32_b),
.rf_filter_coeff33_a(rf_filter_coeff33_a),
.rf_filter_coeff33_b(rf_filter_coeff33_b),
.rf_filter_coeff34_a(rf_filter_coeff34_a),
.rf_filter_coeff34_b(rf_filter_coeff34_b),
.rf_filter_coeff35_a(rf_filter_coeff35_a),
.rf_filter_coeff35_b(rf_filter_coeff35_b),
.rf_filter_coeff36_a(rf_filter_coeff36_a),
.rf_filter_coeff36_b(rf_filter_coeff36_b),
.rf_filter_coeff37_a(rf_filter_coeff37_a),
.rf_filter_coeff37_b(rf_filter_coeff37_b),
.rf_filter_coeff38_a(rf_filter_coeff38_a),
.rf_filter_coeff38_b(rf_filter_coeff38_b),
.rf_filter_coeff39_a(rf_filter_coeff39_a),
.rf_filter_coeff39_b(rf_filter_coeff39_b),
.rf_filter_coeff40_a(rf_filter_coeff40_a),
.rf_filter_coeff40_b(rf_filter_coeff40_b),
.rf_filter_coeff41_a(rf_filter_coeff41_a),
.rf_filter_coeff41_b(rf_filter_coeff41_b),
.rf_filter_coeff42_a(rf_filter_coeff42_a),
.rf_filter_coeff42_b(rf_filter_coeff42_b),
.rf_filter_coeff43_a(rf_filter_coeff43_a),
.rf_filter_coeff43_b(rf_filter_coeff43_b),
.rf_filter_coeff44_a(rf_filter_coeff44_a),
.rf_filter_coeff44_b(rf_filter_coeff44_b),
.rf_filter_coeff45_a(rf_filter_coeff45_a),
.rf_filter_coeff45_b(rf_filter_coeff45_b),
.rf_filter_coeff46_a(rf_filter_coeff46_a),
.rf_filter_coeff46_b(rf_filter_coeff46_b),
.rf_filter_coeff47_a(rf_filter_coeff47_a),
.rf_filter_coeff47_b(rf_filter_coeff47_b),
.rf_filter_coeff48_a(rf_filter_coeff48_a),
.rf_filter_coeff48_b(rf_filter_coeff48_b),
.rf_filter_coeff49_a(rf_filter_coeff49_a),
.rf_filter_coeff49_b(rf_filter_coeff49_b),
.rf_filter_coeff50_a(rf_filter_coeff50_a),
.rf_filter_coeff50_b(rf_filter_coeff50_b),
.rf_filter_coeff51_a(rf_filter_coeff51_a),
.rf_filter_coeff51_b(rf_filter_coeff51_b),
.rf_filter_coeff52_a(rf_filter_coeff52_a),
.rf_filter_coeff52_b(rf_filter_coeff52_b),
.rf_filter_coeff53_a(rf_filter_coeff53_a),
.rf_filter_coeff53_b(rf_filter_coeff53_b),
.rf_filter_coeff54_a(rf_filter_coeff54_a),
```

```
.rf_filter_coeff54_b(rf_filter_coeff54_b),
.rf_filter_coeff55_a(rf_filter_coeff55_a),
.rf_filter_coeff55_b(rf_filter_coeff55_b),
.rf_filter_coeff56_a(rf_filter_coeff56_a),
.rf_filter_coeff56_b(rf_filter_coeff56_b),
.rf_filter_coeff57_a(rf_filter_coeff57_a),
.rf_filter_coeff57_b(rf_filter_coeff57_b),
.rf_filter_coeff58_a(rf_filter_coeff58_a),
.rf_filter_coeff58_b(rf_filter_coeff58_b),
.rf_filter_coeff59_a(rf_filter_coeff59_a),
.rf_filter_coeff59_b(rf_filter_coeff59_b),
.rf_filter_coeff60_a(rf_filter_coeff60_a),
.rf_filter_coeff60_b(rf_filter_coeff60_b),
.rf_filter_coeff61_a(rf_filter_coeff61_a),
.rf_filter_coeff61_b(rf_filter_coeff61_b),
.rf_filter_coeff62_a(rf_filter_coeff62_a),
.rf_filter_coeff62_b(rf_filter_coeff62_b),
.rf_filter_coeff63_a(rf_filter_coeff63_a),
.rf_filter_coeff63_b(rf_filter_coeff63_b),
.rf_filter_coeff64_a(rf_filter_coeff64_a),
.rf_filter_coeff64_b(rf_filter_coeff64_b),
.rf_filter_coeff65_a(rf_filter_coeff65_a),
.rf_filter_coeff65_b(rf_filter_coeff65_b),
.rf_filter_coeff66_a(rf_filter_coeff66_a),
.rf_filter_coeff66_b(rf_filter_coeff66_b),
.rf_filter_coeff67_a(rf_filter_coeff67_a),
.rf_filter_coeff67_b(rf_filter_coeff67_b),
.rf_filter_coeff68_a(rf_filter_coeff68_a),
.rf_filter_coeff68_b(rf_filter_coeff68_b),
.rf_filter_coeff69_a(rf_filter_coeff69_a),
.rf_filter_coeff69_b(rf_filter_coeff69_b),
.rf_filter_coeff70_a(rf_filter_coeff70_a),
.rf_filter_coeff70_b(rf_filter_coeff70_b),
.rf_filter_coeff71_a(rf_filter_coeff71_a),
.rf_filter_coeff71_b(rf_filter_coeff71_b),
.rf_filter_coeff72_a(rf_filter_coeff72_a),
.rf_filter_coeff72_b(rf_filter_coeff72_b),
.rf_filter_coeff73_a(rf_filter_coeff73_a),
.rf_filter_coeff73_b(rf_filter_coeff73_b),
.rf_filter_coeff74_a(rf_filter_coeff74_a),
.rf_filter_coeff74_b(rf_filter_coeff74_b),
.rf_filter_coeff75_a(rf_filter_coeff75_a),
.rf_filter_coeff75_b(rf_filter_coeff75_b),
.rf_filter_coeff76_a(rf_filter_coeff76_a),
.rf_filter_coeff76_b(rf_filter_coeff76_b),
.rf_filter_coeff77_a(rf_filter_coeff77_a),
.rf_filter_coeff77_b(rf_filter_coeff77_b),
.rf_filter_coeff78_a(rf_filter_coeff78_a),
.rf_filter_coeff78_b(rf_filter_coeff78_b),
.rf_filter_coeff79_a(rf_filter_coeff79_a),
.rf_filter_coeff79_b(rf_filter_coeff79_b),
.rf_filter_coeff80_a(rf_filter_coeff80_a),
.rf_filter_coeff80_b(rf_filter_coeff80_b),
.rf_filter_coeff81_a(rf_filter_coeff81_a),
.rf_filter_coeff81_b(rf_filter_coeff81_b),
.rf_filter_coeff82_a(rf_filter_coeff82_a),
.rf_filter_coeff82_b(rf_filter_coeff82_b),
.rf_filter_coeff83_a(rf_filter_coeff83_a),
.rf_filter_coeff83_b(rf_filter_coeff83_b),
.rf_filter_coeff84_a(rf_filter_coeff84_a),
.rf_filter_coeff84_b(rf_filter_coeff84_b),
.rf_filter_coeff85_a(rf_filter_coeff85_a),
.rf_filter_coeff85_b(rf_filter_coeff85_b),
.rf_filter_coeff86_a(rf_filter_coeff86_a),
.rf_filter_coeff86_b(rf_filter_coeff86_b),
.rf_filter_coeff87_a(rf_filter_coeff87_a),
.rf_filter_coeff87_b(rf_filter_coeff87_b),
.rf_filter_coeff88_a(rf_filter_coeff88_a),
.rf_filter_coeff88_b(rf_filter_coeff88_b),
.rf_filter_coeff89_a(rf_filter_coeff89_a),
.rf_filter_coeff89_b(rf_filter_coeff89_b),
```

```
.rf_filter_coeff90_a(rf_filter_coeff90_a),
.rf_filter_coeff90_b(rf_filter_coeff90_b),
.rf_filter_coeff91_a(rf_filter_coeff91_a),
.rf_filter_coeff91_b(rf_filter_coeff91_b),
.rf_filter_coeff92_a(rf_filter_coeff92_a),
.rf_filter_coeff92_b(rf_filter_coeff92_b),
.rf_filter_coeff93_a(rf_filter_coeff93_a),
.rf_filter_coeff93_b(rf_filter_coeff93_b),
.rf_filter_coeff94_a(rf_filter_coeff94_a),
.rf_filter_coeff94_b(rf_filter_coeff94_b),
.rf_filter_coeff95_a(rf_filter_coeff95_a),
.rf_filter_coeff95_b(rf_filter_coeff95_b),
.rf_filter_coeff96_a(rf_filter_coeff96_a),
.rf_filter_coeff96_b(rf_filter_coeff96_b),
.rf_filter_coeff97_a(rf_filter_coeff97_a),
.rf_filter_coeff97_b(rf_filter_coeff97_b),
.rf_filter_coeff98_a(rf_filter_coeff98_a),
.rf_filter_coeff98_b(rf_filter_coeff98_b),
.rf_filter_coeff99_a(rf_filter_coeff99_a),
.rf_filter_coeff99_b(rf_filter_coeff99_b),
.rf_filter_coeff100_a(rf_filter_coeff100_a),
.rf_filter_coeff100_b(rf_filter_coeff100_b),
.rf_filter_coeff101_a(rf_filter_coeff101_a),
.rf_filter_coeff101_b(rf_filter_coeff101_b),
.rf_filter_coeff102_a(rf_filter_coeff102_a),
.rf_filter_coeff102_b(rf_filter_coeff102_b),
.rf_filter_coeff103_a(rf_filter_coeff103_a),
.rf_filter_coeff103_b(rf_filter_coeff103_b),
.rf_filter_coeff104_a(rf_filter_coeff104_a),
.rf_filter_coeff104_b(rf_filter_coeff104_b),
.rf_filter_coeff105_a(rf_filter_coeff105_a),
.rf_filter_coeff105_b(rf_filter_coeff105_b),
.rf_filter_coeff106_a(rf_filter_coeff106_a),
.rf_filter_coeff106_b(rf_filter_coeff106_b),
.rf_filter_coeff107_a(rf_filter_coeff107_a),
.rf_filter_coeff107_b(rf_filter_coeff107_b),
.rf_filter_coeff108_a(rf_filter_coeff108_a),
.rf_filter_coeff108_b(rf_filter_coeff108_b),
.rf_filter_coeff109_a(rf_filter_coeff109_a),
.rf_filter_coeff109_b(rf_filter_coeff109_b),
.rf_filter_coeff110_a(rf_filter_coeff110_a),
.rf_filter_coeff110_b(rf_filter_coeff110_b),
.rf_filter_coeff111_a(rf_filter_coeff111_a),
.rf_filter_coeff111_b(rf_filter_coeff111_b),
.rf_filter_coeff112_a(rf_filter_coeff112_a),
.rf_filter_coeff112_b(rf_filter_coeff112_b),
.rf_filter_coeff113_a(rf_filter_coeff113_a),
.rf_filter_coeff113_b(rf_filter_coeff113_b),
.rf_filter_coeff114_a(rf_filter_coeff114_a),
.rf_filter_coeff114_b(rf_filter_coeff114_b),
.rf_filter_coeff115_a(rf_filter_coeff115_a),
.rf_filter_coeff115_b(rf_filter_coeff115_b),
.rf_filter_coeff116_a(rf_filter_coeff116_a),
.rf_filter_coeff116_b(rf_filter_coeff116_b),
.rf_filter_coeff117_a(rf_filter_coeff117_a),
.rf_filter_coeff117_b(rf_filter_coeff117_b),
.rf_filter_coeff118_a(rf_filter_coeff118_a),
.rf_filter_coeff118_b(rf_filter_coeff118_b),
.rf_filter_coeff119_a(rf_filter_coeff119_a),
.rf_filter_coeff119_b(rf_filter_coeff119_b),
.rf_filter_coeff120_a(rf_filter_coeff120_a),
.rf_filter_coeff120_b(rf_filter_coeff120_b),
.rf_filter_coeff121_a(rf_filter_coeff121_a),
.rf_filter_coeff121_b(rf_filter_coeff121_b),
.rf_filter_coeff122_a(rf_filter_coeff122_a),
.rf_filter_coeff122_b(rf_filter_coeff122_b),
.rf_filter_coeff123_a(rf_filter_coeff123_a),
.rf_filter_coeff123_b(rf_filter_coeff123_b),
.rf_filter_coeff124_a(rf_filter_coeff124_a),
.rf_filter_coeff124_b(rf_filter_coeff124_b),
.rf_filter_coeff125_a(rf_filter_coeff125_a),
```

```
.rf_filter_coeff125_b(rf_filter_coeff125_b),
.rf_filter_coeff126_a(rf_filter_coeff126_a),
.rf_filter_coeff126_b(rf_filter_coeff126_b),
.rf_filter_coeff127_a(rf_filter_coeff127_a),
.rf_filter_coeff127_b(rf_filter_coeff127_b),
.rf_filter_coeff128_a(rf_filter_coeff128_a),
.rf_filter_coeff128_b(rf_filter_coeff128_b),
.rf_filter_coeff129_a(rf_filter_coeff129_a),
.rf_filter_coeff129_b(rf_filter_coeff129_b),
.rf_filter_coeff130_a(rf_filter_coeff130_a),
.rf_filter_coeff130_b(rf_filter_coeff130_b),
.rf_filter_coeff131_a(rf_filter_coeff131_a),
.rf_filter_coeff131_b(rf_filter_coeff131_b),
.rf_filter_coeff132_a(rf_filter_coeff132_a),
.rf_filter_coeff132_b(rf_filter_coeff132_b),
.rf_filter_coeff133_a(rf_filter_coeff133_a),
.rf_filter_coeff133_b(rf_filter_coeff133_b),
.rf_filter_coeff134_a(rf_filter_coeff134_a),
.rf_filter_coeff134_b(rf_filter_coeff134_b),
.rf_filter_coeff135_a(rf_filter_coeff135_a),
.rf_filter_coeff135_b(rf_filter_coeff135_b),
.rf_filter_coeff136_a(rf_filter_coeff136_a),
.rf_filter_coeff136_b(rf_filter_coeff136_b),
.rf_filter_coeff137_a(rf_filter_coeff137_a),
.rf_filter_coeff137_b(rf_filter_coeff137_b),
.rf_filter_coeff138_a(rf_filter_coeff138_a),
.rf_filter_coeff138_b(rf_filter_coeff138_b),
.rf_filter_coeff139_a(rf_filter_coeff139_a),
.rf_filter_coeff139_b(rf_filter_coeff139_b),
.rf_filter_coeff140_a(rf_filter_coeff140_a),
.rf_filter_coeff140_b(rf_filter_coeff140_b),
.rf_filter_coeff141_a(rf_filter_coeff141_a),
.rf_filter_coeff141_b(rf_filter_coeff141_b),
.rf_filter_coeff142_a(rf_filter_coeff142_a),
.rf_filter_coeff142_b(rf_filter_coeff142_b),
.rf_filter_coeff143_a(rf_filter_coeff143_a),
.rf_filter_coeff143_b(rf_filter_coeff143_b),
.rf_filter_coeff144_a(rf_filter_coeff144_a),
.rf_filter_coeff144_b(rf_filter_coeff144_b),
.rf_filter_coeff145_a(rf_filter_coeff145_a),
.rf_filter_coeff145_b(rf_filter_coeff145_b),
.rf_filter_coeff146_a(rf_filter_coeff146_a),
.rf_filter_coeff146_b(rf_filter_coeff146_b),
.rf_filter_coeff147_a(rf_filter_coeff147_a),
.rf_filter_coeff147_b(rf_filter_coeff147_b),
.rf_filter_coeff148_a(rf_filter_coeff148_a),
.rf_filter_coeff148_b(rf_filter_coeff148_b),
.rf_filter_coeff149_a(rf_filter_coeff149_a),
.rf_filter_coeff149_b(rf_filter_coeff149_b),
.rf_filter_coeff150_a(rf_filter_coeff150_a),
.rf_filter_coeff150_b(rf_filter_coeff150_b),
.rf_filter_coeff151_a(rf_filter_coeff151_a),
.rf_filter_coeff151_b(rf_filter_coeff151_b),
.rf_filter_coeff152_a(rf_filter_coeff152_a),
.rf_filter_coeff152_b(rf_filter_coeff152_b),
.rf_filter_coeff153_a(rf_filter_coeff153_a),
.rf_filter_coeff153_b(rf_filter_coeff153_b),
.rf_filter_coeff154_a(rf_filter_coeff154_a),
.rf_filter_coeff154_b(rf_filter_coeff154_b),
.rf_filter_coeff155_a(rf_filter_coeff155_a),
.rf_filter_coeff155_b(rf_filter_coeff155_b),
.rf_filter_coeff156_a(rf_filter_coeff156_a),
.rf_filter_coeff156_b(rf_filter_coeff156_b),
.rf_filter_coeff157_a(rf_filter_coeff157_a),
.rf_filter_coeff157_b(rf_filter_coeff157_b),
.rf_filter_coeff158_a(rf_filter_coeff158_a),
.rf_filter_coeff158_b(rf_filter_coeff158_b),
.rf_filter_coeff159_a(rf_filter_coeff159_a),
.rf_filter_coeff159_b(rf_filter_coeff159_b),
.rf_filter_coeff160_a(rf_filter_coeff160_a),
.rf_filter_coeff160_b(rf_filter_coeff160_b),
```

```
.rf_filter_coeff161_a(rf_filter_coeff161_a),
.rf_filter_coeff161_b(rf_filter_coeff161_b),
.rf_filter_coeff162_a(rf_filter_coeff162_a),
.rf_filter_coeff162_b(rf_filter_coeff162_b),
.rf_filter_coeff163_a(rf_filter_coeff163_a),
.rf_filter_coeff163_b(rf_filter_coeff163_b),
.rf_filter_coeff164_a(rf_filter_coeff164_a),
.rf_filter_coeff164_b(rf_filter_coeff164_b),
.rf_filter_coeff165_a(rf_filter_coeff165_a),
.rf_filter_coeff165_b(rf_filter_coeff165_b),
.rf_filter_coeff166_a(rf_filter_coeff166_a),
.rf_filter_coeff166_b(rf_filter_coeff166_b),
.rf_filter_coeff167_a(rf_filter_coeff167_a),
.rf_filter_coeff167_b(rf_filter_coeff167_b),
.rf_filter_coeff168_a(rf_filter_coeff168_a),
.rf_filter_coeff168_b(rf_filter_coeff168_b),
.rf_filter_coeff169_a(rf_filter_coeff169_a),
.rf_filter_coeff169_b(rf_filter_coeff169_b),
.rf_filter_coeff170_a(rf_filter_coeff170_a),
.rf_filter_coeff170_b(rf_filter_coeff170_b),
.rf_filter_coeff171_a(rf_filter_coeff171_a),
.rf_filter_coeff171_b(rf_filter_coeff171_b),
.rf_filter_coeff172_a(rf_filter_coeff172_a),
.rf_filter_coeff172_b(rf_filter_coeff172_b),
.rf_filter_coeff173_a(rf_filter_coeff173_a),
.rf_filter_coeff173_b(rf_filter_coeff173_b),
.rf_filter_coeff174_a(rf_filter_coeff174_a),
.rf_filter_coeff174_b(rf_filter_coeff174_b),
.rf_filter_coeff175_a(rf_filter_coeff175_a),
.rf_filter_coeff175_b(rf_filter_coeff175_b),
.rf_filter_coeff176_a(rf_filter_coeff176_a),
.rf_filter_coeff176_b(rf_filter_coeff176_b),
.rf_filter_coeff177_a(rf_filter_coeff177_a),
.rf_filter_coeff177_b(rf_filter_coeff177_b),
.rf_filter_coeff178_a(rf_filter_coeff178_a),
.rf_filter_coeff178_b(rf_filter_coeff178_b),
.rf_filter_coeff179_a(rf_filter_coeff179_a),
.rf_filter_coeff179_b(rf_filter_coeff179_b),
.rf_filter_coeff180_a(rf_filter_coeff180_a),
.rf_filter_coeff180_b(rf_filter_coeff180_b),
.rf_filter_coeff181_a(rf_filter_coeff181_a),
.rf_filter_coeff181_b(rf_filter_coeff181_b),
.rf_filter_coeff182_a(rf_filter_coeff182_a),
.rf_filter_coeff182_b(rf_filter_coeff182_b),
.rf_filter_coeff183_a(rf_filter_coeff183_a),
.rf_filter_coeff183_b(rf_filter_coeff183_b),
.rf_filter_coeff184_a(rf_filter_coeff184_a),
.rf_filter_coeff184_b(rf_filter_coeff184_b),
.rf_filter_coeff185_a(rf_filter_coeff185_a),
.rf_filter_coeff185_b(rf_filter_coeff185_b),
.rf_filter_coeff186_a(rf_filter_coeff186_a),
.rf_filter_coeff186_b(rf_filter_coeff186_b),
.rf_filter_coeff187_a(rf_filter_coeff187_a),
.rf_filter_coeff187_b(rf_filter_coeff187_b),
.rf_filter_coeff188_a(rf_filter_coeff188_a),
.rf_filter_coeff188_b(rf_filter_coeff188_b),
.rf_filter_coeff189_a(rf_filter_coeff189_a),
.rf_filter_coeff189_b(rf_filter_coeff189_b),
.rf_filter_coeff190_a(rf_filter_coeff190_a),
.rf_filter_coeff190_b(rf_filter_coeff190_b),
.rf_filter_coeff191_a(rf_filter_coeff191_a),
.rf_filter_coeff191_b(rf_filter_coeff191_b),
.rf_filter_coeff192_a(rf_filter_coeff192_a),
.rf_filter_coeff192_b(rf_filter_coeff192_b),
.rf_filter_coeff193_a(rf_filter_coeff193_a),
.rf_filter_coeff193_b(rf_filter_coeff193_b),
.rf_filter_coeff194_a(rf_filter_coeff194_a),
.rf_filter_coeff194_b(rf_filter_coeff194_b),
.rf_filter_coeff195_a(rf_filter_coeff195_a),
.rf_filter_coeff195_b(rf_filter_coeff195_b),
.rf_filter_coeff196_a(rf_filter_coeff196_a),
```

```
.rf_filter_coeff196_b(rf_filter_coeff196_b),
.rf_filter_coeff197_a(rf_filter_coeff197_a),
.rf_filter_coeff197_b(rf_filter_coeff197_b),
.rf_filter_coeff198_a(rf_filter_coeff198_a),
.rf_filter_coeff198_b(rf_filter_coeff198_b),
.rf_filter_coeff199_a(rf_filter_coeff199_a),
.rf_filter_coeff199_b(rf_filter_coeff199_b),
.rf_filter_coeff200_a(rf_filter_coeff200_a),
.rf_filter_coeff200_b(rf_filter_coeff200_b),
.rf_filter_coeff201_a(rf_filter_coeff201_a),
.rf_filter_coeff201_b(rf_filter_coeff201_b),
.rf_filter_coeff202_a(rf_filter_coeff202_a),
.rf_filter_coeff202_b(rf_filter_coeff202_b),
.rf_filter_coeff203_a(rf_filter_coeff203_a),
.rf_filter_coeff203_b(rf_filter_coeff203_b),
.rf_filter_coeff204_a(rf_filter_coeff204_a),
.rf_filter_coeff204_b(rf_filter_coeff204_b),
.rf_filter_coeff205_a(rf_filter_coeff205_a),
.rf_filter_coeff205_b(rf_filter_coeff205_b),
.rf_filter_coeff206_a(rf_filter_coeff206_a),
.rf_filter_coeff206_b(rf_filter_coeff206_b),
.rf_filter_coeff207_a(rf_filter_coeff207_a),
.rf_filter_coeff207_b(rf_filter_coeff207_b),
.rf_filter_coeff208_a(rf_filter_coeff208_a),
.rf_filter_coeff208_b(rf_filter_coeff208_b),
.rf_filter_coeff209_a(rf_filter_coeff209_a),
.rf_filter_coeff209_b(rf_filter_coeff209_b),
.rf_filter_coeff210_a(rf_filter_coeff210_a),
.rf_filter_coeff210_b(rf_filter_coeff210_b),
.rf_filter_coeff211_a(rf_filter_coeff211_a),
.rf_filter_coeff211_b(rf_filter_coeff211_b),
.rf_filter_coeff212_a(rf_filter_coeff212_a),
.rf_filter_coeff212_b(rf_filter_coeff212_b),
.rf_filter_coeff213_a(rf_filter_coeff213_a),
.rf_filter_coeff213_b(rf_filter_coeff213_b),
.rf_filter_coeff214_a(rf_filter_coeff214_a),
.rf_filter_coeff214_b(rf_filter_coeff214_b),
.rf_filter_coeff215_a(rf_filter_coeff215_a),
.rf_filter_coeff215_b(rf_filter_coeff215_b),
.rf_filter_coeff216_a(rf_filter_coeff216_a),
.rf_filter_coeff216_b(rf_filter_coeff216_b),
.rf_filter_coeff217_a(rf_filter_coeff217_a),
.rf_filter_coeff217_b(rf_filter_coeff217_b),
.rf_filter_coeff218_a(rf_filter_coeff218_a),
.rf_filter_coeff218_b(rf_filter_coeff218_b),
.rf_filter_coeff219_a(rf_filter_coeff219_a),
.rf_filter_coeff219_b(rf_filter_coeff219_b),
.rf_filter_coeff220_a(rf_filter_coeff220_a),
.rf_filter_coeff220_b(rf_filter_coeff220_b),
.rf_filter_coeff221_a(rf_filter_coeff221_a),
.rf_filter_coeff221_b(rf_filter_coeff221_b),
.rf_filter_coeff222_a(rf_filter_coeff222_a),
.rf_filter_coeff222_b(rf_filter_coeff222_b),
.rf_filter_coeff223_a(rf_filter_coeff223_a),
.rf_filter_coeff223_b(rf_filter_coeff223_b),
.rf_filter_coeff224_a(rf_filter_coeff224_a),
.rf_filter_coeff224_b(rf_filter_coeff224_b),
.rf_filter_coeff225_a(rf_filter_coeff225_a),
.rf_filter_coeff225_b(rf_filter_coeff225_b),
.rf_filter_coeff226_a(rf_filter_coeff226_a),
.rf_filter_coeff226_b(rf_filter_coeff226_b),
.rf_filter_coeff227_a(rf_filter_coeff227_a),
.rf_filter_coeff227_b(rf_filter_coeff227_b),
.rf_filter_coeff228_a(rf_filter_coeff228_a),
.rf_filter_coeff228_b(rf_filter_coeff228_b),
.rf_filter_coeff229_a(rf_filter_coeff229_a),
.rf_filter_coeff229_b(rf_filter_coeff229_b),
.rf_filter_coeff230_a(rf_filter_coeff230_a),
.rf_filter_coeff230_b(rf_filter_coeff230_b),
.rf_filter_coeff231_a(rf_filter_coeff231_a),
.rf_filter_coeff231_b(rf_filter_coeff231_b),
```

```
.rf_filter_coeff232_a(rf_filter_coeff232_a),
.rf_filter_coeff232_b(rf_filter_coeff232_b),
.rf_filter_coeff233_a(rf_filter_coeff233_a),
.rf_filter_coeff233_b(rf_filter_coeff233_b),
.rf_filter_coeff234_a(rf_filter_coeff234_a),
.rf_filter_coeff234_b(rf_filter_coeff234_b),
.rf_filter_coeff235_a(rf_filter_coeff235_a),
.rf_filter_coeff235_b(rf_filter_coeff235_b),
.rf_filter_coeff236_a(rf_filter_coeff236_a),
.rf_filter_coeff236_b(rf_filter_coeff236_b),
.rf_filter_coeff237_a(rf_filter_coeff237_a),
.rf_filter_coeff237_b(rf_filter_coeff237_b),
.rf_filter_coeff238_a(rf_filter_coeff238_a),
.rf_filter_coeff238_b(rf_filter_coeff238_b),
.rf_filter_coeff239_a(rf_filter_coeff239_a),
.rf_filter_coeff239_b(rf_filter_coeff239_b),
.rf_filter_coeff240_a(rf_filter_coeff240_a),
.rf_filter_coeff240_b(rf_filter_coeff240_b),
.rf_filter_coeff241_a(rf_filter_coeff241_a),
.rf_filter_coeff241_b(rf_filter_coeff241_b),
.rf_filter_coeff242_a(rf_filter_coeff242_a),
.rf_filter_coeff242_b(rf_filter_coeff242_b),
.rf_filter_coeff243_a(rf_filter_coeff243_a),
.rf_filter_coeff243_b(rf_filter_coeff243_b),
.rf_filter_coeff244_a(rf_filter_coeff244_a),
.rf_filter_coeff244_b(rf_filter_coeff244_b),
.rf_filter_coeff245_a(rf_filter_coeff245_a),
.rf_filter_coeff245_b(rf_filter_coeff245_b),
.rf_filter_coeff246_a(rf_filter_coeff246_a),
.rf_filter_coeff246_b(rf_filter_coeff246_b),
.rf_filter_coeff247_a(rf_filter_coeff247_a),
.rf_filter_coeff247_b(rf_filter_coeff247_b),
.rf_filter_coeff248_a(rf_filter_coeff248_a),
.rf_filter_coeff248_b(rf_filter_coeff248_b),
.rf_filter_coeff249_a(rf_filter_coeff249_a),
.rf_filter_coeff249_b(rf_filter_coeff249_b),
.rf_filter_coeff250_a(rf_filter_coeff250_a),
.rf_filter_coeff250_b(rf_filter_coeff250_b),
.rf_filter_coeff251_a(rf_filter_coeff251_a),
.rf_filter_coeff251_b(rf_filter_coeff251_b),
.rf_filter_coeff252_a(rf_filter_coeff252_a),
.rf_filter_coeff252_b(rf_filter_coeff252_b),
.rf_filter_coeff253_a(rf_filter_coeff253_a),
.rf_filter_coeff253_b(rf_filter_coeff253_b),
.rf_filter_coeff254_a(rf_filter_coeff254_a),
.rf_filter_coeff254_b(rf_filter_coeff254_b),
.rf_filter_coeff255_a(rf_filter_coeff255_a),
.rf_filter_coeff255_b(rf_filter_coeff255_b),
.rf_filter_coeff256_a(rf_filter_coeff256_a),
.rf_filter_coeff256_b(rf_filter_coeff256_b),
.rf_filter_coeff257_a(rf_filter_coeff257_a),
.rf_filter_coeff257_b(rf_filter_coeff257_b),
.rf_filter_coeff258_a(rf_filter_coeff258_a),
.rf_filter_coeff258_b(rf_filter_coeff258_b),
.rf_filter_coeff259_a(rf_filter_coeff259_a),
.rf_filter_coeff259_b(rf_filter_coeff259_b),
.rf_filter_coeff260_a(rf_filter_coeff260_a),
.rf_filter_coeff260_b(rf_filter_coeff260_b),
.rf_filter_coeff261_a(rf_filter_coeff261_a),
.rf_filter_coeff261_b(rf_filter_coeff261_b),
.rf_filter_coeff262_a(rf_filter_coeff262_a),
.rf_filter_coeff262_b(rf_filter_coeff262_b),
.rf_filter_coeff263_a(rf_filter_coeff263_a),
.rf_filter_coeff263_b(rf_filter_coeff263_b),
.rf_filter_coeff264_a(rf_filter_coeff264_a),
.rf_filter_coeff264_b(rf_filter_coeff264_b),
.rf_filter_coeff265_a(rf_filter_coeff265_a),
.rf_filter_coeff265_b(rf_filter_coeff265_b),
.rf_filter_coeff266_a(rf_filter_coeff266_a),
.rf_filter_coeff266_b(rf_filter_coeff266_b),
.rf_filter_coeff267_a(rf_filter_coeff267_a),
```

```
.rf_filter_coeff267_b(rf_filter_coeff267_b),
.rf_filter_coeff268_a(rf_filter_coeff268_a),
.rf_filter_coeff268_b(rf_filter_coeff268_b),
.rf_filter_coeff269_a(rf_filter_coeff269_a),
.rf_filter_coeff269_b(rf_filter_coeff269_b),
.rf_filter_coeff270_a(rf_filter_coeff270_a),
.rf_filter_coeff270_b(rf_filter_coeff270_b),
.rf_filter_coeff271_a(rf_filter_coeff271_a),
.rf_filter_coeff271_b(rf_filter_coeff271_b),
.rf_filter_coeff272_a(rf_filter_coeff272_a),
.rf_filter_coeff272_b(rf_filter_coeff272_b),
.rf_filter_coeff273_a(rf_filter_coeff273_a),
.rf_filter_coeff273_b(rf_filter_coeff273_b),
.rf_filter_coeff274_a(rf_filter_coeff274_a),
.rf_filter_coeff274_b(rf_filter_coeff274_b),
.rf_filter_coeff275_a(rf_filter_coeff275_a),
.rf_filter_coeff275_b(rf_filter_coeff275_b),
.rf_filter_coeff276_a(rf_filter_coeff276_a),
.rf_filter_coeff276_b(rf_filter_coeff276_b),
.rf_filter_coeff277_a(rf_filter_coeff277_a),
.rf_filter_coeff277_b(rf_filter_coeff277_b),
.rf_filter_coeff278_a(rf_filter_coeff278_a),
.rf_filter_coeff278_b(rf_filter_coeff278_b),
.rf_filter_coeff279_a(rf_filter_coeff279_a),
.rf_filter_coeff279_b(rf_filter_coeff279_b),
.rf_filter_coeff280_a(rf_filter_coeff280_a),
.rf_filter_coeff280_b(rf_filter_coeff280_b),
.rf_filter_coeff281_a(rf_filter_coeff281_a),
.rf_filter_coeff281_b(rf_filter_coeff281_b),
.rf_filter_coeff282_a(rf_filter_coeff282_a),
.rf_filter_coeff282_b(rf_filter_coeff282_b),
.rf_filter_coeff283_a(rf_filter_coeff283_a),
.rf_filter_coeff283_b(rf_filter_coeff283_b),
.rf_filter_coeff284_a(rf_filter_coeff284_a),
.rf_filter_coeff284_b(rf_filter_coeff284_b),
.rf_filter_coeff285_a(rf_filter_coeff285_a),
.rf_filter_coeff285_b(rf_filter_coeff285_b),
.rf_filter_coeff286_a(rf_filter_coeff286_a),
.rf_filter_coeff286_b(rf_filter_coeff286_b),
.rf_filter_coeff287_a(rf_filter_coeff287_a),
.rf_filter_coeff287_b(rf_filter_coeff287_b),
.rf_filter_coeff288_a(rf_filter_coeff288_a),
.rf_filter_coeff288_b(rf_filter_coeff288_b),
.rf_filter_coeff289_a(rf_filter_coeff289_a),
.rf_filter_coeff289_b(rf_filter_coeff289_b),
.rf_filter_coeff290_a(rf_filter_coeff290_a),
.rf_filter_coeff290_b(rf_filter_coeff290_b),
.rf_filter_coeff291_a(rf_filter_coeff291_a),
.rf_filter_coeff291_b(rf_filter_coeff291_b),
.rf_filter_coeff292_a(rf_filter_coeff292_a),
.rf_filter_coeff292_b(rf_filter_coeff292_b),
.rf_filter_coeff293_a(rf_filter_coeff293_a),
.rf_filter_coeff293_b(rf_filter_coeff293_b),
.rf_filter_coeff294_a(rf_filter_coeff294_a),
.rf_filter_coeff294_b(rf_filter_coeff294_b),
.rf_filter_coeff295_a(rf_filter_coeff295_a),
.rf_filter_coeff295_b(rf_filter_coeff295_b),
.rf_filter_coeff296_a(rf_filter_coeff296_a),
.rf_filter_coeff296_b(rf_filter_coeff296_b),
.rf_filter_coeff297_a(rf_filter_coeff297_a),
.rf_filter_coeff297_b(rf_filter_coeff297_b),
.rf_filter_coeff298_a(rf_filter_coeff298_a),
.rf_filter_coeff298_b(rf_filter_coeff298_b),
.rf_filter_coeff299_a(rf_filter_coeff299_a),
.rf_filter_coeff299_b(rf_filter_coeff299_b),
.rf_filter_coeff300_a(rf_filter_coeff300_a),
.rf_filter_coeff300_b(rf_filter_coeff300_b),
.rf_filter_coeff301_a(rf_filter_coeff301_a),
.rf_filter_coeff301_b(rf_filter_coeff301_b),
.rf_filter_coeff302_a(rf_filter_coeff302_a),
.rf_filter_coeff302_b(rf_filter_coeff302_b),
```

```
.rf_filter_coeff303_a(rf_filter_coeff303_a),
.rf_filter_coeff303_b(rf_filter_coeff303_b),
.rf_filter_coeff304_a(rf_filter_coeff304_a),
.rf_filter_coeff304_b(rf_filter_coeff304_b),
.rf_filter_coeff305_a(rf_filter_coeff305_a),
.rf_filter_coeff305_b(rf_filter_coeff305_b),
.rf_filter_coeff306_a(rf_filter_coeff306_a),
.rf_filter_coeff306_b(rf_filter_coeff306_b),
.rf_filter_coeff307_a(rf_filter_coeff307_a),
.rf_filter_coeff307_b(rf_filter_coeff307_b),
.rf_filter_coeff308_a(rf_filter_coeff308_a),
.rf_filter_coeff308_b(rf_filter_coeff308_b),
.rf_filter_coeff309_a(rf_filter_coeff309_a),
.rf_filter_coeff309_b(rf_filter_coeff309_b),
.rf_filter_coeff310_a(rf_filter_coeff310_a),
.rf_filter_coeff310_b(rf_filter_coeff310_b),
.rf_filter_coeff311_a(rf_filter_coeff311_a),
.rf_filter_coeff311_b(rf_filter_coeff311_b),
.rf_filter_coeff312_a(rf_filter_coeff312_a),
.rf_filter_coeff312_b(rf_filter_coeff312_b),
.rf_filter_coeff313_a(rf_filter_coeff313_a),
.rf_filter_coeff313_b(rf_filter_coeff313_b),
.rf_filter_coeff314_a(rf_filter_coeff314_a),
.rf_filter_coeff314_b(rf_filter_coeff314_b),
.rf_filter_coeff315_a(rf_filter_coeff315_a),
.rf_filter_coeff315_b(rf_filter_coeff315_b),
.rf_filter_coeff316_a(rf_filter_coeff316_a),
.rf_filter_coeff316_b(rf_filter_coeff316_b),
.rf_filter_coeff317_a(rf_filter_coeff317_a),
.rf_filter_coeff317_b(rf_filter_coeff317_b),
.rf_filter_coeff318_a(rf_filter_coeff318_a),
.rf_filter_coeff318_b(rf_filter_coeff318_b),
.rf_filter_coeff319_a(rf_filter_coeff319_a),
.rf_filter_coeff319_b(rf_filter_coeff319_b),
.rf_filter_coeff320_a(rf_filter_coeff320_a),
.rf_filter_coeff320_b(rf_filter_coeff320_b),
.rf_filter_coeff321_a(rf_filter_coeff321_a),
.rf_filter_coeff321_b(rf_filter_coeff321_b),
.rf_filter_coeff322_a(rf_filter_coeff322_a),
.rf_filter_coeff322_b(rf_filter_coeff322_b),
.rf_filter_coeff323_a(rf_filter_coeff323_a),
.rf_filter_coeff323_b(rf_filter_coeff323_b),
.rf_filter_coeff324_a(rf_filter_coeff324_a),
.rf_filter_coeff324_b(rf_filter_coeff324_b),
.rf_filter_coeff325_a(rf_filter_coeff325_a),
.rf_filter_coeff325_b(rf_filter_coeff325_b),
.rf_filter_coeff326_a(rf_filter_coeff326_a),
.rf_filter_coeff326_b(rf_filter_coeff326_b),
.rf_filter_coeff327_a(rf_filter_coeff327_a),
.rf_filter_coeff327_b(rf_filter_coeff327_b),
.rf_filter_coeff328_a(rf_filter_coeff328_a),
.rf_filter_coeff328_b(rf_filter_coeff328_b),
.rf_filter_coeff329_a(rf_filter_coeff329_a),
.rf_filter_coeff329_b(rf_filter_coeff329_b),
.rf_filter_coeff330_a(rf_filter_coeff330_a),
.rf_filter_coeff330_b(rf_filter_coeff330_b),
.rf_filter_coeff331_a(rf_filter_coeff331_a),
.rf_filter_coeff331_b(rf_filter_coeff331_b),
.rf_filter_coeff332_a(rf_filter_coeff332_a),
.rf_filter_coeff332_b(rf_filter_coeff332_b),
.rf_filter_coeff333_a(rf_filter_coeff333_a),
.rf_filter_coeff333_b(rf_filter_coeff333_b),
.rf_filter_coeff334_a(rf_filter_coeff334_a),
.rf_filter_coeff334_b(rf_filter_coeff334_b),
.rf_filter_coeff335_a(rf_filter_coeff335_a),
.rf_filter_coeff335_b(rf_filter_coeff335_b),
.rf_filter_coeff336_a(rf_filter_coeff336_a),
.rf_filter_coeff336_b(rf_filter_coeff336_b),
.rf_filter_coeff337_a(rf_filter_coeff337_a),
.rf_filter_coeff337_b(rf_filter_coeff337_b),
.rf_filter_coeff338_a(rf_filter_coeff338_a),
```

```
.rf_filter_coeff338_b(rf_filter_coeff338_b),
.rf_filter_coeff339_a(rf_filter_coeff339_a),
.rf_filter_coeff339_b(rf_filter_coeff339_b),
.rf_filter_coeff340_a(rf_filter_coeff340_a),
.rf_filter_coeff340_b(rf_filter_coeff340_b),
.rf_filter_coeff341_a(rf_filter_coeff341_a),
.rf_filter_coeff341_b(rf_filter_coeff341_b),
.rf_filter_coeff342_a(rf_filter_coeff342_a),
.rf_filter_coeff342_b(rf_filter_coeff342_b),
.rf_filter_coeff343_a(rf_filter_coeff343_a),
.rf_filter_coeff343_b(rf_filter_coeff343_b),
.rf_filter_coeff344_a(rf_filter_coeff344_a),
.rf_filter_coeff344_b(rf_filter_coeff344_b),
.rf_filter_coeff345_a(rf_filter_coeff345_a),
.rf_filter_coeff345_b(rf_filter_coeff345_b),
.rf_filter_coeff346_a(rf_filter_coeff346_a),
.rf_filter_coeff346_b(rf_filter_coeff346_b),
.rf_filter_coeff347_a(rf_filter_coeff347_a),
.rf_filter_coeff347_b(rf_filter_coeff347_b),
.rf_filter_coeff348_a(rf_filter_coeff348_a),
.rf_filter_coeff348_b(rf_filter_coeff348_b),
.rf_filter_coeff349_a(rf_filter_coeff349_a),
.rf_filter_coeff349_b(rf_filter_coeff349_b),
.rf_filter_coeff350_a(rf_filter_coeff350_a),
.rf_filter_coeff350_b(rf_filter_coeff350_b),
.rf_filter_coeff351_a(rf_filter_coeff351_a),
.rf_filter_coeff351_b(rf_filter_coeff351_b),
.rf_filter_coeff352_a(rf_filter_coeff352_a),
.rf_filter_coeff352_b(rf_filter_coeff352_b),
.rf_filter_coeff353_a(rf_filter_coeff353_a),
.rf_filter_coeff353_b(rf_filter_coeff353_b),
.rf_filter_coeff354_a(rf_filter_coeff354_a),
.rf_filter_coeff354_b(rf_filter_coeff354_b),
.rf_filter_coeff355_a(rf_filter_coeff355_a),
.rf_filter_coeff355_b(rf_filter_coeff355_b),
.rf_filter_coeff356_a(rf_filter_coeff356_a),
.rf_filter_coeff356_b(rf_filter_coeff356_b),
.rf_filter_coeff357_a(rf_filter_coeff357_a),
.rf_filter_coeff357_b(rf_filter_coeff357_b),
.rf_filter_coeff358_a(rf_filter_coeff358_a),
.rf_filter_coeff358_b(rf_filter_coeff358_b),
.rf_filter_coeff359_a(rf_filter_coeff359_a),
.rf_filter_coeff359_b(rf_filter_coeff359_b),
.rf_filter_coeff360_a(rf_filter_coeff360_a),
.rf_filter_coeff360_b(rf_filter_coeff360_b),
.rf_filter_coeff361_a(rf_filter_coeff361_a),
.rf_filter_coeff361_b(rf_filter_coeff361_b),
.rf_filter_coeff362_a(rf_filter_coeff362_a),
.rf_filter_coeff362_b(rf_filter_coeff362_b),
.rf_filter_coeff363_a(rf_filter_coeff363_a),
.rf_filter_coeff363_b(rf_filter_coeff363_b),
.rf_filter_coeff364_a(rf_filter_coeff364_a),
.rf_filter_coeff364_b(rf_filter_coeff364_b),
.rf_filter_coeff365_a(rf_filter_coeff365_a),
.rf_filter_coeff365_b(rf_filter_coeff365_b),
.rf_filter_coeff366_a(rf_filter_coeff366_a),
.rf_filter_coeff366_b(rf_filter_coeff366_b),
.rf_filter_coeff367_a(rf_filter_coeff367_a),
.rf_filter_coeff367_b(rf_filter_coeff367_b),
.rf_filter_coeff368_a(rf_filter_coeff368_a),
.rf_filter_coeff368_b(rf_filter_coeff368_b),
.rf_filter_coeff369_a(rf_filter_coeff369_a),
.rf_filter_coeff369_b(rf_filter_coeff369_b),
.rf_filter_coeff370_a(rf_filter_coeff370_a),
.rf_filter_coeff370_b(rf_filter_coeff370_b),
.rf_filter_coeff371_a(rf_filter_coeff371_a),
.rf_filter_coeff371_b(rf_filter_coeff371_b),
.rf_filter_coeff372_a(rf_filter_coeff372_a),
.rf_filter_coeff372_b(rf_filter_coeff372_b),
.rf_filter_coeff373_a(rf_filter_coeff373_a),
.rf_filter_coeff373_b(rf_filter_coeff373_b),
```

```
.rf_filter_coeff374_a(rf_filter_coeff374_a),
.rf_filter_coeff374_b(rf_filter_coeff374_b),
.rf_filter_coeff375_a(rf_filter_coeff375_a),
.rf_filter_coeff375_b(rf_filter_coeff375_b),
.rf_filter_coeff376_a(rf_filter_coeff376_a),
.rf_filter_coeff376_b(rf_filter_coeff376_b),
.rf_filter_coeff377_a(rf_filter_coeff377_a),
.rf_filter_coeff377_b(rf_filter_coeff377_b),
.rf_filter_coeff378_a(rf_filter_coeff378_a),
.rf_filter_coeff378_b(rf_filter_coeff378_b),
.rf_filter_coeff379_a(rf_filter_coeff379_a),
.rf_filter_coeff379_b(rf_filter_coeff379_b),
.rf_filter_coeff380_a(rf_filter_coeff380_a),
.rf_filter_coeff380_b(rf_filter_coeff380_b),
.rf_filter_coeff381_a(rf_filter_coeff381_a),
.rf_filter_coeff381_b(rf_filter_coeff381_b),
.rf_filter_coeff382_a(rf_filter_coeff382_a),
.rf_filter_coeff382_b(rf_filter_coeff382_b),
.rf_filter_coeff383_a(rf_filter_coeff383_a),
.rf_filter_coeff383_b(rf_filter_coeff383_b),
.rf_filter_coeff384_a(rf_filter_coeff384_a),
.rf_filter_coeff384_b(rf_filter_coeff384_b),
.rf_filter_coeff385_a(rf_filter_coeff385_a),
.rf_filter_coeff385_b(rf_filter_coeff385_b),
.rf_filter_coeff386_a(rf_filter_coeff386_a),
.rf_filter_coeff386_b(rf_filter_coeff386_b),
.rf_filter_coeff387_a(rf_filter_coeff387_a),
.rf_filter_coeff387_b(rf_filter_coeff387_b),
.rf_filter_coeff388_a(rf_filter_coeff388_a),
.rf_filter_coeff388_b(rf_filter_coeff388_b),
.rf_filter_coeff389_a(rf_filter_coeff389_a),
.rf_filter_coeff389_b(rf_filter_coeff389_b),
.rf_filter_coeff390_a(rf_filter_coeff390_a),
.rf_filter_coeff390_b(rf_filter_coeff390_b),
.rf_filter_coeff391_a(rf_filter_coeff391_a),
.rf_filter_coeff391_b(rf_filter_coeff391_b),
.rf_filter_coeff392_a(rf_filter_coeff392_a),
.rf_filter_coeff392_b(rf_filter_coeff392_b),
.rf_filter_coeff393_a(rf_filter_coeff393_a),
.rf_filter_coeff393_b(rf_filter_coeff393_b),
.rf_filter_coeff394_a(rf_filter_coeff394_a),
.rf_filter_coeff394_b(rf_filter_coeff394_b),
.rf_filter_coeff395_a(rf_filter_coeff395_a),
.rf_filter_coeff395_b(rf_filter_coeff395_b),
.rf_filter_coeff396_a(rf_filter_coeff396_a),
.rf_filter_coeff396_b(rf_filter_coeff396_b),
.rf_filter_coeff397_a(rf_filter_coeff397_a),
.rf_filter_coeff397_b(rf_filter_coeff397_b),
.rf_filter_coeff398_a(rf_filter_coeff398_a),
.rf_filter_coeff398_b(rf_filter_coeff398_b),
.rf_filter_coeff399_a(rf_filter_coeff399_a),
.rf_filter_coeff399_b(rf_filter_coeff399_b),
.rf_filter_coeff400_a(rf_filter_coeff400_a),
.rf_filter_coeff400_b(rf_filter_coeff400_b),
.rf_filter_coeff401_a(rf_filter_coeff401_a),
.rf_filter_coeff401_b(rf_filter_coeff401_b),
.rf_filter_coeff402_a(rf_filter_coeff402_a),
.rf_filter_coeff402_b(rf_filter_coeff402_b),
.rf_filter_coeff403_a(rf_filter_coeff403_a),
.rf_filter_coeff403_b(rf_filter_coeff403_b),
.rf_filter_coeff404_a(rf_filter_coeff404_a),
.rf_filter_coeff404_b(rf_filter_coeff404_b),
.rf_filter_coeff405_a(rf_filter_coeff405_a),
.rf_filter_coeff405_b(rf_filter_coeff405_b),
.rf_filter_coeff406_a(rf_filter_coeff406_a),
.rf_filter_coeff406_b(rf_filter_coeff406_b),
.rf_filter_coeff407_a(rf_filter_coeff407_a),
.rf_filter_coeff407_b(rf_filter_coeff407_b),
.rf_filter_coeff408_a(rf_filter_coeff408_a),
.rf_filter_coeff408_b(rf_filter_coeff408_b),
.rf_filter_coeff409_a(rf_filter_coeff409_a),
```

```
.rf_filter_coeff409_b(rf_filter_coeff409_b),
.rf_filter_coeff410_a(rf_filter_coeff410_a),
.rf_filter_coeff410_b(rf_filter_coeff410_b),
.rf_filter_coeff411_a(rf_filter_coeff411_a),
.rf_filter_coeff411_b(rf_filter_coeff411_b),
.rf_filter_coeff412_a(rf_filter_coeff412_a),
.rf_filter_coeff412_b(rf_filter_coeff412_b),
.rf_filter_coeff413_a(rf_filter_coeff413_a),
.rf_filter_coeff413_b(rf_filter_coeff413_b),
.rf_filter_coeff414_a(rf_filter_coeff414_a),
.rf_filter_coeff414_b(rf_filter_coeff414_b),
.rf_filter_coeff415_a(rf_filter_coeff415_a),
.rf_filter_coeff415_b(rf_filter_coeff415_b),
.rf_filter_coeff416_a(rf_filter_coeff416_a),
.rf_filter_coeff416_b(rf_filter_coeff416_b),
.rf_filter_coeff417_a(rf_filter_coeff417_a),
.rf_filter_coeff417_b(rf_filter_coeff417_b),
.rf_filter_coeff418_a(rf_filter_coeff418_a),
.rf_filter_coeff418_b(rf_filter_coeff418_b),
.rf_filter_coeff419_a(rf_filter_coeff419_a),
.rf_filter_coeff419_b(rf_filter_coeff419_b),
.rf_filter_coeff420_a(rf_filter_coeff420_a),
.rf_filter_coeff420_b(rf_filter_coeff420_b),
.rf_filter_coeff421_a(rf_filter_coeff421_a),
.rf_filter_coeff421_b(rf_filter_coeff421_b),
.rf_filter_coeff422_a(rf_filter_coeff422_a),
.rf_filter_coeff422_b(rf_filter_coeff422_b),
.rf_filter_coeff423_a(rf_filter_coeff423_a),
.rf_filter_coeff423_b(rf_filter_coeff423_b),
.rf_filter_coeff424_a(rf_filter_coeff424_a),
.rf_filter_coeff424_b(rf_filter_coeff424_b),
.rf_filter_coeff425_a(rf_filter_coeff425_a),
.rf_filter_coeff425_b(rf_filter_coeff425_b),
.rf_filter_coeff426_a(rf_filter_coeff426_a),
.rf_filter_coeff426_b(rf_filter_coeff426_b),
.rf_filter_coeff427_a(rf_filter_coeff427_a),
.rf_filter_coeff427_b(rf_filter_coeff427_b),
.rf_filter_coeff428_a(rf_filter_coeff428_a),
.rf_filter_coeff428_b(rf_filter_coeff428_b),
.rf_filter_coeff429_a(rf_filter_coeff429_a),
.rf_filter_coeff429_b(rf_filter_coeff429_b),
.rf_filter_coeff430_a(rf_filter_coeff430_a),
.rf_filter_coeff430_b(rf_filter_coeff430_b),
.rf_filter_coeff431_a(rf_filter_coeff431_a),
.rf_filter_coeff431_b(rf_filter_coeff431_b),
.rf_filter_coeff432_a(rf_filter_coeff432_a),
.rf_filter_coeff432_b(rf_filter_coeff432_b),
.rf_filter_coeff433_a(rf_filter_coeff433_a),
.rf_filter_coeff433_b(rf_filter_coeff433_b),
.rf_filter_coeff434_a(rf_filter_coeff434_a),
.rf_filter_coeff434_b(rf_filter_coeff434_b),
.rf_filter_coeff435_a(rf_filter_coeff435_a),
.rf_filter_coeff435_b(rf_filter_coeff435_b),
.rf_filter_coeff436_a(rf_filter_coeff436_a),
.rf_filter_coeff436_b(rf_filter_coeff436_b),
.rf_filter_coeff437_a(rf_filter_coeff437_a),
.rf_filter_coeff437_b(rf_filter_coeff437_b),
.rf_filter_coeff438_a(rf_filter_coeff438_a),
.rf_filter_coeff438_b(rf_filter_coeff438_b),
.rf_filter_coeff439_a(rf_filter_coeff439_a),
.rf_filter_coeff439_b(rf_filter_coeff439_b),
.rf_filter_coeff440_a(rf_filter_coeff440_a),
.rf_filter_coeff440_b(rf_filter_coeff440_b),
.rf_filter_coeff441_a(rf_filter_coeff441_a),
.rf_filter_coeff441_b(rf_filter_coeff441_b),
.rf_filter_coeff442_a(rf_filter_coeff442_a),
.rf_filter_coeff442_b(rf_filter_coeff442_b),
.rf_filter_coeff443_a(rf_filter_coeff443_a),
.rf_filter_coeff443_b(rf_filter_coeff443_b),
.rf_filter_coeff444_a(rf_filter_coeff444_a),
.rf_filter_coeff444_b(rf_filter_coeff444_b),
```

```
.rf_filter_coeff445_a(rf_filter_coeff445_a),
.rf_filter_coeff445_b(rf_filter_coeff445_b),
.rf_filter_coeff446_a(rf_filter_coeff446_a),
.rf_filter_coeff446_b(rf_filter_coeff446_b),
.rf_filter_coeff447_a(rf_filter_coeff447_a),
.rf_filter_coeff447_b(rf_filter_coeff447_b),
.rf_filter_coeff448_a(rf_filter_coeff448_a),
.rf_filter_coeff448_b(rf_filter_coeff448_b),
.rf_filter_coeff449_a(rf_filter_coeff449_a),
.rf_filter_coeff449_b(rf_filter_coeff449_b),
.rf_filter_coeff450_a(rf_filter_coeff450_a),
.rf_filter_coeff450_b(rf_filter_coeff450_b),
.rf_filter_coeff451_a(rf_filter_coeff451_a),
.rf_filter_coeff451_b(rf_filter_coeff451_b),
.rf_filter_coeff452_a(rf_filter_coeff452_a),
.rf_filter_coeff452_b(rf_filter_coeff452_b),
.rf_filter_coeff453_a(rf_filter_coeff453_a),
.rf_filter_coeff453_b(rf_filter_coeff453_b),
.rf_filter_coeff454_a(rf_filter_coeff454_a),
.rf_filter_coeff454_b(rf_filter_coeff454_b),
.rf_filter_coeff455_a(rf_filter_coeff455_a),
.rf_filter_coeff455_b(rf_filter_coeff455_b),
.rf_filter_coeff456_a(rf_filter_coeff456_a),
.rf_filter_coeff456_b(rf_filter_coeff456_b),
.rf_filter_coeff457_a(rf_filter_coeff457_a),
.rf_filter_coeff457_b(rf_filter_coeff457_b),
.rf_filter_coeff458_a(rf_filter_coeff458_a),
.rf_filter_coeff458_b(rf_filter_coeff458_b),
.rf_filter_coeff459_a(rf_filter_coeff459_a),
.rf_filter_coeff459_b(rf_filter_coeff459_b),
.rf_filter_coeff460_a(rf_filter_coeff460_a),
.rf_filter_coeff460_b(rf_filter_coeff460_b),
.rf_filter_coeff461_a(rf_filter_coeff461_a),
.rf_filter_coeff461_b(rf_filter_coeff461_b),
.rf_filter_coeff462_a(rf_filter_coeff462_a),
.rf_filter_coeff462_b(rf_filter_coeff462_b),
.rf_filter_coeff463_a(rf_filter_coeff463_a),
.rf_filter_coeff463_b(rf_filter_coeff463_b),
.rf_filter_coeff464_a(rf_filter_coeff464_a),
.rf_filter_coeff464_b(rf_filter_coeff464_b),
.rf_filter_coeff465_a(rf_filter_coeff465_a),
.rf_filter_coeff465_b(rf_filter_coeff465_b),
.rf_filter_coeff466_a(rf_filter_coeff466_a),
.rf_filter_coeff466_b(rf_filter_coeff466_b),
.rf_filter_coeff467_a(rf_filter_coeff467_a),
.rf_filter_coeff467_b(rf_filter_coeff467_b),
.rf_filter_coeff468_a(rf_filter_coeff468_a),
.rf_filter_coeff468_b(rf_filter_coeff468_b),
.rf_filter_coeff469_a(rf_filter_coeff469_a),
.rf_filter_coeff469_b(rf_filter_coeff469_b),
.rf_filter_coeff470_a(rf_filter_coeff470_a),
.rf_filter_coeff470_b(rf_filter_coeff470_b),
.rf_filter_coeff471_a(rf_filter_coeff471_a),
.rf_filter_coeff471_b(rf_filter_coeff471_b),
.rf_filter_coeff472_a(rf_filter_coeff472_a),
.rf_filter_coeff472_b(rf_filter_coeff472_b),
.rf_filter_coeff473_a(rf_filter_coeff473_a),
.rf_filter_coeff473_b(rf_filter_coeff473_b),
.rf_filter_coeff474_a(rf_filter_coeff474_a),
.rf_filter_coeff474_b(rf_filter_coeff474_b),
.rf_filter_coeff475_a(rf_filter_coeff475_a),
.rf_filter_coeff475_b(rf_filter_coeff475_b),
.rf_filter_coeff476_a(rf_filter_coeff476_a),
.rf_filter_coeff476_b(rf_filter_coeff476_b),
.rf_filter_coeff477_a(rf_filter_coeff477_a),
.rf_filter_coeff477_b(rf_filter_coeff477_b),
.rf_filter_coeff478_a(rf_filter_coeff478_a),
.rf_filter_coeff478_b(rf_filter_coeff478_b),
.rf_filter_coeff479_a(rf_filter_coeff479_a),
.rf_filter_coeff479_b(rf_filter_coeff479_b),
.rf_filter_coeff480_a(rf_filter_coeff480_a),
```

```
        .rf_filter_coeff480_b(rf_filter_coeff480_b),
        .rf_filter_coeff481_a(rf_filter_coeff481_a),
        .rf_filter_coeff481_b(rf_filter_coeff481_b),
        .rf_filter_coeff482_a(rf_filter_coeff482_a),
        .rf_filter_coeff482_b(rf_filter_coeff482_b),
        .rf_filter_coeff483_a(rf_filter_coeff483_a),
        .rf_filter_coeff483_b(rf_filter_coeff483_b),
        .rf_filter_coeff484_a(rf_filter_coeff484_a),
        .rf_filter_coeff484_b(rf_filter_coeff484_b),
        .rf_filter_coeff485_a(rf_filter_coeff485_a),
        .rf_filter_coeff485_b(rf_filter_coeff485_b),
        .rf_filter_coeff486_a(rf_filter_coeff486_a),
        .rf_filter_coeff486_b(rf_filter_coeff486_b),
        .rf_filter_coeff487_a(rf_filter_coeff487_a),
        .rf_filter_coeff487_b(rf_filter_coeff487_b),
        .rf_filter_coeff488_a(rf_filter_coeff488_a),
        .rf_filter_coeff488_b(rf_filter_coeff488_b),
        .rf_filter_coeff489_a(rf_filter_coeff489_a),
        .rf_filter_coeff489_b(rf_filter_coeff489_b),
        .rf_filter_coeff490_a(rf_filter_coeff490_a),
        .rf_filter_coeff490_b(rf_filter_coeff490_b),
        .rf_filter_coeff491_a(rf_filter_coeff491_a),
        .rf_filter_coeff491_b(rf_filter_coeff491_b),
        .rf_filter_coeff492_a(rf_filter_coeff492_a),
        .rf_filter_coeff492_b(rf_filter_coeff492_b),
        .rf_filter_coeff493_a(rf_filter_coeff493_a),
        .rf_filter_coeff493_b(rf_filter_coeff493_b),
        .rf_filter_coeff494_a(rf_filter_coeff494_a),
        .rf_filter_coeff494_b(rf_filter_coeff494_b),
        .rf_filter_coeff495_a(rf_filter_coeff495_a),
        .rf_filter_coeff495_b(rf_filter_coeff495_b),
        .rf_filter_coeff496_a(rf_filter_coeff496_a),
        .rf_filter_coeff496_b(rf_filter_coeff496_b),
        .rf_filter_coeff497_a(rf_filter_coeff497_a),
        .rf_filter_coeff497_b(rf_filter_coeff497_b),
        .rf_filter_coeff498_a(rf_filter_coeff498_a),
        .rf_filter_coeff498_b(rf_filter_coeff498_b),
        .rf_filter_coeff499_a(rf_filter_coeff499_a),
        .rf_filter_coeff499_b(rf_filter_coeff499_b),
        .rf_filter_coeff500_a(rf_filter_coeff500_a),
        .rf_filter_coeff500_b(rf_filter_coeff500_b),
        .rf_filter_coeff501_a(rf_filter_coeff501_a),
        .rf_filter_coeff501_b(rf_filter_coeff501_b),
        .rf_filter_coeff502_a(rf_filter_coeff502_a),
        .rf_filter_coeff502_b(rf_filter_coeff502_b),
        .rf_filter_coeff503_a(rf_filter_coeff503_a),
        .rf_filter_coeff503_b(rf_filter_coeff503_b),
        .rf_filter_coeff504_a(rf_filter_coeff504_a),
        .rf_filter_coeff504_b(rf_filter_coeff504_b),
        .rf_filter_coeff505_a(rf_filter_coeff505_a),
        .rf_filter_coeff505_b(rf_filter_coeff505_b),
        .rf_filter_coeff506_a(rf_filter_coeff506_a),
        .rf_filter_coeff506_b(rf_filter_coeff506_b),
        .rf_filter_coeff507_a(rf_filter_coeff507_a),
        .rf_filter_coeff507_b(rf_filter_coeff507_b),
        .rf_filter_coeff508_a(rf_filter_coeff508_a),
        .rf_filter_coeff508_b(rf_filter_coeff508_b),
        .rf_filter_coeff509_a(rf_filter_coeff509_a),
        .rf_filter_coeff509_b(rf_filter_coeff509_b),
        .rf_filter_coeff510_a(rf_filter_coeff510_a),
        .rf_filter_coeff510_b(rf_filter_coeff510_b),
        .rf_filter_coeff511_a(rf_filter_coeff511_a),
        .rf_filter_coeff511_b(rf_filter_coeff511_b)
);

initial begin
        // Initialize Inputs
        rst_n = 0;
        clk = 0;
        addr = 0;
        wdata = 0;
```

```
                w_enable = 0;
                wxfc = 0;
                rxfc = 0;
                ro_fifo_underrun = 0;
                ro_fifo_overrun = 0;
                ro_filter_ovf_flag = 0;
                ro_filter_ovf_flag_clear = 0;
                rf_i2so_clk2sck_div_a = 0;
                rf_i2so_clk2sck_div_b = 0;

                // Wait 100 ns for global reset to finish
                #100;

                // Add stimulus here


        end

endmodule
```

trig_generator_testbench.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             trig_generator_testbench.v
// Create Date:             10/20/2015
// Last Modification:       3/16/2016
// Author:                  Julie Swift
// Description: ???????????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module trig_generator_testbench;

          // Inputs
          reg         [10:0]          address;
          reg         [ 7:0]          wdata;
          reg                         wxfc;
          reg                         clk;
          reg         [31:0]          count;
          wire                        rst_n;

          // Outputs
          wire                        trig_i2si_fifo_overrun_clr;
          wire                        trig_i2so_fifo_underrun_clr;

          // Instantiate the Unit Under Test (UUT)
          trig_generator uut (
                    .address(address),
                    .wdata(wdata),
                    .wxfc(wxfc),
                    .clk(clk),
                    .rst_n(rst_n),
                    .trig_i2si_fifo_overrun_clr(trig_i2si_fifo_overrun_clr),
                    .trig_i2so_fifo_underrun_clr(trig_i2so_fifo_underrun_clr)
          );

          always
          begin
        forever
                    begin
                              #5 clk = ~clk;
                              count = count + 1;
                    end
          end

          assign rst_n = !(count < 20);

          initial begin
                    // Initialize Inputs
        count = 0;
        clk = 0;
                    wdata = 8'hFF;

                    // Wait 100 ns for global reset to finish
                    #1000;
    end

always @(posedge clk or negedge rst_n)
          begin
                    if (~rst_n)
                    begin
                              address <= 0;
                              xfc <= 0;
                    end
                    else if (address < 11'h20) //hex 20 12 bits of data
                    begin
                              address <= address + 4;
                              xfc <= 1;
                    end
                    else
```

```
                        xfc <=0;
        end

endmodule
```

trig_generator_testbench1.v:

```
////////////////////////////////////////////////////////////////////////////////
// Module Name:             trig_generator_testbench1.v
// Create Date:             10/20/2015
// Last Modification:       1/13/2015
// Author:                  Julie Swift
// Description: ???????????
////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

module trig_generator_testbench1;

          // Inputs
          reg         [10:0]          address;
          reg         [ 7:0]          wdata;
          reg                         xfc;
          reg                         clk;
          reg         [31:0]          count;
          wire                        rst_n;


          // Outputs
          wire                        trig_i2si_fifo_overrun_clr;
          wire                        trig_i2so_fifo_underrun_clr;

          // Instantiate the Unit Under Test (UUT)
          trig_generator uut (
                    .address(address),
                    .wdata(wdata),
                    .xfc(xfc),
                    .clk(clk),
                    .rst_n(rst_n),
                    .trig_i2si_fifo_overrun_clr(trig_i2si_fifo_overrun_clr),
                    .trig_i2so_fifo_underrun_clr(trig_i2so_fifo_underrun_clr)
          );
          always
          begin
          // Generates a clock with a clock cycle of 10 ns
        forever
                    begin
                              #5 clk = ~clk;
                              count = count + 1;
                    end
          end

          assign rst_n = !(count < 20);


          initial begin
                    // Initialize Inputs
        count = 0;
        clk = 0;
                    address = 11'h00c;

                    // Wait 100 ns for global reset to finish
                    #1000;
    end

always @(posedge clk or negedge rst_n)
          begin
                    // initializing xfc and wdata to 0
                    if (~rst_n)
                    begin
                              wdata <= 0;
                              xfc <= 0;
                    end
                    /*
                    if wdata is 12 bits of data and less than the hex value 20 wdata is
                    and file transfer is set to 1 - complete
```

```
                    */
                    else if (wdata < 11'h020)
                    begin
                              wdata <= wdata + 1;
                              xfc <= 1;
                    end
                    else
                              xfc <=0;
          end

endmodule
```

Appendix E: Industry Specifications

DIP Package 582
GDSII Stream Format 584
I2C Specification 631
I2S Specification 695

NOTES:

1. Refer to applicable symbol list.
2. Dimensioning and tolerancing per ANSI Y14.5-1973.
3. Leads within .127 radius of True Position (TP) at gauge plane with maximum material condition and unit installed.
4. e_1 and e_A applies in zone L_2 when unit installed.
5. α applies to spread leads prior to installation.
6. N is the maximum quantity of lead positions.
7. N_1 is the allowable quantity of missing leads.
8. E_1 does not include mold flash.
9. Outlines on which the seating plane is coincident with the base plane (A_1 = 0) terminal lead standoffs are not required, and B_1 may equal B along any part of the lead above the seating/base plane.
10. Controlling Dimension: INCH

VARIATIONS (ALL DIMENSIONS IN MILLIMETERS)

SYMBOL	AJ MIN.	AJ MAX.	NOTE	AK MIN.	AK MAX.	NOTE	AL MIN.	AL MAX.	NOTE	AM MIN.	AM MAX.	NOTE
A	2.29	3.55		2.29	3.55		4.07	5.08		2.92	5.46	
A_1	.51	1.77		.51	1.77		.51	1.14		0.38	1.75	
B	.381	.584		.381	.584		.381	.508		0.38	0.53	
B_1	.89	1.39		.89	1.39		1.12	1.77		1.02	1.78	
C	.204	.304		.204	.304		.204	.304		0.20	0.30	
D	17.40	19.30		20.32	21.33		20.71	22.60		8.13	9.39	
E	7.62	8.25	8	7.62	8.25	8	7.37	8.89	8	7.37	8.25	
E_1	6.10	7.23		6.10	7.23		6.10	6.60		5.71	7.11	8
e_1	2.54 TP		3,4	2.54 TP		3,4	2.54 TP		3,4	2.54 TP		3,4
e_A	7.62 TP		3,4	7.62 TP		3,4	7.62 TP		3,4	7.62 TP		3,4
L	2.54	3.81		2.54	3.81		3.05	3.81		2.54	3.81	
L_2	.00	.76		.00	.76		.00	.76		.00	0.76	
α	0°	15°	5	0°	15°	5	0°	15°	5	0°	15°	5
N	14		6	16		5	14		6	6		6
N_1	0		7	0		7	0		7	0		7
Q_1	1.53	2.03		1.53	2.03		1.27	2.15		-	-	
S	1.15	2.28		1.02	2.03		1.33	2.41		1.27	2.28	
NOTE	1,2,10			1,2,10			1,2,10			1,2,10		
REF.										JC-11.3-83-7		
ISSUE	D JUNE 1976			D JUNE 1976			D JUNE 1976			F JUNE 1983		

JEDEC SOLID STATE PRODUCTS OUTLINES	TITLE DUAL IN LINE (DIP) FAMILY 7.62 Row Spacing	ISSUE F	DATE JUNE 1983	MO-001 AJ-AM

NOTES:

1. Refer to applicable symbol list.
2. Dimensioning and tolerancing per ANSI Y14.5-1973.
3. Leads within .005 radius of True Position (TP) at gauge plane with maximum material condition and unit installed.
4. e_1 and e_A applies in zone L_2 when unit installed.
5. α applies to spread leads prior to installation.
6. N is the maximum quantity of lead positions.
7. N_1 is the allowable quantity of missing leads.
8. E_1 does not include mold flash.
9. Outlines on which the seating plane is coincident with the base plane (A_1 = 0) terminal lead standoffs are not required, and B_1 may equal B along any part of the lead above the seating/base plane.
10. Controlling Dimension: INCH

VARIATIONS (ALL DIMENSIONS IN INCHES)

SYMBOL	AJ MIN.	AJ MAX.	NOTE	AK MIN.	AK MAX.	NOTE	AL MIN.	AL MAX.	NOTE	AM MIN.	AM MAX.	NOTE
A A_1	.090 .020	.140 .070		.090 .020	.140 .070		.160 .020	.200 .045		.115 .015	.215 .070	
B B_1	.015 .035	.023 .055		.015 .035	.023 .055		.015 .044	.020 .070		.015 .040	.021 .070	
C D	.008 .685	.012 .760		.008 .800	.012 .840		.008 .815	.012 .890		.008 .320	.012 .370	
E E_1	.300 .240	.325 .285	8	.300 .240	.325 .285	8	.290 .240	.350 .260	8	.290 .225	.325 .280	8
e_1 e_A	.100 TP .300 TP		3,4 3,4	.100 TP .300 TP		3,4 3,4	.100 TP .300 TP		3,4 3,4	.100 TP .300 TP		3,4 3,4
L L_2	.100 .000	.150 .030		.100 .000	.150 .030		.120 .000	.150 .030		.100 .000	.150 .030	
α N	0° 14	15°	5 6	0° 16	15°	5 6	0° 16	15°	5 6	0° 6	15°	5 6
N_1 Q_1	0 .060	.080	7	0 .060	.080	7	0 .050	.085	7	0 -	-	7
S	.045	.090		.040	.080		.052	.095		.050	.090	
NOTE	1,2,10			1,2,10			1,2,10			1,2,10		
REF.										JC-11.3-83-7		
ISSUE	D JUNE 1976			D JUNE 1976			D JUNE 1976			A January 1977		

JEDEC SOLID STATE PRODUCTS OUTLINES	TITLE DUAL IN LINE (DIP) FAMILY .300 Row Spacing	ISSUE F	DATE JUNE 1983	MO-001 AJ-AM

GDSII™ Stream Format Manual

Documentation No.: B97E060

Release 6.0

February 1987

calma

Calma Company (Calma) has prepared this document for use by Calma employees and customers only. The only undertakings of Calma respecting information in this document are contained in contracts between Calma and its customers, and nothing contained in this document shall be construed as changing said contracts. The use of this information except as defined by said contracts, or for any purpose other than that for which it is intended, is not authorized and, with respect to any such unauthorized use, neither Calma nor any of the contributors to this document makes any representation or warranty, nor shall any warranty be implied, as to the completeness, accuracy, or usefulness of the information contained in this document or that such use of such information may not infringe privately owned rights, nor do they assume any responsibility for liability or damage of any kind which may result from such use of such information. This publication contains proprietary information of Calma and is for use by Calma personnel and Calma customers only. Duplication in whole or in part by any means (including xerographic photocopying and/or computerized magnetic tape/disk information storage/retrieval systems) for any purpose without the publisher's express written permission is prohibited. This document may contain portions reproduced with permission/license of the copyright owner.

Copyright© February 1987, Calma Company.
Proprietary and trade secret.
Published only in a limited, copyright sense.

Contents

1.0 Notes to the User 1-1
2.0 Record Description 2-1
3.0 Data Type Description 3-1
4.0 Record Types 4-1
5.0 Stream Syntax 5-1
Index Index-1

Figures

2-1 Typical Record Header 2-1

4-1 A BGNLIB Record 4-2
4-2 An Array Lattice 4-7
4-3 A PRESENTATION Record 4-9
4-4 An STRANS Record 4-10
4-5 Pathtypes 4-13
4-6 An ELFLAGS Record 4-15

Tables

3-1 Stream Data Types 3-1

4-1 GDSII Release and Version numbers 4-1

5-1 Bachus Naur Symbols 5-1

1.0 Notes to the User

Stream format is the standard output format for GDSII data. Stream format is the format written by OUTFORM and STREAMOUT and read by INFORM. Libraries preserved in this format can be easily transferred to other systems for processing. Stream format is upward compatible between releases. Libraries archived under an old release will always be readable by newer releases. For this reason, libraries preserved in Stream format can be archived.

Sections 2 through 4 describe the Stream format components. Sections 5 and 6 describe the Stream syntax. Section 7 provides an example and description of a Stream format file.

2.0 Record Description

The Stream format output file is composed of variable length records. The minimum record length is four bytes. Records can be infinitely long. The first four bytes of a record are the header. The first two bytes of the header contain a count (in eight-bit bytes) of the total record length. The count tells you where one record ends and another begins. The next record begins immediately after the last byte included in the count.

The third byte of the header is the record type. The fourth byte of the header describes the type of data contained within the record. The fifth through last bytes of a record are data. *Figure 2-1* shows a typical record header.

Figure 2-1. Typical Record Header

If the output file is a magnetic tape, then the records of the library are written out in 2048-byte physical blocks. Records may overlap physical block boundaries; a record is not required to be wholly contained in a single physical block.

A null word consists of two consecutive zero bytes. Use null words to fill the space between

- the last record of a library and the end of its physical block, or
- the last record of a tape in a multi-reel Stream file and the end of its physical block.

Sections 3 and 4 describe data and record types. Section 5 shows how the Stream records must be arranged.

3.0 Data Type Description

Table 3-1 lists the possible data types and their values. You can find the type value in the fourth byte of the record.

Table 3-1. Stream Data Types

Data Type	Value
No Data Present	0
Bit Array	1
Two-Byte Signed Integer	2
Four-Byte Signed Integer	3
Four-Byte Real	4 (at present, this data type is not used)
Eight-Byte Real	5
ASCII String	6

The following paragraphs describe the data types listed in *Table 3-1*.

Remember: A word consists of 16 bits, numbered 0 to 15, left to right.

- *Bit Array (1):*

  A bit array is a word which uses the value of a particular bit or group of bits to represent data. A bit array allows one word to represent a number of simple pieces of information.

- *Two-Byte Signed Integer (2):*

  2-byte integer = 1 word 2s-complement representation

  The range of two-byte signed integers is -32,768 to 32,767.

  Following is a representation of a two-byte integer, where S is the sign and M is magnitude.

  ```
  SMMMMMMM MMMMMMMM
  ```

  Following are examples of two-byte integers:

  ```
  00000000 00000001 = 1
  00000000 00000010 = 2
  00000000 10001001 = 137
  11111111 11111111 = -1
  11111111 11111110 = -2
  11111111 01110111 = -137
  ```

- *Four-Byte Signed Integer (3):*

  4-byte integer = 2 word 2s-complement representation

  The range of four-byte signed integers is -2,147,483,648 to 2,147,483,647.

  Following is a representation of a four-byte integer, where S is the sign and M is magnitude.

  ```
  SMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM
  ```

  Examples of four-byte integers:

  ```
  00000000 00000000 00000000 00000001 = 1
  00000000 00000000 00000000 00000010 = 2
  00000000 00000000 00000000 10001001 = 137
  11111111 11111111 11111111 11111111 = -1
  11111111 11111111 11111111 11111110 = -2
  11111111 11111111 11111111 01110111 = -137
  ```

- *Four-Byte Real (4) and Eight-Byte Real (5):*

  4-byte real = 2 word floating point representation

  8-byte real = 4 word floating point representation

  For all non-zero values:

  - A floating point number is made up of three parts: the sign, the exponent, and the mantissa.
  - The value of a floating point number is defined to be:

    (Mantissa) x (16 raised to the true value of the exponent field).
  - The exponent field (bits 1-7) is in Excess-64 representation. The 7-bit field shows a number that is 64 greater than the actual exponent.
  - The mantissa is always a positive fraction >=1/16 and <1. For a 4-byte real, the mantissa is bits 8-31. For an 8-byte real, the mantissa is bits 8-63.
  - The binary point is just to the left of bit 8.
  - Bit 8 represents the value 1/2, bit 9 represents 1/4, etc.
  - In order to keep the mantissa in the range of 1/16 to 1, the results of floating point arithmetic are *normalized*. Normalization is a process whereby the mantissa is shifted left one hex digit at a time until its left FOUR bits represent a non-zero quantity. For every hex digit shifted, the exponent is decreased by one. Since the mantissa is shifted four bits at a time, it is possible for the left three bits of a normalized mantissa to be zero. A zero value, also called *true zero*, is represented by a number with all bits zero.

Following are representations of 4-byte and 8-byte reals, where S is the sign, E is the exponent, and M is the magnitude. Examples of 4-byte reals are included. The representation of the negative values of real numbers is exactly the same as the positive, except that the highest order bit is 1, not 0.

In the eight-byte real representation, the first four bytes are exactly the same as in the four-byte real representation. The last four bytes contain additional binary places for more resolution.

4-byte real:

```
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM
```

8-byte real:

```
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM
MMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM
```

Examples of 4-byte real:

Note: In the first six lines of the following example, the 7-bit exponent field = 65. The actual exponent is 65-64=1.

```
01000001 00010000 00000000 00000000 = 1
01000001 00100000 00000000 00000000 = 2
01000001 00110000 00000000 00000000 = 3
11000001 00010000 00000000 00000000 = -1
11000001 00100000 00000000 00000000 = -2
11000001 00110000 00000000 00000000 = -3

01000000 10000000 00000000 00000000 = .5
01000000 10011001 10011001 10011001 = .6
01000000 10110011 00110011 00110011 = .7
01000001 00011000 00000000 00000000 = 1.5
01000001 00011001 10011001 10011001 = 1.6
01000001 00011011 00110011 00110011 = 1.7
```

```
00000000 00000000 00000000 00000000 = 0
01000001 00010000 00000000 00000000 = 1
01000001 10100000 00000000 00000000 = 10
01000010 01100100 00000000 00000000 = 100
01000011 00111110 10000000 00000000 = 1000
01000100 00100111 00010000 00000000 = 10000
01000101 00011000 01101010 00000000 = 100000
```

- *ASCII String (6):*

  A collection of ASCII characters, where each character is represented by one byte. All odd length strings must be padded with a null character (the number zero) and the byte count for the record containing the ASCII string must include this null character. Stream read-in programs must look for the null character and decrease the length of the string by one if the null character is present.

4.0 Record Types

Records are always an even number of bytes long. If a character string is an odd number of bytes long it is padded with a null character.

Following are records and a brief description of each, where the first two numbers in brackets are the record type and the last two numbers in brackets are the data type. All record numbers are expressed in hexidecimal.

0 HEADER
[0002]

Two-Byte Signed Integer

Contains two bytes of data representing the version number. *Table 4-1* lists corresponding version numbers and GDSII Release numbers. Note that with Release 6.0, the version number changes to three digits.

Table 4-1. GDSII Release and Version numbers

Release Number	Version Number
Prior to 3.0	0
3.0	3
4.0	4
5.0	5
6.0	600 (258 Hex)

1 BGNLIB
[0102]

Two-Byte Signed Integer

Contains last modification time of library (two bytes each for year, month, day, hour, minute, and second) as well as time of last access (same format) and marks beginning of library. Refer to *Figure 4-1.*

Bit #	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 (1 1 1 1 1 1)
Word 1	1C(hex) # of bytes in record
Word 2	01(hex) : 02(hex)
Word 3	year (last modification time)
Word 4	month
Word 5	day
Word 6	hour
Word 7	minute
Word 8	second
Word 9	year (last access time)
Word 10	month
Word 11	day
Word 12	hour
Word 13	minute
Word 14	second

Figure 4-1. A BGNLIB Record

2 LIBNAME
[0206]

ASCII String

Contains a string which is the library name. The library name must adhere to CDOS file name conventions for length and valid characters. The library name may include the file extension (.DB in most cases).

3 UNITS
[0305]

Eight-Byte Real

Contains 2 8-byte real numbers. The first is the size of a database unit in user units. The second is the size of a database unit in meters. For example, if your library was created with the default units (user unit = 1 micron and 1000 database units per user unit), then the first number would be .001 and the second number would be 1E-9. Typically, the first number is less than 1, since you use more than 1 database unit per user unit.

To calculate the size of a user unit in meters, divide the second number by the first.

4 ENDLIB
[0400]

No Data Present

Marks the end of a library.

5 BGNSTR
[0502]

Two-Byte Signed Integer

Contains creation time and last modification time of a structure (in the same format as for the BGNLIB record) and marks the beginning of a structure.

6 STRNAME
[0606]

ASCII String

Contains a string which is the structure name. A structure name may be up to 32 characters long. Legal structure name characters are:

- A through Z
- a through z
- 0 through 9
- Underscore (_)
- Question mark (?)
- Dollar sign ($)

7 ENDSTR
[0700]

No Data Present

Marks the end of a structure.

8 BOUNDARY
[0800]

No Data Present

Marks the beginning of a boundary element.

9 PATH
[0900]

No Data Present

Marks the beginning of a path element.

10 SREF
[0A00]

No Data Present

Marks the beginning of an SREF (structure reference) element.

11 AREF
[0B00]

No Data Present

Marks the beginning of an AREF (array reference) element.

12 TEXT
[0C00]

No Data Present

Marks the beginning of a text element.

13 LAYER
[0D02]

Two-Byte Signed Integer

Contains 2 bytes which specify the layer. The value of the layer must be in the range of 0 to 63.

14 DATATYPE
[0E02]

Two-Byte Signed Integer

Contains 2 bytes which specify datatype. The value of the datatype must be in the range of 0 to 63.

15 WIDTH
[0F03]

Four-Byte Signed Integer

Contains four bytes which specify the width of a path or text lines in data base units. A negative value for width means that the width is absolute, i.e., it is not affected by the magnification factor of any parent reference. If omitted, zero is assumed.

16 XY
[1003]

Four-Byte Signed Integer

Contains an array of XY coordinates in database units. Each X or Y coordinate is four bytes long.

Path and boundary elements may have up to 200 pairs of coordinates. A path must have at least 2, and a boundary at least 4 pairs of coordinates. The first and last point of a boundary must coincide.

A text or SREF element must have only one pair of coordinates.

An AREF has exactly three pairs of coordinates, which specify the orthogonal array lattice. In an AREF the first point is the array reference point. The second point locates a position which is displaced from the reference point by the inter-column spacing times the number of columns. The third point locates a position which is displaced from the reference point by the inter-row spacing times the number of rows.

A node may have from 1 to 50 pairs of coordinates.

A box must have five pairs of coordinates with the first and last points coinciding.

For an example of an array lattice, see *Figure 4-2.*

Figure 4-2. An Array Lattice

17 ENDEL
[1100]

No Data Present

Marks the end of an element.

18 SNAME
[1206]

ASCII String

Contains the name of a referenced structure. See also **STRNAME**.

19 COLROW
[1302]

Two-Byte Signed Integer

Contains 4 bytes. The first 2 bytes contain the number of columns in the array. The third and fourth bytes contain the number of rows. Neither the number of columns nor the number of rows may exceed 32,767 (decimal), and both are positive.

20 TEXTNODE
[1400]

No Data Present

Marks the beginning of a text node. (Not currently used.)

21 NODE
[1500]

No Data Present

Marks the beginning of a node.

22 TEXTTYPE
[1602]

Two-Byte Signed Integer

Contains 2 bytes representing texttype. The value of the texttype must be in the range 0 to 63.

23 PRESENTATION
[1701]

Bit Array

Contains 1 word (2 bytes) of bit flags for text presentation. Bits 10 and 11, taken together as a binary number, specify the font (00 means font 0, 01 means font 1, 10 means font 2, and 11 means font 3). Bits 12 and 13 specify the vertical presentation (00 means top, 01 means middle, and 10 means bottom). Bits 14 and 15 specify the horizontal presentation (00 means left, 01 means center, and 10 means right). Bits 0 through 9 are reserved for future use and must be cleared. If this record

is omitted, then top-left justification and font 0 are assumed.

Figure 4-3 shows a presentation record.

Figure 4-3. A PRESENTATION Record

24 SPACING Discontinued

25 STRING [1906] ASCII String

Contains a character string for text presentation, up to 512 characters long.

26 STRANS
[1A01]

Bit Array

Contains two bytes of bit flags for SREF, AREF, and text transformation. Bit 0 (the leftmost bit) specifies reflection. If it is set, then reflection about the X-axis is applied before angular rotation. For AREFs, the entire array lattice is reflected, with the individual array elements rigidly attached. Bit 13 flags absolute magnification. Bit 14 flags absolute angle. Bit 15 (the rightmost bit) and all remaining bits are reserved for future use and must be cleared. If this record is omitted, then the element is assumed to have no reflection and its magnification and angle are assumed to be non-absolute.

Figure 4-4 shows an STRANS record.

Figure 4-4. An STRANS Record

27 MAG
[1B05]

Eight-Byte Real

Contains a double-precision real number (8 bytes) which is the magnification factor. If omitted, a magnification of 1 is assumed.

28 ANGLE
[1C05]

Eight-Byte Real

Contains a double-precision real number (8 bytes) which is the angular rotation factor, measured in degrees and in counterclockwise direction.

For an AREF, the ANGLE rotates the entire array lattice (with the individual array elements rigidly attached) about the array reference point. If this record is omitted, an angle of zero degrees is assumed.

29 UINTEGER

User Integer (No longer used)

User Integer data was used in Release 2.0 only. If any Stream format files from Release 2.0 are read in to the current software, the Stream format input program INFORM converts the User Integer data to property data having attribute number 126. See also PROPATTR and PROPVALUE.

30 USTRING

Character String (No longer used)

User String data, formerly called character string data (CSD), was used in Releases 1.0 and 2.0. If any Stream format files from these releases are read in to the current software, the Stream format input program INFORM converts the User String data to property data having attribute number 127. If this

record is not present then it is the null string. See also PROPATTR and PROPVALUE.

31 REFLIBS
[1F06]

ASCII String

Contains the names of the reference libraries. This record must be present if there are any reference libraries bound to the current library.

The name for the first reference library starts at byte 0 and the name of the second library starts at byte 45 (decimal). The reference library names may include directory specifiers (separated with ":") and an extension (separated with "."). If either library is not named, its place is filled with nulls.

32 FONTS
[2006]

ASCII String

Contains names of textfont definition files. This record must be present if any of the 4 fonts have a corresponding textfont definition file. This record must not be present if none of the fonts have a textfont definition file. The name of font 0 starts the record, followed by the remaining 3 fonts. Each name is 44 bytes long and is null if there is no corresponding textfont definition. Each name is padded with nulls if it is shorter than 44 bytes. The textfont definition file names may include directory specifiers (separated with ":") and an extension (separated with ".").

33 PATHTYPE
[2102]

Two-Byte Signed Integer

This record contains a value of 0 for square-ended paths that end flush with their endpoints, 1 for round-ended paths, and 2 for square-ended paths

that extend a half-width beyond their endpoints. Pathtype 4 (for the CustomPlus product only) signifies a path with variable square-end extensions (see records 48 and 49). If not specified, a Pathtype of 0 is assumed. *Figure 4-5* shows the pathtypes.

Figure 4-5. Pathtypes

34 GENERATIONS [2202]

Two-Byte Signed Integer

This record contains a positive count of the number of copies of deleted or backed-up structures to retain. This number must be at least 2 and not more than 99. If the GENERATIONS record is not present, a value of 3 is assumed.

35 ATTRTABLE [2306]

ASCII String

Contains the name of the attribute definition file. This record is present only if there is an attribute definition file bound to the library. The attribute definition file name may include directory specifiers (separated with ":") and an extension (separated with "."). Maximum size is 44 bytes.

36 STYPTABLE [2406]

ASCII String (Unreleased feature)

37 STRTYPE [2502]

Two-Byte Signed Integer (Unreleased feature)

38 ELFLAGS [2601]

Bit Array

Contains 2 bytes of bit flags. Bit 15 (the rightmost bit) specifies Template data. Bit 14 specifies External data (also referred to as Exterior data). All other bits are currently unused and must be cleared to 0. If this record is omitted, then all bits are assumed to be 0.

For additional information on Template data, consult the **GDSII Reference Manual**. For additional information on External data, consult the **CustomPlus User's Manual**. *Figure 4-6* shows an ELFLAGS record.

Figure 4-6. An ELFLAGS Record

39 ELKEY
[2703]

Four-Byte Signed Integer (Unreleased feature)

40 LINKTYPE
[28]

Two-Byte Signed Integer (Unreleased feature)

41 LINKKEYS
[29]

Four-Byte Signed Integer (Unreleased feature)

42 NODETYPE
[2A02]

Two-Byte Signed Integer

Contains 2 bytes which specify nodetype. The value of the nodetype must be in the range of 0 to 63.

43 PROPATTR
[2B02]

Two-Byte Signed Integer

Contains 2 bytes which specify the attribute number. The attribute number is an integer from 1 to 127. Attribute numbers 126 and 127 are reserved for the user integer and user string (CSD) properties, which existed prior to Release 3.0. (User string and user integer data from previous releases is converted to property data having attribute number

127 and 126 by the Stream format input program INFORM.)

44 PROPVALUE
[2C06]

ASCII String

Contains the string value associated with the attribute named in the preceding PROPATTR record. Maximum length is 126 characters. The attribute-value pairs associated with any one element must all have distinct attribute numbers. Also, there is a limit on the total amount of property data that may be associated with any one element: the total length of all the strings, plus twice the number of attribute-value pairs, must not exceed 128 (or 512 if the element is an SREF, AREF, or node).

For example, if a boundary element used property attribute 2 with property value "metal", and property attribute 10 with property value "property", then the total amount of property data would be 18 bytes. This is 6 bytes for "metal" (odd-length strings must be padded with a null) + 8 for "property" + 2 times the 2 attributes (4) = 18.

45 BOX
[2D00]

No Data Present

Marks the beginning of a box element.

46 BOXTYPE
[2E02]

Two-Byte Signed Integer

Contains 2 bytes which specify boxtype. The value of the boxtype must be in the range of 0 to 63.

47 PLEX
[2F03]

Four-Byte Signed Integer

A unique positive number which is common to all elements of the plex to which this element belongs. The head of the plex is flagged by setting the seventh bit; therefore, plex numbers should be small enough to occupy only the rightmost 24 bits. If this record is omitted, then the element is not a plex member.

48 BGNEXTN
[3003]

Four-Byte Signed Integer. (This record type only occurs in CustomPlus.)

Applies to Pathtype 4. Contains four bytes which specify in database units the extension of a path outline beyond the first point of the path. Value can be negative.

49 ENDEXTN
[3103]

Four-Byte Signed Integer. (This record type only occurs in CustomPlus.)

Applies to Pathtype 4. Contains four bytes which specify in database units the extension of a path outline beyond the last point of the path. Value can be negative.

50 TAPENUM
[3202]

Two-Byte Signed Integer

Contains two bytes which specify the number of the current reel of tape for a multi-reel Stream file. For the first tape, the TAPENUM is 1; for the second tape, the TAPENUM is 2; etc.

51 TAPECODE
[3302]

Two-Byte Signed Integer

Contains 12 bytes. This is a unique 6-integer code which is common to all the reels of a multi-reel Stream file. It verifies that the correct reels are being read in.

52 STRCLASS
[3401]

Two-Byte Bit Array. (Only for Calma internal use with CustomPlus structures.)

If Stream tapes are produced by non-Calma programs, then this record should either be omitted or cleared to zero.

53 RESERVED
[3503]

Four-Byte Signed Integer. (Reserved for future use.)

This record type was used for NUMTYPES but was not required.

54 FORMAT
[3602]

Two-Byte Signed Integer. (Optional)

Defines the format type of a Stream tape in two bytes. The two possible values are: 0 for Archive format, 1 for Filtered format.

An Archive Stream file contains elements for all the layers and data types. It is created with OUTFORM. In an Archive Stream file, the FORMAT record is followed immediately by the UNITS record. A file which does not have the FORMAT record is assumed to be an Archive file.

A Filtered Stream file contains only the elements on the layers and with the data types specified by the user during execution of STREAMOUT. The list of layers and data types specified for STREAMOUT

follows the FORMAT record in MASK records. The MASK records are terminated with the ENDMASKS record. At least one MASK record must immediately follow the FORMAT record. The Filtered Stream file is created with STREAMOUT.

See MASK and ENDMASKS below.

55 MASK [3706]

ASCII String. (Required for Filtered format, and present only in Filtered Stream file.)

Contains the list of layers and data types specified by the user for STREAMOUT. At least one MASK record must follow the FORMAT record. More than one MASK record may follow the FORMAT record. The last MASK record is followed by the ENDMASKS record.

See FORMAT above and ENDMASKS below.

In the MASK list, data types are separated from the layers with a semi-colon. Individual layers or data types are separated with a space. A range of layers or data types is specified with a dash. An example MASK list looks like this:

```
1 5-7 10 ; 0-63
```

56 ENDMASKS [3800]

No Data Present. (Required for Filtered format, and present only in Filtered Stream file.)

Terminates the MASK records. The ENDMASKS record must follow the last MASK record. ENDMASKS is immediately followed by the UNITS record.

See FORMAT and MASK above.

57 LIBDIRSIZE
[3902]

Two-Byte Signed Integer

Contains the number of pages in the Library directory. This information is used only if INFORM is reading the data into a new library. If this record is present, it should occur between the BGNLIB record and the LIBNAME record.

58 SRFNAME
[3A06]

ASCII String

Contains the name of the Sticks Rules File, if one is bound to the library. This information is used only if INFORM is reading the data into a new library. If this record is present, it should occur between the BGNLIB record and the LIBNAME record.

59 LIBSECUR
[3B02]

Two-Byte Signed Integer

Contains an array of Access Control List (ACL) data. There may be from 1 to 32 ACL entries, each consisting of:

- A group number
- A user number
- Access rights

This information is used only if INFORM is reading the data into a new library. If this record is present, it should occur between the BGNLIB record and the LIBNAME record.

5.0 Stream Syntax

Following is a Bachus Naur representation of the Stream syntax. An element shown in `ALL CAPS` is the name of an actual record type. An element shown in `lower case` means that name can be further broken down into a set of actual record types. *Table 5-1* shows the meaning of the different symbols used.

Table 5-1. Bachus Naur Symbols

Symbol Name	Symbol	Meaning
Double Colon	::	"Is composed of"
Square brackets	[]	An element which can occur zero or one time.
Braces	{}	Choose one of the elements within the braces.
Braces with an asterisk	{}*	The elements within the braces can occur zero or more times.
Braces with a plus	{}+	The elements within braces must occur one or more times.
Angle brackets	<>	These elements are further defined in the Stream syntax list.
Vertical bar	|	"Or"

```
<stream format>::=      HEADER BGNLIB [LIBDIRSIZE] [SRFNAME]
                        [LIBSECUR] LIBNAME [REFLIBS] [FONTS]
                        [ATTRTABLE] [GENERATIONS] [<FormatType>]
                        UNITS {<structure>}* ENDLIB

<FormatType>::=         FORMAT | FORMAT {MASK}+ ENDMASKS

<structure>::=          BGNSTR STRNAME [STRCLASS] {<element>}*
                        ENDSTR

<element>::=            {<boundary> | <path> | <SREF> | <AREF>
                        | <text> | <node> | <box>} {<property>}*
                        ENDEL

<boundary>::=           BOUNDARY [ELFLAGS] [PLEX] LAYER DATATYPE XY

<path>::=               PATH [ELFLAGS] [PLEX] LAYER DATATYPE
                        [PATHTYPE] [WIDTH] [BGNEXTN] [ENDEXTN] XY

<SREF>::=               SREF [ELFLAGS] [PLEX] SNAME [<strans>] XY

<AREF>::=               AREF [ELFLAGS] [PLEX] SNAME [<strans>]
                        COLROW XY

<text>::=               TEXT [ELFLAGS] [PLEX] LAYER <textbody>

<node>::=               NODE [ELFLAGS] [PLEX] LAYER NODETYPE XY

<box>::=                BOX [ELFLAGS] [PLEX] LAYER BOXTYPE XY

<textbody>::=           TEXTTYPE [PRESENTATION] [PATHTYPE] [WIDTH]
                        [<strans>]XY STRING

<strans>::=             STRANS [MAG] [ANGLE]

<property>::=           PROPATTR PROPVALUE
```

6.0 Multi-Reel Stream Format

You can put Stream format onto multiple reels of tape. The first tape must end with the records TAPENUM, TAPECODE, and LIBNAME, in that order. Each subsequent tape must begin with the same records, in the same order, and must end with the record TAPENUM. Stream tapes must contain only complete Stream records, i.e., no Stream record should begin on one tape and continue on the next tape.

Note: Use TAPENUM and TAPECODE only as described. These records cannot appear anywhere else in the Stream file.

The records TAPENUM, TAPECODE, and LIBNAME, used in this manner, are used only for identification of the tapes and are not incorporated into the library in any way. LIBNAME occurs normally as the third record of a Stream file. Tapes may end after any record in Stream format.

Following are illustrations of multi-reel Stream tapes.

Tape 1:

HEADER	Several Complete Stream records	TAPENUM (1)	TAPECODE <code>	LIBNAME <library name>

Intermediate Tape (i):

TAPENUM (i)	TAPECODE <code>	LIBNAME <library name>	More Complete Stream records	TAPENUM (i)

Last Tape (n):

TAPENUM (n)	TAPECODE <code>	LIBNAME <library name>	More complete Stream records	ENDLIB

Following is a Bachus Naur representation of multi-reel Stream tapes. Refer to *Table 5-1* for an explanation of the symbols used.

```
<multi-reel Stream tape>::=   <tape1> {<continuation tapes>}+

<tape1>::=                    HEADER {<complete records in Stream
                              syntax>}* <tape-id>

<continuation tapes>::=       <tape-id> {<complete records continuing
                              in Stream syntax>}+ TAPENUM

<tape-id>::=                  TAPENUM TAPECODE LIBNAME
```

The entire concatenation of Stream records, without the tape-id groups and **TAPENUM**s, should conform to the Stream syntax described in **Section 5.0**.

7.0 Example of a Stream Format File

Figure 7-1 shows an FPRINT of a Stream format file. An explanation follows the example.

```
? FPRINT
Source File Name: EXAMPLE.SF
Format (Octal): HEX
Output File: $TTO
000 0006 0002 0258 001C 0102 0055 0009 0003 ...........U....
008 0000 0000 0000 0055 0009 0003 000A 0010 .......U........
010 0000 0006 3902 0028 000A 3B02 0003 0005 ....9..(..;.....
018 0007 000E 0206 4558 414D 504C 452E 4442 ......EXAMPLE.DB
020 005C 1F06 4744 5349 493A 5245 4631 2E44 ...GDSII:REF1.D
028 4200 0000 0000 0000 0000 0000 0000 0000 B...............
030 0000 0000 0000 0000 0000 0000 0000 0000 ................
****
048 0000 0000 0000 0000 0000 0000 00B4 2006 ............... .
050 4744 5349 493A 4341 4C4D 4146 4F4E 542E GDSII:CALMAFONT.
058 5458 0000 0000 0000 0000 0000 0000 0000 TX..............
060 0000 0000 0000 0000 0000 0000 4744 5349 ............GDSI
068 493A 5445 5854 2E54 5800 0000 0000 0000 I:TEXT.TX.......
070 0000 0000 0000 0000 0000 0000 0000 0000 ................
078 0000 0000 0000 0000 4744 5349 493A 464F ........GDSII:FO
080 4E54 2E54 5800 0000 0000 0000 0000 0000 NT.TX...........
088 0000 0000 0000 0000 0000 0000 0000 0000 ................
090 0000 0000 4744 5349 493A 5047 464F 4E54 ....GDSII:PGFONT
098 2E54 5800 0000 0000 0000 0000 0000 0000 .TX.............
0A0 0000 0000 0000 0000 0000 0000 0000 0000 ................
0A8 0012 2306 4744 5349 493A 4154 5452 532E ..#.GDSII:ATTRS.
```

Figure 7-1. Sample Stream Format File (Page 1 of 2)

```
0B0 4154 0006 2202 0003 0014 0305 3E41 8937 AT.."......>A.7
0B8 4BC6 A7EF 3944 B82F A09B 5A51 001C 0502 K...9D./..ZQ....
0C0 0055 0007 000C 0011 001D 000A 0055 0007 .U..........U..
0C8 0011 0011 003A 0014 000C 0606 4558 414D .....:......EXAM
0D0 504C 4532 0004 0B00 000C 1206 4558 414D PLE2........EXAM
0D8 504C 4531 0006 1A01 8000 000C 1C05 425A PLE1..........BZ
0E0 0000 0000 0000 0008 1302 0002 0002 001C ................
0E8 1003 0000 4E20 0000 4E20 0000 4E20 0001 ....N ..N ..N ..
0F0 4FF0 0001 3880 0000 4E20 0004 1100 0004 O...8...N ......
0F8 0700 001C 0502 0055 0007 000C 000B 001C .......U........
100 0009 0055 0008 001C 000F 0039 003A 000C ...U.......9.:..
108 0606 4558 414D 504C 4531 0004 0C00 0006 ..EXAMPLE1......
110 0D02 0000 0006 1602 0000 0006 1701 0005 ................
118 0006 1A01 8006 000C 1B05 4120 0000 0000 ..........A ....
120 0000 000C 1003 0000 4E20 0000 4E20 000E ........N ..N ..
128 1906 4920 414D 2048 4552 450D 0004 1100 ..I AM HERE.....
130 0004 0800 0006 2601 0001 0006 0D02 0002 ......&.........
138 0006 0E02 0003 0024 1003 0000 1388 0000 .......$........
140 6D60 0000 2EE0 0000 6D60 0000 1F40 0000 m'......m'...@..
148 84D0 0000 1388 0000 6D60 0004 1100 0004 ........m'......
150 0900 0006 0D02 0004 0006 0E02 003F 0006 .............?..
158 2102 0001 0008 0F03 0000 03E8 0024 1003 !............$..
160 0000 3A98 0000 36B0 0000 6590 0000 36B0 ..:...6...e...6.
168 0000 84D0 0000 2328 0000 55F0 0000 1770 ......#(..U....p
170 0006 2B02 0002 000A 2C06 4D45 5441 4C00 ..+......,.METAL.
178 0006 2B02 000A 000C 2C06 5052 4F50 4552 ..+......,.PROPER
180 5459 0004 1100 0004 0700 0004 0400 TY............
```

Figure 7-1. Sample Stream Format File (Page 2 of 2)

The database that produced this stream format output had only two structures. They were called EXAMPLE1 and EXAMPLE2. EXAMPLE2 contained only one element, a 2 x 2 AREF of EXAMPLE1. EXAMPLE1 contained a boundary that was template data, a path with two properties, and a middle-center justified text element containing the string `I AM HERE`.

The records contained in this stream file are as follows:

0006 0002 0258

The first word says that this record is 6 bytes long. The second word says that this is the HEADER record (00 hex), and that it contains data of type 2-byte signed integer (02 hex). The information in the third word is the GDSII version number, which in this case is version 600 (258 hex).

001C 0102 0055 0009 0003 0000 0000 0000 0055 0009 0003 000A 0010 0000

This record is 28 (1C hex) bytes long. It is the BGNLIB (01 hex) record and it contains data of type 2-byte signed integer (02). The 24 bytes of information following the first two header words contain the modification and last access date and time. The last 6 words of information, for example, contain: the year - 85 (55 hex), the month - September (9 hex), the day - 3 (3 hex), the hour - 10 a.m. (A hex), the minute - 16 (10 hex) and the seconds - 0. All together this record says that this library was last modified on September 3, 1985 at 10:16:00 a.m.

0006 3902 0028

This record is 6 bytes long. It is the LIBDIRSIZE (39 hex) record, and it contains data of type 2-byte signed integer (02). In this example, the directory size is 40 (28 hex) pages.

000A 3B02 0003 0005 0007

This record is 10 (A hex) bytes long. It is the LIBSECUR (3B hex) record and it contains data of type 2-byte signed integer (02). In this example, there is only 1 ACL entry. The entry has a group number of 3, a user number of 5, and access rights of 7. This means that the only one with any access rights to this library is user number 5 in group number 3. The access code (0007) means this user has read and write access and is also the owner of the library.

000E 0206 4558 414D 504C 452E 4442

This record is 14 (E hex) bytes long. It is the LIBNAME (02 hex) record and it contains data of type ASCII string (06). The 5 words of information contain the library name EXAMPLE.DB.

```
005C 1F06 4744 5349 493A 5245 4631 2E44 4200 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000
```

This record is 92 (5C hex) bytes long. It is the REFLIBS (1F hex) record and it contains data of type ASCII string (06). All 92 bytes of this record must be present if there are any reference libraries bound to the working library. In this example, the library GDSII:REF1.DB is the bound reference library. The library name is padded with nulls to equal 44 bytes. There is no second reference library, so the last 44 bytes are filled with nulls.

```
00B4 2006 4744 5349 493A 4341 4C4D 4146 4F4E 542E 5458 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 4744 5349
493A 5445 5854 2E54 5800 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 4744 5349 493A 464F 4E54 2E54
5800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 4744 5349 493A 5047 464F 4E54 2E54 5800 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

This record is 180 (B4 hex) bytes long. It is the FONTS (20 hex) record and it contains data of type ASCII string (06). All 180 bytes of this record must be present if there are any textfont definition files bound to this library. In this example, there are four (the maximum possible) textfont definition files bound to this library. They are GDSII:CALMAFONT.TX, GDSII:TEXT.TX, GDSII:FONT.TX, and GDSII:PGFONT.TX. Each string is padded with nulls out to 44 bytes.

```
0012 2306 4744 5349 493A 4154 5452 532E 4154
```

This record is 18 (12 hex) bytes long. It is the ATTRTABLE (23 hex) record and it contains data of type ASCII string (06). This record is only present if an attribute table is bound to the library. The name of the attribute table is GDSII:ATTRS.AT.

```
0006 2202 0003
```

This record is 6 bytes long. It is the GENERATIONS (22 hex) record and it contains data of type 2-byte signed integer (02). In this example, 3 generations of a structure are retained in the library.

```
0014 0305 3E41 8937 4BC6 A7EF 3944 B82F A09B 5A51
```

This record is 20 (14 hex) bytes long. It is the UNITS (03 hex) record and it contains data of type 8-byte real (05). In this example, 3E41 8937 4BC6 A7EF is 1E-3. This implies that a database unit is 1 thousandth of a user unit. The record 3944 B82F A09B 5A51 is 1E-9. This implies that a database unit is 1E-9 meters (1E-3 microns).

```
001C 0502 0055 0007 000C 0011 001D 000A 0055 0007 0011 0011 003A
0014
```

This record is 28 (1C hex) bytes long. It is the BGNSTR (05 hex) record and it contains data of type 2-byte signed integer (02). The information in this record is the creation time and last modification time of the structure and is in the same format as in the BGNLIB record. This structure was created July 12, 1985 at 5:29:10 p.m. and last modified July 12, 1985 at 5:48:20 p.m.

```
000C 0606 4558 414D 504C 4532
```

This record is 12 (C hex) bytes long. It is the STRNAME (06 hex) record and it contains data of type ASCII string (06). The structure name is EXAMPLE2.

```
0004 0B00
```

This record is 4 bytes long. It is the AREF (0B hex) record and it contains no data (00). It marks the start of an AREF.

```
000C 1206 4558 414D 504C 4531
```

This record is 12 (C hex) bytes long. It is the SNAME (12 hex) record and it contains data of type ASCII string (06). This element contains an SNAME of structure EXAMPLE1.

```
0006 1A01 8000
```

This record is 6 bytes long. It is the STRANS (1A hex) record and it contains bit array data (01). In this example, only bit 0 is set, which implies that this AREF is reflected. Since bits 13 and 14 are not set, this structure's magnification and angle, respectively, are not absolute.

000C 1C05 425A 0000 0000 0000

This record is 12 (C hex) bytes long. It is the ANGLE (1C hex) record and it contains 8-byte real data (05). The data 425A 0000 0000 0000 represents 90.0, which implies that this AREF was placed at an angle of 90 degrees.

0008 1302 0002 0002

This record is 8 bytes long. It is the COLROW (13 hex) record and it contains 2-byte signed integer data (02). In this example, we have a 2 x 2 AREF.

001C 1003 0000 4E20 0000 4E20 0000 4E20 0001 4FF0 0001 3880 0000 4E20

This record is 28 (1C hex) bytes long. It is the XY (10 hex) record and it contains data of type 4-byte signed integer (03). The data, taken 2 words at a time, can be translated to decimal as: 20000, 20000, 20000, 86000, 80000, 20000. Multiply these numbers by 1 thousandth (because a data base unit is 1 thousandth of a user unit) and we get the coordinates: (20, 20), (20, 86), and (80, 20). The first coordinate is the array reference point. The second coordinate is a point which is displaced from the array reference point in the Y-direction by the number of columns times the inter-column spacing. In this example, the second point was displaced 66 (86 - 20) units from the array reference point. Since there are 2 columns, this implies that the inter-column spacing was 33 units. A similar calculation could be carried out to verify that the inter-row spacing was 30 units.

0004 1100

This record is 4 bytes long. It is the ENDEL (11 hex) record and it contains no data (00). It marks the end of an element.

0004 0700

This record is 4 bytes long. It is the ENDSTR (07 hex) record and it contains no data (00). It marks the end of a structure.

001C 0502 0055 0007 000C 000B 001C 0009 0055 0008 001C 000F 0039 003A

This is another BGNSTR record. This structure was created July 12, 1985 at 11:28:09 a.m. and last modified August 28, 1985 at 3:57:58 p.m.

000C 0606 4558 414D 504C 4531

This is another STRNAME record. It contains the string EXAMPLE1.

0004 0C00

This record is 4 bytes long. It is the TEXT (0C hex) record and it contains no data (00). It marks the start of a text element.

0006 0D02 0000

This record is 6 bytes long. It is the LAYER (0D hex) record and it contains data of type 2-byte signed integer (02). This text element is on layer 0.

0006 1602 0000

This record is 6 bytes long. It is the TEXTTYPE (16 hex) record and it contains data of type 2-byte signed integer (02). This text element is of text type 0.

0006 1701 0005

This record is 6 bytes long. It is the PRESENTATION (17 hex) record and it contains bit array data (01). The hex number 0005 in binary has all bits set to zero except bits 13 and 15. Since bits 10 and 11 are 00, the text element used font 0. Since bits 12 and 13 are 01, the text has a middle vertical presentation. And since bits 14 and 15 are 01, the text has a center horizontal presentation.

0006 1A01 8006

This is another STRANS record. This text is reflected and has an absolute magnification and absolute angle.

000C 1B05 4120 0000 0000 0000

This record is 12 (C hex) bytes long. It is the MAG (1B hex) record and it contains data of type 8-byte real (05). The data in this record represents 2.0, therefore, this text is magnified 2 times.

000C 1003 0000 4E20 0000 4E20

This is another XY record. The text is placed at coordinate (20, 20).

000E 1906 4920 414D 2048 4552 450D

This record is 14 (E hex) bytes long. It is the STRING (19 hex) record and it contains data of type ASCII string (06). The text string is I AM HERE.

0004 1100

This is another ENDEL record.

0004 0800

This record is 4 bytes long. It is the BOUNDARY (08 hex) record and it contains no data (00). It marks the start of a boundary element.

0006 2601 0001

This record is 6 bytes long. It is the ELFLAGS (17 hex) record and it contains bit array data (01). Since bit 15 is set, this element is template data. However, since bit 14 is not set, it is not external data.

0006 0D02 0002

This is another LAYER record. The boundary is on layer 2.

0006 0E02 0003

This record is 6 bytes long. It is the DATATYPE (0E hex) record and it contains data of type 2-byte signed integer (02). This boundary is of data type 3.

0024 1003 0000 1388 0000 6D60 0000 2EE0 0000 6D60 0000 1F40 0000 84D0 0000 1388 0000 6D60

This is another XY record. The coordinates are (5, 28), (12, 28), (8, 34), 5(5, 28).

0004 1100

This is another ENDEL record.

0004 0900

This record is 4 bytes long. It is the PATH (09 hex) record and it contains no data (00). It marks the start of a path element.

0006 0D02 0004

This is another LAYER record. The path is on layer 4.

0006 0E02 003F

This is another DATATYPE record. The path is data type 63 (3F hex).

0006 2102 0001

This record is 6 bytes long. It is the PATHTYPE (21 hex) record and it contains data of type 2-byte signed integer (02). This path is of path type 1.

0008 0F03 0000 03E8

This record is 8 bytes long. It is the WIDTH (0F hex) record and it contains data of type 4-byte signed integer (03). The number 03E8 hex is 1000 in decimal. Multiply this by 1 thousandth (because a data base unit is 1 thousandth of a user unit) and the result is 1. Therefore, the width of this path is 1.

0024 1003 0000 3A98 0000 36B0 0000 6590 0000 36B0 0000 84D0 0000 2328 0000 55F0 0000 1770

This is another XY record. This path's coordinates are (15, 14), (26, 14), (34, 9), (22, 6).

0006 2B02 0002

This record is 6 bytes long. It is the PROPATTR (2B hex) record and it contains data of type 2-byte signed integer (02). This path has a property with attribute number 2.

000A 2C06 4D45 5441 4C00

This record is 10 (A hex) bytes long. It is the PROPVALUE (2C hex) record and it contains data of type ASCII string (06). The property value for the property attribute described in the PROPATTR record is METAL. Note that this odd length string is padded with a null.

0006 2B02 000A

This is another PROPATTR record. This path has another property associated with it and it has attribute number 10 (A hex).

000C 2C06 5052 4F50 4552 5459

This is another PROPVALUE record. Property attribute 10 (above) has the property PROPERTY.

0004 1100

This is another ENDEL record.

0004 0700

This is another ENDSTR record.

0004 0400

This record is 4 bytes long. This record is the ENDLIB (04 hex) record and it contains no data (00). ENDLIB marks the end of a stream format file.

UM10204

I^{2}C-bus specification and user manual

Rev. 6 — 4 April 2014 **User manual**

Document information

Info	Content
Keywords	I2C, I2C-bus, Standard-mode, Fast-mode, Fast-mode Plus, Fm+, Ultra Fast-mode, UFm, High Speed, Hs, inter-IC, SDA, SCL, USDA, USCL
Abstract	Philips Semiconductors (now NXP Semiconductors) developed a simple bidirectional 2-wire bus for efficient inter-IC control. This bus is called the Inter-IC or I^{2}C-bus. Only two bus lines are required: a serial data line (SDA) and a serial clock line (SCL). Serial, 8-bit oriented, bidirectional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in the Fast-mode Plus (Fm+), or up to 3.4 Mbit/s in the High-speed mode. The Ultra Fast-mode is a uni-directional mode with data transfers of up to 5 Mbit/s.

Revision history

Rev	Date	Description
v.6	20140404	User manual; sixth release
Modifications:		• Figure 41 "$R_{p(max)}$ as a function of bus capacitance" updated (recalculated) • Figure 42 "$R_{p(min)}$ as a function of V_{DD}" updated (recalculated)
v.5	20121009	User manual; fifth release
v.4	20120213	User manual Rev. 4
v.3	20070619	Many of today's applications require longer buses and/or faster speeds. Fast-mode Plus was introduced to meet this need by increasing drive strength by as much as 10× and increasing the data rate to 1 Mbit/s while maintaining downward compatibility to Fast-mode and Standard-mode speeds and software commands.
v2.1	2000	Version 2.1 of the I²C-bus specification
v2.0	1998	The I²C-bus has become a de facto world standard that is now implemented in over 1000 different ICs and licensed to more than 50 companies. Many of today's applications, however, require higher bus speeds and lower supply voltages. This updated version of the I²C-bus specification meets those requirements.
v1.0	1992	Version 1.0 of the I²C-bus specification
Original	1982	first release

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

All information provided in this document is subject to legal disclaimers.
© NXP Semiconductors N.V. 2014. All rights reserved.

1. Introduction

The I²C-bus is a de facto world standard that is now implemented in over 1000 different ICs manufactured by more than 50 companies. Additionally, the versatile I²C-bus is used in various control architectures such as System Management Bus (SMBus), Power Management Bus (PMBus), Intelligent Platform Management Interface (IPMI), Display Data Channel (DDC) and Advanced Telecom Computing Architecture (ATCA).

This document assists device and system designers to understand how the I²C-bus works and implement a working application. Various operating modes are described. It contains a comprehensive introduction to the I²C-bus data transfer, handshaking and bus arbitration schemes. Detailed sections cover the timing and electrical specifications for the I²C-bus in each of its operating modes.

Designers of I²C-compatible chips should use this document as a reference and ensure that new devices meet all limits specified in this document. Designers of systems that include I²C devices should review this document and also refer to individual component data sheets.

2. I²C-bus features

In consumer electronics, telecommunications and industrial electronics, there are often many similarities between seemingly unrelated designs. For example, nearly every system includes:

- Some intelligent control, usually a single-chip microcontroller
- General-purpose circuits like LCD and LED drivers, remote I/O ports, RAM, EEPROM, real-time clocks or A/D and D/A converters
- Application-oriented circuits such as digital tuning and signal processing circuits for radio and video systems, temperature sensors, and smart cards

To exploit these similarities to the benefit of both systems designers and equipment manufacturers, as well as to maximize hardware efficiency and circuit simplicity, Philips Semiconductors (now NXP Semiconductors) developed a simple bidirectional 2-wire bus for efficient inter-IC control. This bus is called the Inter IC or I²C-bus. All I²C-bus compatible devices incorporate an on-chip interface which allows them to communicate directly with each other via the I²C-bus. This design concept solves the many interfacing problems encountered when designing digital control circuits.

Here are some of the features of the I²C-bus:

- Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCL).
- Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times; masters can operate as master-transmitters or as master-receivers.
- It is a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer.
- Serial, 8-bit oriented, bidirectional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in Fast-mode Plus, or up to 3.4 Mbit/s in the High-speed mode.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

- Serial, 8-bit oriented, unidirectional data transfers up to 5 Mbit/s in Ultra Fast-mode
- On-chip filtering rejects spikes on the bus data line to preserve data integrity.
- The number of ICs that can be connected to the same bus is limited only by a maximum bus capacitance. More capacitance may be allowed under some conditions. Refer to Section 7.2.

Figure 1 shows an example of I²C-bus applications.

Fig 1. Example of I²C-bus applications

2.1 Designer benefits

I²C-bus compatible ICs allow a system design to progress rapidly directly from a functional block diagram to a prototype. Moreover, since they ‘clip’ directly onto the I²C-bus without any additional external interfacing, they allow a prototype system to be modified or upgraded simply by ‘clipping’ or ‘unclipping’ ICs to or from the bus.

Here are some of the features of I²C-bus compatible ICs that are particularly attractive to designers:

- Functional blocks on the block diagram correspond with the actual ICs; designs proceed rapidly from block diagram to final schematic.
- No need to design bus interfaces because the I²C-bus interface is already integrated on-chip.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

- Integrated addressing and data-transfer protocol allow systems to be completely software-defined.
- The same IC types can often be used in many different applications.
- Design-time reduces as designers quickly become familiar with the frequently used functional blocks represented by I²C-bus compatible ICs.
- ICs can be added to or removed from a system without affecting any other circuits on the bus.
- Fault diagnosis and debugging are simple; malfunctions can be immediately traced.
- Software development time can be reduced by assembling a library of reusable software modules.

In addition to these advantages, the CMOS ICs in the I²C-bus compatible range offer designers special features which are particularly attractive for portable equipment and battery-backed systems.

They all have:

- Extremely low current consumption
- High noise immunity
- Wide supply voltage range
- Wide operating temperature range.

2.2 Manufacturer benefits

I²C-bus compatible ICs not only assist designers, they also give a wide range of benefits to equipment manufacturers because:

- The simple 2-wire serial I²C-bus minimizes interconnections so ICs have fewer pins and there are not so many PCB tracks; result — smaller and less expensive PCBs.
- The completely integrated I²C-bus protocol eliminates the need for address decoders and other ‘glue logic’.
- The multi-master capability of the I²C-bus allows rapid testing and alignment of end-user equipment via external connections to an assembly line.
- The availability of I²C-bus compatible ICs in various leadless packages reduces space requirements even more.

These are just some of the benefits. In addition, I²C-bus compatible ICs increase system design flexibility by allowing simple construction of equipment variants and easy upgrading to keep designs up-to-date. In this way, an entire family of equipment can be developed around a basic model. Upgrades for new equipment, or enhanced-feature models (that is, extended memory, remote control, etc.) can then be produced simply by clipping the appropriate ICs onto the bus. If a larger ROM is needed, it is simply a matter of selecting a microcontroller with a larger ROM from our comprehensive range. As new ICs supersede older ones, it is easy to add new features to equipment or to increase its performance by simply unclipping the outdated IC from the bus and clipping on its successor.

All information provided in this document is subject to legal disclaimers.
© NXP Semiconductors N.V. 2014. All rights reserved.

2.3 IC designer benefits

Designers of microcontrollers are frequently under pressure to conserve output pins. The I²C protocol allows connection of a wide variety of peripherals without the need for separate addressing or chip enable signals. Additionally, a microcontroller that includes an I²C interface is more successful in the marketplace due to the wide variety of existing peripheral devices available.

3. The I²C-bus protocol

3.1 Standard-mode, Fast-mode and Fast-mode Plus I²C-bus protocols

Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. Each device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device. An LCD driver may be only a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers (see Table 1). A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

Table 1. Definition of I²C-bus terminology

Term	Description
Transmitter	the device which sends data to the bus
Receiver	the device which receives data from the bus
Master	the device which initiates a transfer, generates clock signals and terminates a transfer
Slave	the device addressed by a master
Multi-master	more than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	procedure to ensure that, if more than one master simultaneously tries to control the bus, only one is allowed to do so and the winning message is not corrupted
Synchronization	procedure to synchronize the clock signals of two or more devices

The I²C-bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. As masters are usually microcontrollers, let us consider the case of a data transfer between two microcontrollers connected to the I²C-bus (see Figure 2).

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

Fig 2. Example of an I²C-bus configuration using two microcontrollers

This example highlights the master-slave and receiver-transmitter relationships found on the I²C-bus. Note that these relationships are not permanent, but only depend on the direction of data transfer at that time. The transfer of data would proceed as follows:

1. Suppose microcontroller A wants to send information to microcontroller B:
   - microcontroller A (master), addresses microcontroller B (slave)
   - microcontroller A (master-transmitter), sends data to microcontroller B (slave-receiver)
   - microcontroller A terminates the transfer.
2. If microcontroller A wants to receive information from microcontroller B:
   - microcontroller A (master) addresses microcontroller B (slave)
   - microcontroller A (master-receiver) receives data from microcontroller B (slave-transmitter)
   - microcontroller A terminates the transfer.

Even in this case, the master (microcontroller A) generates the timing and terminates the transfer.

The possibility of connecting more than one microcontroller to the I²C-bus means that more than one master could try to initiate a data transfer at the same time. To avoid the chaos that might ensue from such an event, an arbitration procedure has been developed. This procedure relies on the wired-AND connection of all I²C interfaces to the I²C-bus.

If two or more masters try to put information onto the bus, the first to produce a ‘one’ when the other produces a ‘zero’ loses the arbitration. The clock signals during arbitration are a synchronized combination of the clocks generated by the masters using the wired-AND connection to the SCL line (for more detailed information concerning arbitration see Section 3.1.8).

Generation of clock signals on the I²C-bus is always the responsibility of master devices; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow slave device holding down the clock line or by another master when arbitration occurs.

Table 2 summarizes the use of mandatory and optional portions of the I²C-bus specification and which system configurations use them.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Table 2. Applicability of I²C-bus protocol features

M = mandatory; O = optional; n/a = not applicable.

Feature	Configuration		
	Single master	Multi-master	Slave[1]
START condition	M	M	M
STOP condition	M	M	M
Acknowledge	M	M	M
Synchronization	n/a	M	n/a
Arbitration	n/a	M	n/a
Clock stretching	O[2]	O[2]	O
7-bit slave address	M	M	M
10-bit slave address	O	O	O
General Call address	O	O	O
Software Reset	O	O	O
START byte	n/a	O[3]	n/a
Device ID	n/a	n/a	O

[1] Also refers to a master acting as a slave.

[2] Clock stretching is a feature of some slaves. If no slaves in a system can stretch the clock (hold SCL LOW), the master need not be designed to handle this procedure.

[3] 'Bit banging' (software emulation) multi-master systems should consider a START byte. See Section 3.1.15.

3.1.1 SDA and SCL signals

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a current-source or pull-up resistor (see Figure 3). When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function. Data on the I²C-bus can be transferred at rates of up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in Fast-mode Plus, or up to 3.4 Mbit/s in the High-speed mode. The bus capacitance limits the number of interfaces connected to the bus.

For a single master application, the master's SCL output can be a push-pull driver design if there are no devices on the bus which would stretch the clock.

V_{DD2}, V_{DD3} are device-dependent (for example, 12 V).

Fig 3. Devices with various supply voltages sharing the same bus

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

3.1.2 SDA and SCL logic levels

Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I²C-bus, the levels of the logical ‘0’ (LOW) and ‘1’ (HIGH) are not fixed and depend on the associated level of V_{DD}. Input reference levels are set as 30 % and 70 % of V_{DD}; V_{IL} is $0.3V_{DD}$ and V_{IH} is $0.7V_{DD}$. See Figure 38, timing diagram. Some legacy device input levels were fixed at V_{IL} = 1.5 V and V_{IH} = 3.0 V, but all new devices require this 30 %/70 % specification. See Section 6 for electrical specifications.

3.1.3 Data validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (see Figure 4). One clock pulse is generated for each data bit transferred.

Fig 4. Bit transfer on the I²C-bus

3.1.4 START and STOP conditions

All transactions begin with a START (S) and are terminated by a STOP (P) (see Figure 5). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

Fig 5. START and STOP conditions

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition. This bus free situation is specified in Section 6.

The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical. For the remainder of this document, therefore, the S symbol is used as a generic term to represent both the START and repeated START conditions, unless Sr is particularly relevant.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However, microcontrollers with no such interface have to sample the SDA line at least twice per clock period to sense the transition.

3.1.5 Byte format

Every byte put on the SDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an Acknowledge bit. Data is transferred with the Most Significant Bit (MSB) first (see Figure 6). If a slave cannot receive or transmit another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the clock line SCL LOW to force the master into a wait state. Data transfer then continues when the slave is ready for another byte of data and releases clock line SCL.

Fig 6. Data transfer on the I²C-bus

3.1.6 Acknowledge (ACK) and Not Acknowledge (NACK)

The acknowledge takes place after every byte. The acknowledge bit allows the receiver to signal the transmitter that the byte was successfully received and another byte may be sent. The master generates all clock pulses, including the acknowledge ninth clock pulse.

The Acknowledge signal is defined as follows: the transmitter releases the SDA line during the acknowledge clock pulse so the receiver can pull the SDA line LOW and it remains stable LOW during the HIGH period of this clock pulse (see Figure 4). Set-up and hold times (specified in Section 6) must also be taken into account.

When SDA remains HIGH during this ninth clock pulse, this is defined as the Not Acknowledge signal. The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer. There are five conditions that lead to the generation of a NACK:

1. No receiver is present on the bus with the transmitted address so there is no device to respond with an acknowledge.
2. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master.
3. During the transfer, the receiver gets data or commands that it does not understand.
4. During the transfer, the receiver cannot receive any more data bytes.
5. A master-receiver must signal the end of the transfer to the slave transmitter.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

3.1.7 Clock synchronization

Two masters can begin transmitting on a free bus at the same time and there must be a method for deciding which takes control of the bus and complete its transmission. This is done by clock synchronization and arbitration. In single master systems, clock synchronization and arbitration are not needed.

Clock synchronization is performed using the wired-AND connection of I²C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line causes the masters concerned to start counting off their LOW period and, once a master clock has gone LOW, it holds the SCL line in that state until the clock HIGH state is reached (see Figure 7). However, if another clock is still within its LOW period, the LOW to HIGH transition of this clock may not change the state of the SCL line. The SCL line is therefore held LOW by the master with the longest LOW period. Masters with shorter LOW periods enter a HIGH wait-state during this time.

Fig 7. Clock synchronization during the arbitration procedure

When all masters concerned have counted off their LOW period, the clock line is released and goes HIGH. There is then no difference between the master clocks and the state of the SCL line, and all the masters start counting their HIGH periods. The first master to complete its HIGH period pulls the SCL line LOW again.

In this way, a synchronized SCL clock is generated with its LOW period determined by the master with the longest clock LOW period, and its HIGH period determined by the one with the shortest clock HIGH period.

3.1.8 Arbitration

Arbitration, like synchronization, refers to a portion of the protocol required only if more than one master is used in the system. Slaves are not involved in the arbitration procedure. A master may start a transfer only if the bus is free. Two masters may generate a START condition within the minimum hold time ($t_{HD;STA}$) of the START condition which results in a valid START condition on the bus. Arbitration is then required to determine which master will complete its transmission.

Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks to see if the SDA level matches what it has sent. This process may take many bits. Two masters can actually complete an entire transaction without error, as long as the

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

transmissions are identical. The first time a master tries to send a HIGH, but detects that the SDA level is LOW, the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transaction.

No information is lost during the arbitration process. A master that loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration and must restart its transaction when the bus is free.

If a master also incorporates a slave function and it loses arbitration during the addressing stage, it is possible that the winning master is trying to address it. The losing master must therefore switch over immediately to its slave mode.

Figure 8 shows the arbitration procedure for two masters. More may be involved depending on how many masters are connected to the bus. The moment there is a difference between the internal data level of the master generating DATA1 and the actual level on the SDA line, the DATA1 output is switched off. This does not affect the data transfer initiated by the winning master.

Fig 8. Arbitration procedure of two masters

Since control of the I²C-bus is decided solely on the address and data sent by competing masters, there is no central master, nor any order of priority on the bus.

There is an undefined condition if the arbitration procedure is still in progress at the moment when one master sends a repeated START or a STOP condition while the other master is still sending data. In other words, the following combinations result in an undefined condition:

- Master 1 sends a repeated START condition and master 2 sends a data bit.
- Master 1 sends a STOP condition and master 2 sends a data bit.
- Master 1 sends a repeated START condition and master 2 sends a STOP condition.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

3.1.9 Clock stretching

Clock stretching pauses a transaction by holding the SCL line LOW. The transaction cannot continue until the line is released HIGH again. Clock stretching is optional and in fact, most slave devices do not include an SCL driver so they are unable to stretch the clock.

On the byte level, a device may be able to receive bytes of data at a fast rate, but needs more time to store a received byte or prepare another byte to be transmitted. Slaves can then hold the SCL line LOW after reception and acknowledgment of a byte to force the master into a wait state until the slave is ready for the next byte transfer in a type of handshake procedure (see Figure 7).

On the bit level, a device such as a microcontroller with or without limited hardware for the I²C-bus, can slow down the bus clock by extending each clock LOW period. The speed of any master is adapted to the internal operating rate of this device.

In Hs-mode, this handshake feature can only be used on byte level (see Section 5.3.2).

3.1.10 The slave address and R/$\overline{W}$ bit

Data transfers follow the format shown in Figure 9. After the START condition (S), a slave address is sent. This address is seven bits long followed by an eighth bit which is a data direction bit (R/$\overline{W}$) — a ‘zero’ indicates a transmission (WRITE), a ‘one’ indicates a request for data (READ) (refer to Figure 10). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

Fig 9. A complete data transfer

Fig 10. The first byte after the START procedure

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Possible data transfer formats are:

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed (see Figure 11). The slave receiver acknowledges each byte.
- Master reads slave immediately after first byte (see Figure 12). At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes a slave-transmitter. This first acknowledge is still generated by the slave. The master generates subsequent acknowledges. The STOP condition is generated by the master, which sends a not-acknowledge ($\overline{A}$) just before the STOP condition.
- Combined format (see Figure 13). During a change of direction within a transfer, the START condition and the slave address are both repeated, but with the R/$\overline{W}$ bit reversed. If a master-receiver sends a repeated START condition, it sends a not-acknowledge ($\overline{A}$) just before the repeated START condition.

Notes:

1. Combined formats can be used, for example, to control a serial memory. The internal memory location must be written during the first data byte. After the START condition and slave address is repeated, data can be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations, etc., are taken by the designer of the device.
3. Each byte is followed by an acknowledgment bit as indicated by the A or $\overline{A}$ blocks in the sequence.
4. I²C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address, even if these START conditions are not positioned according to the proper format.
5. A START condition immediately followed by a STOP condition (void message) is an illegal format. Many devices however are designed to operate properly under this condition.
6. Each device connected to the bus is addressable by a unique address. Normally a simple master/slave relationship exists, but it is possible to have multiple identical slaves that can receive and respond simultaneously, for example in a group broadcast. This technique works best when using bus switching devices like the PCA9546A where all four channels are on and identical devices are configured at the same time, understanding that it is impossible to determine that each slave acknowledges, and then turn on one channel at a time to read back each individual device's configuration to confirm the programming. Refer to individual component data sheets.

All information provided in this document is subject to legal disclaimers.
© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 11. A master-transmitter addressing a slave receiver with a 7-bit address (the transfer direction is not changed)

Fig 12. A master reads a slave immediately after the first byte

Fig 13. Combined format

3.1.11 10-bit addressing

10-bit addressing expands the number of possible addresses. Devices with 7-bit and 10-bit addresses can be connected to the same I²C-bus, and both 7-bit and 10-bit addressing can be used in all bus speed modes. Currently, 10-bit addressing is not being widely used.

The 10-bit slave address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr).

The first seven bits of the first byte are the combination 1111 0XX of which the last two bits (XX) are the two Most-Significant Bits (MSB) of the 10-bit address; the eighth bit of the first byte is the R/$\overline{W}$ bit that determines the direction of the message.

Although there are eight possible combinations of the reserved address bits 1111 XXX, only the four combinations 1111 0XX are used for 10-bit addressing. The remaining four combinations 1111 1XX are reserved for future I²C-bus enhancements.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

All combinations of read/write formats previously described for 7-bit addressing are possible with 10-bit addressing. Two are detailed here:

- Master-transmitter transmits to slave-receiver with a 10-bit slave address.
  The transfer direction is not changed (see Figure 14). When a 10-bit address follows a START condition, each slave compares the first seven bits of the first byte of the slave address (1111 0XX) with its own address and tests if the eighth bit (R/$\overline{W}$ direction bit) is 0. It is possible that more than one device finds a match and generate an acknowledge (A1). All slaves that found a match compare the eight bits of the second byte of the slave address (XXXX XXXX) with their own addresses, but only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.
- Master-receiver reads slave-transmitter with a 10-bit slave address.
  The transfer direction is changed after the second R/$\overline{W}$ bit (Figure 15). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks if the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and tests if the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or until it receives another repeated START condition (Sr) followed by a different slave address. After a repeated START condition (Sr), all the other slave devices will also compare the first seven bits of the first byte of the slave address (1111 0XX) with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them will be addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 1111 0XX slave address (for 7-bit devices) does not match.

Fig 14. A master-transmitter addresses a slave-receiver with a 10-bit address

Fig 15. A master-receiver addresses a slave-transmitter with a 10-bit address

Slave devices with 10-bit addressing react to a ‘general call’ in the same way as slave devices with 7-bit addressing. Hardware masters can transmit their 10-bit address after a ‘general call’. In this case, the ‘general call’ address byte is followed by two successive bytes containing the 10-bit address of the master-transmitter. The format is as shown in Figure 15 where the first DATA byte contains the eight least-significant bits of the master address.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

The START byte 0000 0001 (01h) can precede the 10-bit addressing in the same way as for 7-bit addressing (see Section 3.1.15).

3.1.12 Reserved addresses

Two groups of eight addresses (0000 XXX and 1111 XXX) are reserved for the purposes shown in Table 3.

Table 3. Reserved addresses

X = don't care; 1 = HIGH; 0 = LOW.

Slave address	R/$\overline{W}$ bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	CBUS address[3]
0000 010	X	reserved for different bus format[4]
0000 011	X	reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	1	device ID
1111 0XX	X	10-bit slave addressing

[1] The general call address is used for several functions including software reset.

[2] No device is allowed to acknowledge at the reception of the START byte.

[3] The CBUS address has been reserved to enable the inter-mixing of CBUS compatible and I²C-bus compatible devices in the same system. I²C-bus compatible devices are not allowed to respond on reception of this address.

[4] The address reserved for a different bus format is included to enable I²C and other protocols to be mixed. Only I²C-bus compatible devices that can work with such formats and protocols are allowed to respond to this address.

Assignment of addresses within a local system is up to the system architect who must take into account the devices being used on the bus and any future interaction with other conventional I²C-buses. For example, a device with seven user-assignable address pins allows all 128 addresses to be assigned. If it is known that the reserved address is never going to be used for its intended purpose, a reserved address can be used for a slave address.

3.1.13 General call address

The general call address is for addressing every device connected to the I²C-bus at the same time. However, if a device does not need any of the data supplied within the general call structure, it can ignore this address by not issuing an acknowledgment. If a device does require data from a general call address, it acknowledges this address and behave as a slave-receiver. The master does not actually know how many devices acknowledged if one or more devices respond. The second and following bytes are acknowledged by every slave-receiver capable of handling this data. A slave who cannot process one of these bytes must ignore it by not-acknowledging. Again, if one or more slaves acknowledge, the not-acknowledge will not be seen by the master. The meaning of the general call address is always specified in the second byte (see Figure 16).

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 16. General call address format

There are two cases to consider:

- When the least significant bit B is a ‘zero’.
- When the least significant bit B is a ‘one’.

When bit B is a ‘zero’, the second byte has the following definition:

- **0000 0110 (06h): Reset and write programmable part of slave address by hardware.** On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address. Precautions must be taken to ensure that a device is not pulling down the SDA or SCL line after applying the supply voltage, since these low levels would block the bus.
- **0000 0100 (04h): Write programmable part of slave address by hardware.** Behaves as above, but the device does not reset.
- **0000 0000 (00h): This code is not allowed to be used as the second byte.**

Sequences of programming procedure are published in the appropriate device data sheets. The remaining codes have not been fixed and devices must ignore them.

When bit B is a ‘one’, the 2-byte sequence is a ‘hardware general call’. This means that the sequence is transmitted by a hardware master device, such as a keyboard scanner, which can be programmed to transmit a desired slave address. Since a hardware master does not know in advance to which device the message has to be transferred, it can only generate this hardware general call and its own address — identifying itself to the system (see Figure 17).

Fig 17. Data transfer from a hardware master-transmitter

The seven bits remaining in the second byte contain the address of the hardware master. This address is recognized by an intelligent device (for example, a microcontroller) connected to the bus which then accepts the information from the hardware master. If the hardware master can also act as a slave, the slave address is identical to the master address.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

In some systems, an alternative could be that the hardware master transmitter is set in the slave-receiver mode after the system reset. In this way, a system configuring master can tell the hardware master-transmitter (which is now in slave-receiver mode) to which address data must be sent (see Figure 18). After this programming procedure, the hardware master remains in the master-transmitter mode.

a. Configuring master sends dump address to hardware master

b. Hardware master dumps data to selected slave

Fig 18. Data transfer by a hardware-transmitter capable of dumping data directly to slave devices

3.1.14 Software reset

Following a General Call, (0000 0000), sending 0000 0110 (06h) as the second byte causes a software reset. This feature is optional and not all devices respond to this command. On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address. Precautions must be taken to ensure that a device is not pulling down the SDA or SCL line after applying the supply voltage, since these low levels would block the bus.

3.1.15 START byte

Microcontrollers can be connected to the I²C-bus in two ways. A microcontroller with an on-chip hardware I²C-bus interface can be programmed to be only interrupted by requests from the bus. When the device does not have such an interface, it must constantly monitor the bus via software. Obviously, the more times the microcontroller monitors, or polls the bus, the less time it can spend carrying out its intended function.

There is therefore a speed difference between fast hardware devices and a relatively slow microcontroller which relies on software polling.

In this case, data transfer can be preceded by a start procedure which is much longer than normal (see Figure 19). The start procedure consists of:

- A START condition (S)
- A START byte (0000 0001)
- An acknowledge clock pulse (ACK)
- A repeated START condition (Sr).

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 19. START byte procedure

After the START condition S has been transmitted by a master which requires bus access, the START byte (0000 0001) is transmitted. Another microcontroller can therefore sample the SDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this LOW level on the SDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr which is then used for synchronization.

A hardware receiver resets upon receipt of the repeated START condition Sr and therefore ignores the START byte.

An acknowledge-related clock pulse is generated after the START byte. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

3.1.16 Bus clear

In the unlikely event where the clock (SCL) is stuck LOW, the preferential procedure is to reset the bus using the HW reset signal if your I²C devices have HW reset inputs. If the I²C devices do not have HW reset inputs, cycle power to the devices to activate the mandatory internal Power-On Reset (POR) circuit.

If the data line (SDA) is stuck LOW, the master should send nine clock pulses. The device that held the bus LOW should release it sometime within those nine clocks. If not, then use the HW reset or cycle power to clear the bus.

3.1.17 Device ID

The Device ID field (see Figure 20) is an optional 3-byte read-only (24 bits) word giving the following information:

- Twelve bits with the manufacturer name, unique per manufacturer (for example, NXP)
- Nine bits with the part identification, assigned by manufacturer (for example, PCA9698)
- Three bits with the die revision, assigned by manufacturer (for example, RevX)

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 20. Device ID field

The Device ID is read-only, hard-wired in the device and can be accessed as follows:

1. START condition
2. The master sends the Reserved Device ID I²C-bus address followed by the R/$\overline{W}$ bit set to ‘0’ (write): ‘1111 1000’.
3. The master sends the I²C-bus slave address of the slave device it must identify. The LSB is a ‘Don’t care’ value. Only one device must acknowledge this byte (the one that has the I²C-bus slave address).
4. The master sends a Re-START condition.

   Remark: A STOP condition followed by a START condition resets the slave state machine and the Device ID Read cannot be performed. Also, a STOP condition or a Re-START condition followed by an access to another slave device resets the slave state machine and the Device ID Read cannot be performed.
5. The master sends the Reserved Device ID I²C-bus address followed by the R/$\overline{W}$ bit set to ‘1’ (read): ‘1111 1001’.
6. The Device ID Read can be done, starting with the 12 manufacturer bits (first byte + four MSBs of the second byte), followed by the nine part identification bits (four LSBs of the second byte + five MSBs of the third byte), and then the three die revision bits (three LSBs of the third byte).
7. The master ends the reading sequence by NACKing the last byte, thus resetting the slave device state machine and allowing the master to send the STOP condition.

   Remark: The reading of the Device ID can be stopped anytime by sending a NACK.

If the master continues to ACK the bytes after the third byte, the slave rolls back to the first byte and keeps sending the Device ID sequence until a NACK has been detected.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Table 4. Assigned manufacturer IDs

Manufacturer bits												Company
11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	NXP Semiconductors
0	0	0	0	0	0	0	0	0	0	0	1	NXP Semiconductors (reserved)
0	0	0	0	0	0	0	0	0	0	1	0	NXP Semiconductors (reserved)
0	0	0	0	0	0	0	0	0	0	1	1	NXP Semiconductors (reserved)
0	0	0	0	0	0	0	0	0	1	0	0	Ramtron International
0	0	0	0	0	0	0	0	0	1	0	1	Analog Devices
0	0	0	0	0	0	0	0	0	1	1	0	STMicroelectronics
0	0	0	0	0	0	0	0	0	1	1	1	ON Semiconductor
0	0	0	0	0	0	0	0	1	0	0	0	Sprintek Corporation
0	0	0	0	0	0	0	0	1	0	0	1	ESPROS Photonics AG
0	0	0	0	0	0	0	0	1	0	1	0	Fujitsu Semiconductor
0	0	0	0	0	0	0	0	1	0	1	1	Flir
0	0	0	0	0	0	0	0	1	1	0	0	O_2Micro
0	0	0	0	0	0	0	0	1	1	0	1	Atmel

Designers of new I²C devices who want to implement the device ID feature should contact NXP at **i2c.support@nxp.com** to have a unique manufacturer ID assigned.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

3.2 Ultra Fast-mode I²C-bus protocol

The UFm I²C-bus is a 2-wire push-pull serial bus that operates from DC to 5 MHz transmitting data in one direction. It is most useful for speeds greater than 1 MHz to drive LED controllers and other devices that do not need feedback. The UFm I²C-bus protocol is based on the standard I²C-bus protocol that consists of a START, slave address, command bit, ninth clock, and a STOP bit. The command bit is a 'write' only, and the data bit on the ninth clock is driven HIGH, ignoring the ACK cycle due to the unidirectional nature of the bus. The 2-wire push-pull driver consists of a UFm serial clock (USCL) and serial data (USDA).

Slave devices contain a unique address (whether it is a microcontroller, LCD driver, LED controller, GPO) and operate only as receivers. An LED driver may be only a receiver and can be supported by UFm, whereas a memory can both receive and transmit data and is not supported by UFm.

Since UFm I²C-bus uses push-pull drivers, it does not have the multi-master capability of the wired-AND open-drain Sm, Fm, and Fm+ I²C-buses. In UFm, a master is the only device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. All other devices addressed are considered slaves.

Table 5. Definition of UFm I²C-bus terminology

Term	Description
Transmitter	the device that sends data to the bus
Receiver	the device that receives data from the bus
Master	the device that initiates a transfer, generates clock signals and terminates a transfer
Slave	the device addressed by a master

Let us consider the case of a data transfer between a master and multiple slaves connected to the UFm I²C-bus (see Figure 21).

Fig 21. Example of UFm I²C-bus configuration

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

This highlights the master/transmitter-slave/receiver relationship found on the UFm I²C-bus. Note that these relationships are permanent, as data transfer is only permitted in one direction. The transfer of data would proceed as follows:

Suppose that the master ASIC wants to send information to the LED controller 2:

- ASIC A (master-transmitter), addresses LED controller 2 (slave-receiver) by sending the address on the USDA and generating the clock on USCL.
- ASIC A (master-transmitter), sends data to LED controller 2 (slave-receiver) on the USDA and generates the clock on USCL.
- ASIC A terminates the transfer.

The possibility of connecting more than one UFm master to the UFm I²C-bus is not allowed due to bus contention on the push-pull outputs. If an additional master is required in the system, it must be fully isolated from the other master (that is, with a true 'one hot' MUX) as only one master is allowed on the bus at a time.

Generation of clock signals on the UFm I²C-bus is always the responsibility of the master device, that is, the master generates the clock signals when transferring data on the bus. Bus clock signals from a master cannot be altered by a slave device with clock stretching and the process of arbitration and clock synchronization does not exist within the UFm I²C-bus.

Table 6 summarizes the use of mandatory and optional portions of the UFm I²C-bus specification.

Table 6. Applicability of I²C-bus features to UFm

M = mandatory; O = optional; n/p = not possible

Feature	Configuration
	Single master
START condition	M
STOP condition	M
Acknowledge	n/p
Synchronization	n/p
Arbitration	n/p
Clock stretching	n/p
7-bit slave address	M
10-bit slave address	O
General Call address	O
Software Reset	O
START byte	O
Device ID	n/p

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

3.2.1 USDA and USCL signals

Both USDA and USCL are unidirectional lines, with push-pull outputs. When the bus is free, both lines are pulled HIGH by the upper transistor of the output stage. Data on the I²C-bus can be transferred at rates of up to 5000 kbit/s in the Ultra Fast-mode. The number of interfaces connected to the bus is limited by the bus loading, reflections from cable ends, connectors, and stubs.

Fig 22. Simplified schematic of USCL, USDA outputs

3.2.2 USDA and USCL logic levels

Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I²C-bus, the levels of the logical ‘0’ (LOW) and ‘1’ (HIGH) are not fixed and depend on the associated level of V_{DD}. Input reference levels are set as 30 % and 70 % of V_{DD}; V_{IL} is $0.3V_{DD}$ and V_{IH} is $0.7V_{DD}$. See Figure 40, timing diagram. See Section 6 for electrical specifications.

3.2.3 Data validity

The data on the USDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the USCL line is LOW (see Figure 23). One clock pulse is generated for each data bit transferred.

Fig 23. Bit transfer on the UFm I²C-bus

3.2.4 START and STOP conditions

Both data and clock lines remain HIGH when the bus is not busy. All transactions begin with a START (S) and can be terminated by a STOP (P) (see Figure 24). A HIGH to LOW transition on the USDA line while USCL is HIGH defines a START condition. A LOW to HIGH transition on the USDA line while USCL is HIGH defines a STOP condition.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 24. Definition of START and STOP conditions for UFm I²C-bus

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition. This bus free situation is specified in Section 6. The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical. For the remainder of this document, therefore, the S symbol is used as a generic term to represent both the START and repeated START conditions, unless Sr is particularly relevant.

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However, microcontrollers with no such interface have to sample the USDA line at least twice per clock period to sense the transition.

3.2.5 Byte format

Every byte put on the USDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted. The master drives the USDA HIGH after each byte during the Acknowledge cycle. Data is transferred with the Most Significant Bit (MSB) first (see Figure 25). A slave is not allowed to hold the clock LOW if it cannot receive another complete byte of data or while it is performing some other function, for example servicing an internal interrupt.

Fig 25. Data transfer on the UFm I²C-bus

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

3.2.6 Acknowledge (ACK) and Not Acknowledge (NACK)

Since the slaves are not able to respond the ninth clock cycle, the ACK and NACK are not required. However, the clock cycle is preserved in the UFm to be compatible with the I²C-bus protocol. The ACK and NACK are also referred to as the ninth clock cycle. The master generates all clock pulses, including the ninth clock pulse. The ninth data bit is always driven HIGH ('1'). Slave devices are not allowed to drive the SDA line at any time.

3.2.7 The slave address and R/$\overline{W}$ bit

Data transfers follow the format shown in Figure 26. After the START condition (S), a slave address is sent. This address is seven bits long followed by an eighth bit which is a data direction bit ($\overline{W}$) — a 'zero' indicates a transmission (WRITE); a 'one' indicates a request for data (READ) and is not supported by UFm (except for the START byte, Section 3.2.12) since the communication is unidirectional (refer to Figure 27). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition.

Fig 26. A complete UFm data transfer

Fig 27. The first byte after the START procedure

The UFm data transfer format is:

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed (see Figure 28). The master never acknowledges because it never receives any data but generates the '1' on the ninth bit for the slave to conform to the I²C-bus protocol.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 28. A master-transmitter addressing a slave receiver with a 7-bit address

Notes:

1. Individual transaction or repeated START formats addressing multiple slaves in one transaction can be used. After the START condition and slave address is repeated, data can be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations, etc., are taken by the designer of the device.
3. Each byte is followed by a Not-Acknowledgment bit as indicated by the $\overline{A}$ blocks in the sequence.
4. I²C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address, even if these START conditions are not positioned according to the proper format.
5. A START condition immediately followed by a STOP condition (void message) is an illegal format. Many devices however are designed to operate properly under this condition.
6. Each device connected to the bus is addressable by a unique address. A simple master/slave relationship exists, but it is possible to have multiple identical slaves that can receive and respond simultaneously, for example, in a group broadcast where all identical devices are configured at the same time, understanding that it is impossible to determine that each slave is responsive. Refer to individual component data sheets.

3.2.8 10-bit addressing

10-bit addressing expands the number of possible addresses. Devices with 7-bit and 10-bit addresses can be connected to the same I²C-bus, and both 7-bit and 10-bit addressing can be used in all bus speed modes.

The 10-bit slave address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr). The first seven bits of the first byte are the combination 1111 0XX of which the last two bits (XX) are the two Most Significant Bits (MSBs) of the 10-bit address; the eighth bit of the first byte is the R/$\overline{W}$ bit that determines the direction of the message.

Although there are eight possible combinations of the reserved address bits 1111 XXX, only the four combinations 1111 0XX are used for 10-bit addressing. The remaining four combinations 1111 1XX are reserved for future I²C-bus enhancements.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Only the write format previously described for 7-bit addressing is possible with 10-bit addressing. Detailed here:

- Master-transmitter transmits to slave-receiver with a 10-bit slave address. The transfer direction is not changed (see Figure 29). When a 10-bit address follows a START condition, each slave compares the first seven bits of the first byte of the slave address (1111 0XX) with its own address and tests if the eighth bit (R/$\overline{W}$ direction bit) is 0 ($\overline{W}$). All slaves that found a match compare the eight bits of the second byte of the slave address (XXXX XXXX) with their own addresses, but only one slave finds a match. The matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

Fig 29. A master-transmitter addresses a slave-receiver with a 10-bit address

The START byte 0000 0001 (01h) can precede the 10-bit addressing in the same way as for 7-bit addressing (see Section 3.2.12).

3.2.9 Reserved addresses in UFm

The UFm I²C-bus has a different physical layer than the other I²C-bus modes. Therefore the available slave address range is different. Two groups of eight addresses (0000 XXX and 1111 XXX) are reserved for the purposes shown in Table 7.

Table 7. Reserved addresses

X = don't care; 1 = HIGH; 0 = LOW.

Slave address	R/$\overline{W}$ bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	reserved for future purposes
0000 010	X	reserved for future purposes
0000 011	X	reserved for future purposes
0000 1XX	X	reserved for future purposes
1111 1XX	X	reserved for future purposes
1111 0XX	X	10-bit slave addressing

[1] The general call address is used for several functions including software reset.

[2] No UFm device is allowed to acknowledge at the reception of the START byte.

Assignment of addresses within a local system is up to the system architect who must take into account the devices being used on the bus and any future interaction with reserved addresses. For example, a device with seven user-assignable address pins allows all 128 addresses to be assigned. If it is known that the reserved address is never going to be used for its intended purpose, then a reserved address can be used for a slave address.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

3.2.10 General call address

The general call address is for addressing every device connected to the I²C-bus at the same time. However, if a device does not need any of the data supplied within the general call structure, it can ignore this address. If a device does require data from a general call address, it behaves as a slave-receiver. The master does not actually know how many devices are responsive to the general call. The second and following bytes are received by every slave-receiver capable of handling this data. A slave that cannot process one of these bytes must ignore it. The meaning of the general call address is always specified in the second byte (see Figure 30).

Fig 30. General call address format

There are two cases to consider:

- When the least significant bit B is a ‘zero’
- When the least significant bit B is a ‘one’

When bit B is a ‘zero’, the second byte has the following definition:

0000 0110 (06h) — Reset and write programmable part of slave address by hardware. On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address.

0000 0100 (04h) — Write programmable part of slave address by hardware. Behaves as above, but the device does not reset.

0000 0000 (00h) — This code is not allowed to be used as the second byte. Sequences of programming procedure are published in the appropriate device data sheets. The remaining codes have not been fixed and devices must ignore them.

When bit B is a ‘one’, the 2-byte sequence is ignored.

3.2.11 Software reset

Following a General Call, (0000 0000), sending 0000 0110 (06h) as the second byte causes a software reset. This feature is optional and not all devices respond to this command. On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address.

3.2.12 START byte

Microcontrollers can be connected to the I²C-bus in two ways. A microcontroller with an on-chip hardware I²C-bus interface can be programmed to be only interrupted by requests from the bus. When the device does not have such an interface, it must constantly monitor the bus via software. Obviously, the more times the microcontroller monitors, or polls the bus, the less time it can spend carrying out its intended function.

There is therefore a speed difference between fast hardware devices and a relatively slow microcontroller which relies on software polling.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

In this case, data transfer can be preceded by a start procedure which is much longer than normal (see Figure 31). The start procedure consists of:

- A START condition (S)
- A START byte (0000 0001)
- A Not Acknowledge clock pulse (NACK)
- A repeated START condition (Sr)

Fig 31. START byte procedure

After the START condition S has been transmitted by a master which requires bus access, the START byte (0000 0001) is transmitted. Another microcontroller can therefore sample the USDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this LOW level on the USDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr, which is then used for synchronization. A hardware receiver resets upon receipt of the repeated START condition Sr and therefore ignores the START byte. An acknowledge-related clock pulse is generated after the START byte. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

3.2.13 Unresponsive slave reset

In the unlikely event where the slave becomes unresponsive (for example, determined through external feedback, not through UFm I²C-bus), the preferential procedure is to reset the slave by using the software reset command or the hardware reset signal. If the slaves do not support these features, then cycle power to the devices to activate the mandatory internal Power-On Reset (POR) circuit.

3.2.14 Device ID

The Device ID field is not supported in UFm.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

4. Other uses of the I²C-bus communications protocol

The I²C-bus is used as the communications protocol for several system architectures. These architectures have added command sets and application-specific extensions in addition to the base I²C specification. In general, simple I²C-bus devices such as I/O extenders could be used in any one of these architectures since the protocol and physical interfaces are the same.

4.1 CBUS compatibility

CBUS receivers can be connected to the Standard-mode I²C-bus. However, a third bus line called DLEN must then be connected and the acknowledge bit omitted. Normally, I²C transmissions are sequences of 8-bit bytes; CBUS compatible devices have different formats.

In a mixed bus structure, I²C-bus devices must not respond to the CBUS message. For this reason, a special CBUS address (0000 001X) to which no I²C-bus compatible device responds has been reserved. After transmission of the CBUS address, the DLEN line can be made active and a CBUS-format transmission sent. After the STOP condition, all devices are again ready to accept data.

Master-transmitters can send CBUS formats after sending the CBUS address. The transmission is ended by a STOP condition, recognized by all devices.

Remark: If the CBUS configuration is known, and expansion with CBUS compatible devices is not foreseen, the designer is allowed to adapt the hold time to the specific requirements of the device(s) used.

4.2 SMBus - System Management Bus

The SMBus uses I²C hardware and I²C hardware addressing, but adds second-level software for building special systems. In particular, its specifications include an Address Resolution Protocol that can make dynamic address allocations.

Dynamic reconfiguration of the hardware and software allow bus devices to be ‘hot-plugged’ and used immediately, without restarting the system. The devices are recognized automatically and assigned unique addresses. This advantage results in a plug-and-play user interface. In both those protocols, there is a very useful distinction made between a System Host and all the other devices in the system that can have the names and functions of masters or slaves.

SMBus is used today as a system management bus in most PCs. Developed by Intel and others in 1995, it modified some I²C electrical and software characteristics for better compatibility with the quickly decreasing power supply budget of portable equipment. SMBus also has a ‘High Power’ version 2.0 that includes a 4 mA sink current that cannot be driven by I²C chips unless the pull-up resistor is sized to I²C-bus levels.

4.2.1 I²C/SMBus compliancy

SMBus and I²C protocols are basically the same: A SMBus master is able to control I²C devices and vice versa at the protocol level. The SMBus clock is defined from 10 kHz to 100 kHz while I²C can be 0 Hz to 100 kHz, 0 Hz to 400 kHz, 0 Hz to 1 MHz and 0 Hz to 3.4 MHz, depending on the mode. This means that an I²C-bus running at less than 10 kHz is not SMBus compliant since the SMBus devices may time-out.

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

Logic levels are slightly different also: TTL for SMBus: LOW = 0.8 V and HIGH = 2.1 V, versus the 30 %/70 % V_{DD} CMOS level for I²C. This is not a problem if V_{DD} > 3.0 V. If the I²C device is below 3.0 V, then there could be a problem if the logic HIGH/LOW levels are not properly recognized.

4.2.2 Time-out feature

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency of 10 kHz to prevent locking up the bus. I²C can be a 'DC' bus, meaning that a slave device stretches the master clock when performing some routine while the master is accessing it. This notifies the master that the slave is busy but does not want to lose the communication. The slave device will allow continuation after its task is complete. There is no limit in the I²C-bus protocol as to how long this delay can be, whereas for a SMBus system, it would be limited to 35 ms.

SMBus protocol just assumes that if something takes too long, then it means that there is a problem on the bus and that all devices must reset in order to clear this mode. Slave devices are not then allowed to hold the clock LOW too long.

4.2.3 Differences between SMBus 1.0 and SMBus 2.0

The SMBus specification defines two classes of electrical characteristics: low power and high power. The first class, originally defined in the SMBus 1.0 and 1.1 specifications, was designed primarily with Smart Batteries in mind, but could be used with other low-power devices.

The 2.0 version introduces an alternative higher power set of electrical characteristics. This class is appropriate for use when higher drive capability is required, for example with SMBus devices on PCI add-in cards and for connecting such cards across the PCI connector between each other and to SMBus devices on the system board.

Devices may be powered by the bus V_{DD} or by another power source, V_{Bus} (as with, for example, Smart Batteries), and will inter-operate as long as they adhere to the SMBus electrical specifications for their class.

NXP devices have a higher power set of electrical characteristics than SMBus 1.0. The main difference is the current sink capability with V_{OL} = 0.4 V.

- SMBus low power = 350 μA
- SMBus high power = 4 mA
- I²C-bus = 3 mA

SMBus 'high power' devices and I²C-bus devices will work together if the pull-up resistor is sized for 3 mA.

For more information, refer to: www.smbus.org/.

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

4.3 PMBus - Power Management Bus

PMBus is a standard way to communicate between power converters and a system host over the SMBus to provide more intelligent control of the power converters. The PMBus specification defines a standard set of device commands so that devices from multiple sources function identically. PMBus devices use the SMBus Version 1.1 plus extensions for transport.

For more information, refer to: www.pmbus.org/.

4.4 Intelligent Platform Management Interface (IPMI)

Intelligent Platform Management Interface (IPMI) defines a standardized, abstracted, message-based interface for intelligent platform management hardware. IPMI also defines standardized records for describing platform management devices and their characteristics. IPMI increases reliability of systems by monitoring parameters such as temperatures, voltages, fans and chassis intrusion.

IPMI provides general system management functions such as automatic alerting, automatic system shutdown and restart, remote restart and power control. The standardized interface to intelligent platform management hardware aids in prediction and early monitoring of hardware failures as well as diagnosis of hardware problems.

This standardized bus and protocol for extending management control, monitoring, and event delivery within the chassis:

- I²C based
- Multi-master
- Simple Request/Response Protocol
- Uses IPMI Command sets
- Supports non-IPMI devices
- Physically I²C but write-only (master capable devices); hot swap not required
- Enables the Baseboard Management Controller (BMC) to accept IPMI request messages from other management controllers in the system
- Allows non-intelligent devices as well as management controllers on the bus
- BMC serves as a controller to give system software access to IPMB.

Hardware implementation is isolated from software implementation so that new sensors and events can then be added without any software changes.

For more information, refer to: www.intel.com/design/servers/ipmi/ipmi.htm.

All information provided in this document is subject to legal disclaimers.
© NXP Semiconductors N.V. 2014. All rights reserved.

4.5 Advanced Telecom Computing Architecture (ATCA)

Advanced Telecom Computing Architecture (ATCA) is a follow-on to Compact PCI (cPCI), providing a standardized form-factor with larger card area, larger pitch and larger power supply for use in advanced rack-mounted telecom hardware. It includes a fault-tolerant scheme for thermal management that uses I²C-bus communications between boards.

Advanced Telecom Computing Architecture (ATCA) is backed by more than 100 companies including many of the large players such as Intel, Lucent, and Motorola.

There are two general compliant approaches to an ATCA-compliant fan control: the first is an Intelligent FRU (Field Replaceable Unit) which means that the fan control would be directly connected to the IPMB (Intelligent Platform Management Bus); the second is a Managed or Non-intelligent FRU.

One requirement is the inclusion of hardware and software to manage the dual I²C-buses. This requires an on-board isolated supply to power the circuitry, a buffered dual I²C-bus with rise time accelerators, and 3-state capability. The I²C controller must be able to support a multi-master I²C dual bus and handle the standard set of fan commands outlined in the protocol. In addition, on-board temperature reporting, tray capability reporting, fan turn-off capabilities, and non-volatile storage are required.

For more information, refer to: www.picmg.org/v2internal/resourcepage2.cfm?id=2.

4.6 Display Data Channel (DDC)

The Display Data Channel (DDC) allows a monitor or display to inform the host about its identity and capabilities. The specification for DDC version 2 calls for compliance with the I²C-bus standard mode specification. It allows bidirectional communication between the display and the host, enabling control of monitor functions such as how images are displayed and communication with other devices attached to the I²C-bus.

For more information, refer to: www.vesa.org.

5. Bus speeds

Originally, the I²C-bus was limited to 100 kbit/s operation. Over time there have been several additions to the specification so that there are now five operating speed categories. Standard-mode, Fast-mode (Fm), Fast-mode Plus (Fm+), and High-speed mode (Hs-mode) devices are downward-compatible — any device may be operated at a lower bus speed. Ultra Fast-mode devices are not compatible with previous versions since the bus is unidirectional.

- Bidirectional bus:
  - **Standard-mode (Sm)**, with a bit rate up to 100 kbit/s
  - **Fast-mode (Fm)**, with a bit rate up to 400 kbit/s
  - **Fast-mode Plus (Fm+)**, with a bit rate up to 1 Mbit/s
  - **High-speed mode (Hs-mode)**, with a bit rate up to 3.4 Mbit/s.
- Unidirectional bus:
  - **Ultra Fast-mode (UFm)**, with a bit rate up to 5 Mbit/s

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

5.1 Fast-mode

Fast-mode devices can receive and transmit at up to 400 kbit/s. The minimum requirement is that they can synchronize with a 400 kbit/s transfer; they can then prolong the LOW period of the SCL signal to slow down the transfer. The protocol, format, logic levels and maximum capacitive load for the SDA and SCL lines are the same as the Standard-mode I^2C-bus specification. Fast-mode devices are downward-compatible and can communicate with Standard-mode devices in a 0 to 100 kbit/s I^2C-bus system. As Standard-mode devices, however, are not upward compatible; they should not be incorporated in a Fast-mode I^2C-bus system as they cannot follow the higher transfer rate and unpredictable states would occur.

The Fast-mode I^2C-bus specification has the following additional features compared with the Standard-mode:

- The maximum bit rate is increased to 400 kbit/s.
- Timing of the serial data (SDA) and serial clock (SCL) signals has been adapted. There is no need for compatibility with other bus systems such as CBUS because they cannot operate at the increased bit rate.
- The inputs of Fast-mode devices incorporate spike suppression and a Schmitt trigger at the SDA and SCL inputs.
- The output buffers of Fast-mode devices incorporate slope control of the falling edges of the SDA and SCL signals.
- If the power supply to a Fast-mode device is switched off, the SDA and SCL I/O pins must be floating so that they do not obstruct the bus lines.

The external pull-up devices connected to the bus lines must be adapted to accommodate the shorter maximum permissible rise time for the Fast-mode I^2C-bus. For bus loads up to 200 pF, the pull-up device for each bus line can be a resistor; for bus loads between 200 pF and 400 pF, the pull-up device can be a current source (3 mA max.) or a switched resistor circuit (see Section 7.2.4).

5.2 Fast-mode Plus

Fast-mode Plus (Fm+) devices offer an increase in I^2C-bus transfer speeds and total bus capacitance. Fm+ devices can transfer information at bit rates of up to 1 Mbit/s, yet they remain fully downward compatible with Fast- or Standard-mode devices for bidirectional communication in a mixed-speed bus system. The same serial bus protocol and data format is maintained as with the Fast- or Standard-mode system. Fm+ devices also offer increased drive current over Fast- or Standard-mode devices allowing them to drive longer and/or more heavily loaded buses so that bus buffers do not need to be used.

The drivers in Fast-mode Plus parts are strong enough to satisfy the Fast-mode Plus timing specification with the same 400 pF load as Standard-mode parts. To be backward compatible with Standard-mode, they are also tolerant of the 1 μs rise time of Standard-mode parts. In applications where only Fast-mode Plus parts are present, the high drive strength and tolerance for slow rise and fall times allow the use of larger bus capacitance as long as set-up, minimum LOW time and minimum HIGH time for Fast-mode Plus are all satisfied and the fall time and rise time do not exceed the 300 ns t_f and 1 μs t_r specifications of Standard-mode. Bus speed can be traded against load capacitance to increase the maximum capacitance by about a factor of ten.

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

5.3 Hs-mode

High-speed mode (Hs-mode) devices offer a quantum leap in I²C-bus transfer speeds. Hs-mode devices can transfer information at bit rates of up to 3.4 Mbit/s, yet they remain fully downward compatible with Fast-mode Plus, Fast- or Standard-mode (F/S) devices for bidirectional communication in a mixed-speed bus system. With the exception that arbitration and clock synchronization is not performed during the Hs-mode transfer, the same serial bus protocol and data format is maintained as with the F/S-mode system.

5.3.1 High speed transfer

To achieve a bit transfer of up to 3.4 Mbit/s, the following improvements have been made to the regular I²C-bus specification:

- Hs-mode master devices have an open-drain output buffer for the SDAH signal and a combination of an open-drain pull-down and current-source pull-up circuit on the SCLH output. This current-source circuit shortens the rise time of the SCLH signal. Only the current-source of one master is enabled at any one time, and only during Hs-mode.
- No arbitration or clock synchronization is performed during Hs-mode transfer in multi-master systems, which speeds-up bit handling capabilities. The arbitration procedure always finishes after a preceding master code transmission in F/S-mode.
- Hs-mode master devices generate a serial clock signal with a HIGH to LOW ratio of 1 to 2. This relieves the timing requirements for set-up and hold times.
- As an option, Hs-mode master devices can have a built-in bridge. During Hs-mode transfer, the high-speed data (SDAH) and high-speed serial clock (SCLH) lines of Hs-mode devices are separated by this bridge from the SDA and SCL lines of F/S-mode devices. This reduces the capacitive load of the SDAH and SCLH lines resulting in faster rise and fall times.
- The only difference between Hs-mode slave devices and F/S-mode slave devices is the speed at which they operate. Hs-mode slaves have open-drain output buffers on the SCLH and SDAH outputs. Optional pull-down transistors on the SCLH pin can be used to stretch the LOW level of the SCLH signal, although this is only allowed after the acknowledge bit in Hs-mode transfers.
- The inputs of Hs-mode devices incorporate spike suppression and a Schmitt trigger at the SDAH and SCLH inputs.
- The output buffers of Hs-mode devices incorporate slope control of the falling edges of the SDAH and SCLH signals.

Figure 32 shows the physical I²C-bus configuration in a system with only Hs-mode devices. Pins SDA and SCL on the master devices are only used in mixed-speed bus systems and are not connected in an Hs-mode only system. In such cases, these pins can be used for other functions.

Optional series resistors R_s protect the I/O stages of the I²C-bus devices from high-voltage spikes on the bus lines and minimize ringing and interference.

Pull-up resistors R_p maintain the SDAH and SCLH lines at a HIGH level when the bus is free and ensure that the signals are pulled up from a LOW to a HIGH level within the required rise time. For higher capacitive bus-line loads (>100 pF), the resistor R_p can be replaced by external current source pull-ups to meet the rise time requirements. Unless

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

proceeded by an acknowledge bit, the rise time of the SCLH clock pulses in Hs-mode transfers is shortened by the internal current-source pull-up circuit MCS of the active master.

(1) SDA and SCL are not used here but may be used for other functions.

(2) To input filter.

(3) Only the active master can enable its current-source pull-up circuit.

(4) Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCLH.

Fig 32. I²C-bus configuration with Hs-mode devices only

5.3.2 Serial data format in Hs-mode

Serial data transfer format in Hs-mode meets the Standard-mode I²C-bus specification. Hs-mode can only commence after the following conditions (all of which are in F/S-mode):

1. START condition (S)
2. 8-bit master code (0000 1XXX)
3. Not-acknowledge bit ($\overline{A}$)

Figure 33 and Figure 34 show this in more detail. This master code has two main functions:

- It allows arbitration and synchronization between competing masters at F/S-mode speeds, resulting in one winning master.
- It indicates the beginning of an Hs-mode transfer.

Hs-mode master codes are reserved 8-bit codes, which are not used for slave addressing or other purposes. Furthermore, as each master has its own unique master code, up to eight Hs-mode masters can be present on the one I²C-bus system (although master code 0000 1000 should be reserved for test and diagnostic purposes). The master code for an Hs-mode master device is software programmable and is chosen by the System Designer.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Arbitration and clock synchronization only take place during the transmission of the master code and not-acknowledge bit ($\overline{A}$), after which one winning master remains active. The master code indicates to other devices that an Hs-mode transfer is to begin and the connected devices must meet the Hs-mode specification. As no device is allowed to acknowledge the master code, the master code is followed by a not-acknowledge ($\overline{A}$).

After the not-acknowledge bit ($\overline{A}$), and the SCLH line has been pulled-up to a HIGH level, the active master switches to Hs-mode and enables (at time t_H, see Figure 34) the current-source pull-up circuit for the SCLH signal. As other devices can delay the serial transfer before t_H by stretching the LOW period of the SCLH signal, the active master enables its current-source pull-up circuit when all devices have released the SCLH line and the SCLH signal has reached a HIGH level, thus speeding up the last part of the rise time of the SCLH signal.

The active master then sends a repeated START condition (Sr) followed by a 7-bit slave address (or 10-bit slave address, see Section 3.1.11) with a R/$\overline{W}$ bit address, and receives an acknowledge bit (A) from the selected slave.

After a repeated START condition and after each acknowledge bit (A) or not-acknowledge bit ($\overline{A}$), the active master disables its current-source pull-up circuit. This enables other devices to delay the serial transfer by stretching the LOW period of the SCLH signal. The active master re-enables its current-source pull-up circuit again when all devices have released and the SCLH signal reaches a HIGH level, and so speeds up the last part of the SCLH signal's rise time.

Data transfer continues in Hs-mode after the next repeated START (Sr), and only switches back to F/S-mode after a STOP condition (P). To reduce the overhead of the master code, it is possible that a master links a number of Hs-mode transfers, separated by repeated START conditions (Sr).

Fig 33. Data transfer format in Hs-mode

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 34. A complete Hs-mode transfer

5.3.3 Switching from F/S-mode to Hs-mode and back

After reset and initialization, Hs-mode devices must be in Fast-mode (which is in effect F/S-mode, as Fast-mode is downward compatible with Standard-mode). Each Hs-mode device can switch from Fast-mode to Hs-mode and back and is controlled by the serial transfer on the I²C-bus.

Before time t_1 in Figure 34, each connected device operates in Fast-mode. Between times t_1 and t_H (this time interval can be stretched by any device) each connected device must recognize the 'S 00001XXX A' sequence and has to switch its internal circuit from the Fast-mode setting to the Hs-mode setting. Between times t_1 and t_H, the connected master and slave devices perform this switching by the following actions.

The active (winning) master:

1. Adapts its SDAH and SCLH input filters according to the spike suppression requirement in Hs-mode.
2. Adapts the set-up and hold times according to the Hs-mode requirements.
3. Adapts the slope control of its SDAH and SCLH output stages according to the Hs-mode requirement.
4. Switches to the Hs-mode bit-rate, which is required after time t_H.
5. Enables the current source pull-up circuit of its SCLH output stage at time t_H.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

The non-active, or losing masters:

1. Adapt their SDAH and SCLH input filters according to the spike suppression requirement in Hs-mode.
2. Wait for a STOP condition to detect when the bus is free again.

All slaves:

1. Adapt their SDAH and SCLH input filters according to the spike suppression requirement in Hs-mode.
2. Adapt the set-up and hold times according to the Hs-mode requirements. This requirement may already be fulfilled by the adaptation of the input filters.
3. Adapt the slope control of their SDAH output stages, if necessary. For slave devices, slope control is applicable for the SDAH output stage only and, depending on circuit tolerances, both the Fast-mode and Hs-mode requirements may be fulfilled without switching its internal circuit.

At time t_{FS} in Figure 34, each connected device must recognize the STOP condition (P) and switch its internal circuit from the Hs-mode setting back to the Fast-mode setting as present before time t_1. This must be completed within the minimum bus free time as specified in Table 10 according to the Fast-mode specification.

5.3.4 Hs-mode devices at lower speed modes

Hs-mode devices are fully downwards compatible, and can be connected to an F/S-mode I²C-bus system (see Figure 35). As no master code is transmitted in such a configuration, all Hs-mode master devices stay in F/S-mode and communicate at F/S-mode speeds with their current-source disabled. The SDAH and SCLH pins are used to connect to the F/S-mode bus system, allowing the SDA and SCL pins (if present) on the Hs-mode master device to be used for other functions.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

(1) Bridge not used. SDA and SCL may have an alternative function.

(2) To input filter.

(3) The current-source pull-up circuit stays disabled.

(4) Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCL.

Fig 35. Hs-mode devices at F/S-mode speed

5.3.5 Mixed speed modes on one serial bus system

If a system has a combination of Hs-mode, Fast-mode and/or Standard-mode devices, it is possible, by using an interconnection bridge, to have different bit rates between different devices (see Figure 36 and Figure 37).

One bridge is required to connect/disconnect an Hs-mode section to/from an F/S-mode section at the appropriate time. This bridge includes a level shift function that allows devices with different supply voltages to be connected. For example F/S-mode devices with a V_{DD2} of 5 V can be connected to Hs-mode devices with a V_{DD1} of 3 V or less (that is, where $V_{DD2} \geq V_{DD1}$), provided SDA and SCL pins are 5 V tolerant. This bridge is incorporated in Hs-mode master devices and is completely controlled by the serial signals SDAH, SCLH, SDA and SCL. Such a bridge can be implemented in any IC as an autonomous circuit.

TR1, TR2 and TR3 are N-channel transistors. TR1 and TR2 have a transfer gate function, and TR3 is an open-drain pull-down stage. If TR1 or TR2 are switched on they transfer a LOW level in both directions, otherwise when both the drain and source rise to a HIGH level there is a high-impedance between the drain and source of each switched-on transistor. In the latter case, the transistors act as a level shifter as SDAH and SCLH are pulled-up to V_{DD1} and SDA and SCL are pulled-up to V_{DD2}.

During F/S-mode speed, a bridge on one of the Hs-mode masters connects the SDAH and SCLH lines to the corresponding SDA and SCL lines thus permitting Hs-mode devices to communicate with F/S-mode devices at slower speeds. Arbitration and synchronization are possible during the total F/S-mode transfer between all connected devices as described in Section 3.1.7. During Hs-mode transfer, however, the bridge

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

opens to separate the two bus sections and allows Hs-mode devices to communicate with each other at 3.4 Mbit/s. Arbitration between Hs-mode devices and F/S-mode devices is only performed during the master code (0000 1XXX), and normally won by one Hs-mode master as no slave address has four leading zeros. Other masters can win the arbitration only if they send a reserved 8-bit code (0000 0XXX). In such cases, the bridge remains closed and the transfer proceeds in F/S-mode. Table 8 gives the possible communication speeds in such a system.

(1) Bridge not used. SDA and SCL may have an alternative function.

(2) To input filter.

(3) Only the active master can enable its current-source pull-up circuit.

(4) Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCL or SCLH.

Fig 36. Bus system with transfer at Hs-mode and F/S-mode speeds

Table 8. Communication bit rates in a mixed-speed bus system

Transfer between	Serial bus system configuration			
	Hs + Fast + Standard	Hs + Fast	Hs + Standard	Fast + Standard
Hs ↔ Hs	0 to 3.4 Mbit/s	0 to 3.4 Mbit/s	0 to 3.4 Mbit/s	-
Hs ↔ Fast	0 to 100 kbit/s	0 to 400 kbit/s	-	-
Hs ↔ Standard	0 to 100 kbit/s	-	0 to 100 kbit/s	-
Fast ↔ Standard	0 to 100 kbit/s	-	-	0 to 100 kbit/s
Fast ↔ Fast	0 to 100 kbit/s	0 to 400 kbit/s	-	0 to 100 kbit/s
Standard ↔ Standard	0 to 100 kbit/s	-	0 to 100 kbit/s	0 to 100 kbit/s

Remark: Table 8 assumes that the Hs devices are isolated from the Fm and Sm devices when operating at 3.4 Mbit/s. The bus speed is always constrained to the maximum communication rate of the slowest device attached to the bus.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

5.3.6 Standard, Fast-mode and Fast-mode Plus transfer in a mixed-speed bus system

The bridge shown in Figure 36 interconnects corresponding serial bus lines, forming one serial bus system. As no master code (0000 1XXX) is transmitted, the current-source pull-up circuits stay disabled and all output stages are open-drain. All devices, including Hs-mode devices, communicate with each other according to the protocol, format and speed of the F/S-mode I^2C-bus specification.

5.3.7 Hs-mode transfer in a mixed-speed bus system

Figure 37 shows the timing diagram of a complete Hs-mode transfer, which is invoked by a START condition, a master code, and a not-acknowledge $\overline{A}$ (at F/S-mode speed). Although this timing diagram is split in two parts, it should be viewed as one timing diagram were time point t_H is a common point for both parts.

The master code is recognized by the bridge in the active or non-active master (see Figure 36). The bridge performs the following actions:

1. Between t_1 and t_H (see Figure 37), transistor TR1 opens to separate the SDAH and SDA lines, after which transistor TR3 closes to pull-down the SDA line to V_{SS}.
2. When both SCLH and SCL become HIGH (t_H in Figure 37), transistor TR2 opens to separate the SCLH and SCL lines. TR2 must be opened before SCLH goes LOW after Sr.

Hs-mode transfer starts after t_H with a repeated START condition (Sr). During Hs-mode transfer, the SCL line stays at a HIGH and the SDA line at a LOW steady-state level, and so is prepared for the transfer of a STOP condition (P).

After each acknowledge (A) or not-acknowledge bit ($\overline{A}$), the active master disables its current-source pull-up circuit. This enables other devices to delay the serial transfer by stretching the LOW period of the SCLH signal. The active master re-enables its current-source pull-up circuit again when all devices are released and the SCLH signal reaches a HIGH level, and so speeds up the last part of the SCLH signal rise time. In irregular situations, F/S-mode devices can close the bridge (TR1 and TR2 closed, TR3 open) at any time by pulling down the SCL line for at least 1 μs, for example, to recover from a bus hang-up.

Hs-mode finishes with a STOP condition and brings the bus system back into the F/S-mode. The active master disables its current-source MCS when the STOP condition (P) at SDAH is detected (t_{FS} in Figure 37). The bridge also recognizes this STOP condition and takes the following actions:

1. Transistor TR2 closes after t_{FS} to connect SCLH with SCL; both of which are HIGH at this time. Transistor TR3 opens after t_{FS}, which releases the SDA line and allows it to be pulled HIGH by the pull-up resistor R_p. This is the STOP condition for the F/S-mode devices. TR3 must open fast enough to ensure the bus free time between the STOP condition and the earliest next START condition is according to the Fast-mode specification (see t_{BUF} in Table 10).
2. When SDA reaches a HIGH (t_2 in Figure 37), transistor TR1 closes to connect SDAH with SDA. (Note: interconnections are made when all lines are HIGH, thus preventing spikes on the bus lines.) TR1 and TR2 must be closed within the minimum bus free time according to the Fast-mode specification (see t_{BUF} in Table 10).

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

Fig 37. A complete Hs-mode transfer in a mixed-speed bus system

5.3.8 Timing requirements for the bridge in a mixed-speed bus system

It can be seen from Figure 37 that the actions of the bridge at t_1, t_H and t_{FS} must be so fast that it does not affect the SDAH and SCLH lines. Furthermore the bridge must meet the related timing requirements of the Fast-mode specification for the SDA and SCL lines.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

5.4 Ultra Fast-mode

Ultra Fast-mode (UFm) devices offer an increase in I²C-bus transfer speeds. UFm devices can transfer information at bit rates of up to 5 Mbit/s. UFm devices offer push-pull drivers, eliminating the pull-up resistors, allowing higher transfer rates. The same serial bus protocol and data format is maintained as with the Sm, Fm, or Fm+ system. UFm bus devices are not compatible with bidirectional I²C-bus devices.

6. Electrical specifications and timing for I/O stages and bus lines

6.1 Standard-, Fast-, and Fast-mode Plus devices

The I/O levels, I/O current, spike suppression, output slope control and pin capacitance are given in Table 9. The I²C-bus timing characteristics, bus-line capacitance and noise margin are given in Table 10. Figure 38 shows the timing definitions for the I²C-bus.

The minimum HIGH and LOW periods of the SCL clock specified in Table 10 determine the maximum bit transfer rates of 100 kbit/s for Standard-mode devices, 400 kbit/s for Fast-mode devices, and 1000 kbit/s for Fast-mode Plus. Devices must be able to follow transfers at their own maximum bit rates, either by being able to transmit or receive at that speed or by applying the clock synchronization procedure described in Section 3.1.7 which forces the master into a wait state and stretch the LOW period of the SCL signal. In the latter case, the bit transfer rate is reduced.

All information provided in this document is subject to legal disclaimers.
© NXP Semiconductors N.V. 2014. All rights reserved.

Table 9. Characteristics of the SDA and SCL I/O stages

n/a = not applicable.

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
V_{IL}	LOW-level input voltage[1]		–0.5	$0.3V_{DD}$	–0.5	$0.3V_{DD}$	–0.5	$0.3V_{DD}$	V
V_{IH}	HIGH-level input voltage[1]		$0.7V_{DD}$	[2]	$0.7V_{DD}$	[2]	$0.7V_{DD}$[1]	[2]	V
V_{hys}	hysteresis of Schmitt trigger inputs		-	-	$0.05V_{DD}$	-	$0.05V_{DD}$	-	V
V_{OL1}	LOW-level output voltage 1	(open-drain or open-collector) at 3 mA sink current; V_{DD} > 2 V	0	0.4	0	0.4	0	0.4	V
V_{OL2}	LOW-level output voltage 2	(open-drain or open-collector) at 2 mA sink current[3]; $V_{DD} \leq 2$ V	-	-	0	$0.2V_{DD}$	0	$0.2V_{DD}$	V
I_{OL}	LOW-level output current	V_{OL} = 0.4 V	3	-	3	-	20	-	mA
		V_{OL} = 0.6 V[4]	-	-	6	-	-	-	mA
t_{of}	output fall time from V_{IHmin} to V_{ILmax}		-	250[5]	20 × (V_{DD} / 5.5 V)[6]	250[5]	20 × (V_{DD} / 5.5 V)[6]	120[7]	ns
t_{SP}	pulse width of spikes that must be suppressed by the input filter		-	-	0	50[8]	0	50[8]	ns
I_i	input current each I/O pin	$0.1V_{DD} < V_I < 0.9V_{DDmax}$	–10	+10	–10[9]	+10[9]	–10[9]	+10[9]	µA
C_i	capacitance for each I/O pin[10]		-	10	-	10	-	10	pF

[1] Some legacy Standard-mode devices had fixed input levels of V_{IL} = 1.5 V and V_{IH} = 3.0 V. Refer to component data sheets.

[2] Maximum $V_{IH} = V_{DD(max)}$ + 0.5 V or 5.5 V, which ever is lower. See component data sheets.

[3] The same resistor value to drive 3 mA at 3.0 V V_{DD} provides the same RC time constant when using <2 V V_{DD} with a smaller current draw.

[4] In order to drive full bus load at 400 kHz, 6 mA I_{OL} is required at 0.6 V V_{OL}. Parts not meeting this specification can still function, but not at 400 kHz and 400 pF.

[5] The maximum t_f for the SDA and SCL bus lines quoted in Table 10 (300 ns) is longer than the specified maximum t_{of} for the output stages (250 ns). This allows series protection resistors (R_s) to be connected between the SDA/SCL pins and the SDA/SCL bus lines as shown in Figure 45 without exceeding the maximum specified t_f.

[6] Necessary to be backwards compatible with Fast-mode.

[7] In Fast-mode Plus, fall time is specified the same for both output stage and bus timing. If series resistors are used, designers should allow for this when considering bus timing.

[8] Input filters on the SDA and SCL inputs suppress noise spikes of less than 50 ns.

[9] If V_{DD} is switched off, I/O pins of Fast-mode and Fast-mode Plus devices must not obstruct the SDA and SCL lines.

[10] Special purpose devices such as multiplexers and switches may exceed this capacitance because they connect multiple paths together.

All information provided in this document is subject to legal disclaimers.
© NXP Semiconductors N.V. 2014. All rights reserved.

Table 10. Characteristics of the SDA and SCL bus lines for Standard, Fast, and Fast-mode Plus I²C-bus devices[1]

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency		0	100	0	400	0	1000	kHz
$t_{HD;STA}$	hold time (repeated) START condition	After this period, the first clock pulse is generated.	4.0	-	0.6	-	0.26	-	µs
t_{LOW}	LOW period of the SCL clock		4.7	-	1.3	-	0.5	-	µs
t_{HIGH}	HIGH period of the SCL clock		4.0	-	0.6	-	0.26	-	µs
$t_{SU;STA}$	set-up time for a repeated START condition		4.7	-	0.6	-	0.26	-	µs
$t_{HD;DAT}$	data hold time[2]	CBUS compatible masters (see Remark in Section 4.1)	5.0	-	-	-	-	-	µs
		I²C-bus devices	0[3]	-[4]	0[3]	-[4]	0	-	µs
$t_{SU;DAT}$	data set-up time		250	-	100[5]	-	50	-	ns
t_r	rise time of both SDA and SCL signals		-	1000	20	300	-	120	ns
t_f	fall time of both SDA and SCL signals[3][6][7][8]		-	300	$20 \times (V_{DD} / 5.5\ V)$	300	$20 \times (V_{DD} / 5.5\ V)$[9]	120[8]	ns
$t_{SU;STO}$	set-up time for STOP condition		4.0	-	0.6	-	0.26	-	µs
t_{BUF}	bus free time between a STOP and START condition		4.7	-	1.3	-	0.5	-	µs
C_b	capacitive load for each bus line[10]		-	400	-	400	-	550	pF
$t_{VD;DAT}$	data valid time[11]		-	3.45[4]	-	0.9[4]	-	0.45[4]	µs
$t_{VD;ACK}$	data valid acknowledge time[12]		-	3.45[4]	-	0.9[4]	-	0.45[4]	µs
V_{nL}	noise margin at the LOW level	for each connected device (including hysteresis)	$0.1V_{DD}$	-	$0.1V_{DD}$	-	$0.1V_{DD}$	-	V
V_{nH}	noise margin at the HIGH level	for each connected device (including hysteresis)	$0.2V_{DD}$	-	$0.2V_{DD}$	-	$0.2V_{DD}$	-	V

[1] All values referred to $V_{IH(min)}$ ($0.3V_{DD}$) and $V_{IL(max)}$ ($0.7V_{DD}$) levels (see Table 9).

[2] $t_{HD;DAT}$ is the data hold time that is measured from the falling edge of SCL, applies to data in transmission and the acknowledge.

[3] A device must internally provide a hold time of at least 300 ns for the SDA signal (with respect to the $V_{IH(min)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

[4] The maximum $t_{HD;DAT}$ could be 3.45 µs and 0.9 µs for Standard-mode and Fast-mode, but must be less than the maximum of $t_{VD;DAT}$ or $t_{VD;ACK}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

UM10204

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

[5] A Fast-mode I²C-bus device can be used in a Standard-mode I²C-bus system, but the requirement $t_{SU;DAT}$ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line $t_{r(max)} + t_{SU;DAT} = 1000 + 250 = 1250$ ns (according to the Standard-mode I²C-bus specification) before the SCL line is released. Also the acknowledge timing must meet this set-up time.

[6] If mixed with Hs-mode devices, faster fall times according to Table 10 are allowed.

[7] The maximum t_f for the SDA and SCL bus lines is specified at 300 ns. The maximum fall time for the SDA output stage t_f is specified at 250 ns. This allows series protection resistors to be connected in between the SDA and the SCL pins and the SDA/SCL bus lines without exceeding the maximum specified t_f.

[8] In Fast-mode Plus, fall time is specified the same for both output stage and bus timing. If series resistors are used, designers should allow for this when considering bus timing.

[9] Necessary to be backwards compatible to Fast-mode.

[10] The maximum bus capacitance allowable may vary from this value depending on the actual operating voltage and frequency of the application. Section 7.2 discusses techniques for coping with higher bus capacitances.

[11] $t_{VD;DAT}$ = time for data signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).

[12] $t_{VD;ACK}$ = time for Acknowledgement signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 38. Definition of timing for F/S-mode devices on the I²C-bus

6.2 Hs-mode devices

The I/O levels, I/O current, spike suppression, output slope control and pin capacitance for I²C-bus Hs-mode devices are given in Table 11. The noise margin for HIGH and LOW levels on the bus lines are the same as specified for F/S-mode I²C-bus devices.

Figure 39 shows all timing parameters for the Hs-mode timing. The 'normal' START condition S does not exist in Hs-mode. Timing parameters for Address bits, R/$\overline{W}$ bit, Acknowledge bit and DATA bits are all the same. Only the rising edge of the first SCLH clock signal after an acknowledge bit has a larger value because the external R_p has to pull up SCLH without the help of the internal current-source.

The Hs-mode timing parameters for the bus lines are specified in Table 12. The minimum HIGH and LOW periods and the maximum rise and fall times of the SCLH clock signal determine the highest bit rate.

With an internally generated SCLH signal with LOW and HIGH level periods of 200 ns and 100 ns respectively, an Hs-mode master fulfills the timing requirements for the external SCLH clock pulses (taking the rise and fall times into account) for the maximum bit rate of 3.4 Mbit/s. So a basic frequency of 10 MHz, or a multiple of 10 MHz, can be used by an Hs-mode master to generate the SCLH signal. There are no limits for maximum HIGH and LOW periods of the SCLH clock, and there is no limit for a lowest bit rate.

Timing parameters are independent for capacitive load up to 100 pF for each bus line allowing the maximum possible bit rate of 3.4 Mbit/s. At a higher capacitive load on the bus lines, the bit rate decreases gradually. The timing parameters for a capacitive bus load of 400 pF are specified in Table 12, allowing a maximum bit rate of 1.7 Mbit/s. For

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

capacitive bus loads between 100 pF and 400 pF, the timing parameters must be interpolated linearly. Rise and fall times are in accordance with the maximum propagation time of the transmission lines SDAH and SCLH to prevent reflections of the open ends.

Table 11. Characteristics of the SDAH, SCLH, SDA and SCL I/O stages for Hs-mode I²C-bus devices

Symbol	Parameter	Conditions	Hs-mode		Unit
			Min	Max	
V_{IL}	LOW-level input voltage		–0.5	$0.3V_{DD}$[1]	V
V_{IH}	HIGH-level input voltage		$0.7V_{DD}$[1]	V_{DD} + 0.5[2]	V
V_{hys}	hysteresis of Schmitt trigger inputs		$0.1V_{DD}$[1]	-	V
V_{OL}	LOW-level output voltage	(open-drain) at 3 mA sink current at SDAH, SDA and SCLH			
		V_{DD} > 2 V	0	0.4	V
		V_{DD} ≤ 2 V	0	$0.2V_{DD}$	V
R_{onL}	transfer gate on resistance for currents between SDA and SDAH or SCL and SCLH	V_{OL} level; I_{OL} = 3 mA	-	50	Ω
R_{onH}[2]	transfer gate on resistance between SDA and SDAH or SCL and SCLH	both signals (SDA and SDAH, or SCL and SCLH) at V_{DD} level	50	-	kΩ
I_{CS}	pull-up current of the SCLH current-source	SCLH output levels between $0.3V_{DD}$ and $0.7V_{DD}$	3	12	mA
t_{rCL}	rise time of SCLH signal	output rise time (current-source enabled) with an external pull-up current source of 3 mA			
		capacitive load from 10 pF to 100 pF	10	40	ns
		capacitive load of 400 pF[3]	20	80	ns
t_{fCL}	fall time of SCLH signal	output fall time (current-source enabled) with an external pull-up current source of 3 mA			
		capacitive load from 10 pF to 100 pF	10	40	ns
		capacitive load of 400 pF[3]	20	80	ns
t_{fDA}	fall time of SDAH signal	capacitive load from 10 pF to 100 pF	10	80	ns
		capacitive load of 400 pF[3]	20	160	ns
t_{SP}	pulse width of spikes that must be suppressed by the input filter	SDAH and SCLH	0	10	ns
I_i[4]	input current each I/O pin	input voltage between $0.1V_{DD}$ and $0.9V_{DD}$	-	10	μA
C_i	capacitance for each I/O pin[5]		-	10	pF

[1] Devices that use non-standard supply voltages which do not conform to the intended I²C-bus system levels must relate their input levels to the V_{DD} voltage to which the pull-up resistors R_p are connected.

[2] Devices that offer the level shift function must tolerate a maximum input voltage of 5.5 V at SDA and SCL.

[3] For capacitive bus loads between 100 pF and 400 pF, the rise and fall time values must be linearly interpolated.

[4] If their supply voltage has been switched off, SDAH and SCLH I/O stages of Hs-mode slave devices must have floating outputs. Due to the current-source output circuit, which normally has a clipping diode to V_{DD}, this requirement is not mandatory for the SCLH or the SDAH I/O stage of Hs-mode master devices. This means that the supply voltage of Hs-mode master devices cannot be switched off without affecting the SDAH and SCLH lines.

[5] Special purpose devices such as multiplexers and switches may exceed this capacitance because they connect multiple paths together.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Table 12. Characteristics of the SDAH, SCLH, SDA and SCL bus lines for Hs-mode I²C-bus devices[1]

Symbol	Parameter	Conditions	C_b = 100 pF (max)		C_b = 400 pF[2]		Unit
			Min	Max	Min	Max	
f_{SCLH}	SCLH clock frequency		0	3.4	0	1.7	MHz
$t_{SU;STA}$	set-up time for a repeated START condition		160	-	160	-	ns
$t_{HD;STA}$	hold time (repeated) START condition		160	-	160	-	ns
t_{LOW}	LOW period of the SCL clock		160	-	320	-	ns
t_{HIGH}	HIGH period of the SCL clock		60	-	120	-	ns
$t_{SU;DAT}$	data set-up time		10	-	10	-	ns
$t_{HD;DAT}$	data hold time		0[3]	70	0[3]	150	ns
t_{rCL}	rise time of SCLH signal		10	40	20	80	ns
t_{rCL1}	rise time of SCLH signal after a repeated START condition and after an acknowledge bit		10	80	20	160	ns
t_{fCL}	fall time of SCLH signal		10	40	20	80	ns
t_{rDA}	rise time of SDAH signal		10	80	20	160	ns
t_{fDA}	fall time of SDAH signal		10	80	20	160	ns
$t_{SU;STO}$	set-up time for STOP condition		160	-	160	-	ns
C_b[2]	capacitive load for each bus line	SDAH and SCLH lines	-	100	-	400	pF
		SDAH + SDA line and SCLH + SCL line	-	400	-	400	pF
V_{nL}	noise margin at the LOW level	for each connected device (including hysteresis)	$0.1V_{DD}$	-	$0.1V_{DD}$	-	V
V_{nH}	noise margin at the HIGH level	for each connected device (including hysteresis)	$0.2V_{DD}$	-	$0.2V_{DD}$	-	V

[1] All values referred to $V_{IH(min)}$ and $V_{IL(max)}$ levels (see Table 11).

[2] For bus line loads C_b between 100 pF and 400 pF the timing parameters must be linearly interpolated.

[3] A device must internally provide a data hold time to bridge the undefined part between V_{IH} and V_{IL} of the falling edge of the SCLH signal. An input circuit with a threshold as low as possible for the falling edge of the SCLH signal minimizes this hold time.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

(1) First rising edge of the SCLH signal after Sr and after each acknowledge bit.

Fig 39. Definition of timing for Hs-mode devices on the I²C-bus

6.3 Ultra Fast-mode devices

The I/O levels, I/O current, spike suppression, output slope control and pin capacitance are given in Table 13. The UFm I²C-bus timing characteristics are given in Table 14. Figure 40 shows the timing definitions for the I²C-bus. The minimum HIGH and LOW periods of the SCL clock specified in Table 14 determine the maximum bit transfer rates of 5000 kbit/s for Ultra Fast-mode. Devices must be able to follow transfers at their own maximum bit rates, either by being able to transmit or receive at that speed.

Table 13. Characteristics of the USDA and USCL I/O stages

n/a = not applicable.

Symbol	Parameter	Conditions	Ultra Fast-mode		Unit
			Min	Max	
V_{IL}	LOW-level input voltage[1]		–0.5	+0.3V_{DD}	V
V_{IH}	HIGH-level input voltage[1]		0.7V_{DD}[1]	-[2]	V
V_{hys}	hysteresis of Schmitt trigger inputs		0.05V_{DD}	-	V
V_{OL}	LOW-level output voltage	at 4 mA sink current; V_{DD} > 2 V	0	0.4	V
V_{OH}	HIGH-level output voltage	at 4 mA source current; V_{DD} > 2 V	V_{DD} – 0.4	-	V
I_L	leakage current	V_{DD} = 3.6 V	–1	+1	µA
		V_{DD} = 5.5 V	–10	+10	µA
C_i	input capacitance	[3]	-	10	pF
t_{SP}	pulse width of spikes that must be suppressed by the input filter	[4]	-	10	ns

[1] Refer to component data sheets for actual switching points.

[2] Maximum V_{IH} = $V_{DD(max)}$ + 0.5 V or 5.5 V, whichever is lower. See component data sheets.

[3] Special purpose devices such as multiplexers and switches may exceed this capacitance because they connect multiple paths together.

[4] Input filters on the USDA and USCL slave inputs suppress noise spikes of less than 10 ns.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Table 14. UFm I²C-bus frequency and timing specifications

Symbol	Parameter	Conditions	Ultra Fast-mode		Unit
			Min	Max	
f_{USCL}	USCL clock frequency		0	5000	kHz
t_{BUF}	bus free time between a STOP and START condition		80	-	ns
$t_{HD;STA}$	hold time (repeated) START condition		50	-	ns
$t_{SU;STA}$	set-up time for a repeated START condition		50	-	ns
$t_{SU;STO}$	set-up time for STOP condition		50	-	ns
$t_{HD;DAT}$	data hold time		10	-	ns
$t_{VD;DAT}$	data valid time	[1]	10	-	ns
$t_{SU;DAT}$	data set-up time		30	-	ns
t_{LOW}	LOW period of the USCL clock		50	-	ns
t_{HIGH}	HIGH period of the USCL clock		50	-	ns
t_f	fall time of both USDA and USCL signals		-[2]	50	ns
t_r	rise time of both USDA and USCL signals		-[2]	50	ns

[1] $t_{VD;DAT}$ = minimum time for USDA data out to be valid following USCL LOW.

[2] Typical rise time or fall time for UFm signals is 25 ns measured from the 30 % level to the 70 % level (rise time) or from the 70 % level to the 30 % level (fall time).

Fig 40. Definition of timing for Ultra Fast-mode devices on the I²C-bus

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

7. Electrical connections of I²C-bus devices to the bus lines

7.1 Pull-up resistor sizing

The bus capacitance is the total capacitance of wire, connections and pins. This capacitance limits the maximum value of R_p due to the specified rise time. Figure 41 shows $R_{p(max)}$ as a function of bus capacitance.

Consider the V_{DD} related input threshold of $V_{IH} = 0.7V_{DD}$ and $V_{IL} = 0.3V_{DD}$ for the purposes of RC time constant calculation. Then $V(t) = V_{DD}\ (1 - e^{-t\ /\ RC})$, where t is the time since the charging started and RC is the time constant.

$V(t1) = 0.3 \times V_{DD} = V_{DD}\ (1 - e^{-t1\ /\ RC})$; then $t1 = 0.3566749 \times RC$

$V(t2) = 0.7 \times V_{DD} = V_{DD}\ (1 - e^{-t2\ /\ RC})$; then $t2 = 1.2039729 \times RC$

$T = t2 - t1 = 0.8473 \times RC$

Figure 41 and Equation 1 shows maximum R_p as a function of bus capacitance for Standard-, Fast- and Fast-mode Plus. For each mode, the $R_{p(max)}$ is a function of the rise time maximum (t_r) from Table 10 and the estimated bus capacitance (C_b):

$$R_{p(max)} = \frac{t_r}{0.8473 \times C_b} \quad (1)$$

(1) Standard-mode
(2) Fast-mode
(3) Fast-mode Plus

Fig 41. $R_{p(max)}$ as a function of bus capacitance

(1) Fast-mode and Standard-mode
(2) Fast-mode Plus

Fig 42. $R_{p(min)}$ as a function of V_{DD}

The supply voltage limits the minimum value of resistor R_p due to the specified minimum sink current of 3 mA for Standard-mode and Fast-mode, or 20 mA for Fast-mode Plus. $R_{p(min)}$ as a function of V_{DD} is shown in Figure 42. The traces are calculated using Equation 2:

$$R_{p(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL}} \quad (2)$$

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

The designer now has the minimum and maximum value of R_p that is required to meet the timing specification. Portable designs with sensitivity to supply current consumption can use a value toward the higher end of the range in order to limit I_{DD}.

7.2 Operating above the maximum allowable bus capacitance

Bus capacitance limit is specified to limit rise time reductions and allow operating at the rated frequency. While most designs can easily stay within this limit, some applications may exceed it. There are several strategies available to system designers to cope with excess bus capacitance.

- Reduced f_{SCL} (Section 7.2.1): The bus may be operated at a lower speed (lower f_{SCL}).
- Higher drive outputs (Section 7.2.2): Devices with higher drive current such as those rated for Fast-mode Plus can be used (PCA96xx).
- Bus buffers (Section 7.2.3): There are a number of bus buffer devices available that can divide the bus into segments so that each segment has a capacitance below the allowable limit, such as the PCA9517 bus buffer or the PCA9546A switch.
- Switched pull-up circuit (Section 7.2.4): A switched pull-up circuit can be used to accelerate rising edges by switching a low value pull-up alternately in and out when needed.

7.2.1 Reduced f_{SCL}

To determine a lower allowable bus operating frequency, begin by finding the t_{LOW} and t_{HIGH} of the most limiting device on the bus. Refer to individual component data sheets for these values. Actual rise time (t_r) depends on the RC time constant. The most limiting fall time (t_f) depends on the lowest output drive on the bus. Be sure to allow for any devices that have a minimum t_r or t_f. Refer to Equation 3 for the resulting f_{max}.

$$f_{max} = \frac{1}{t_{LOW(min)} + t_{HIGH(min)} + t_{r(actual)} + t_{f(actual)}} \quad (3)$$

Remark: Very long buses must also account for time of flight of signals.

Actual results are slower, as real parts do not tend to control t_{LOW} and t_{HIGH} to the minimum from 30 % to 70 %, or 70 % to 30 %, respectively.

7.2.2 Higher drive outputs

If higher drive devices like the PCA96xx Fast-mode Plus or the P82B bus buffers are used, the higher strength output drivers sink more current which results in considerably faster edge rates, or, looked at another way, allows a higher bus capacitance. Refer to individual component data sheets for actual output drive capability. Repeat the calculation above using the new values of C_b, R_p, t_r and t_f to determine maximum frequency. Bear in mind that the maximum rating for f_{SCL} as specified in Table 10 (100 kHz, 400 kHz and 1000 kHz) may become limiting.

7.2.3 Bus buffers, multiplexers and switches

Another approach to coping with excess bus capacitance is to divide the bus into smaller segments using bus buffers, multiplexers or switches. Figure 43 shows an example of a bus that uses a PCA9515 buffer to deal with high bus capacitance. Each segment is then allowed to have the maximum capacitance so the total bus can have twice the maximum

All information provided in this document is subject to legal disclaimers. © NXP Semiconductors N.V. 2014. All rights reserved.

capacitance. Keep in mind that adding a buffer always adds delays — a buffer delay plus an additional transition time to each edge, which reduces the maximum operating frequency and may also introduce special V_{IL} and V_{OL} considerations.

Refer to application notes *AN255, I²C / SMBus Repeaters, Hubs and Expanders* and *AN262, PCA954x Family of I²C / SMBus Multiplexers and Switches* for more details on this subject and the devices available from NXP Semiconductors.

Remark: Some buffers allow V_{DD1} and V_{DD2} to be different levels.

Fig 43. Using a buffer to divide bus capacitance

7.2.4 Switched pull-up circuit

The supply voltage (V_{DD}) and the maximum output LOW level determine the minimum value of pull-up resistor R_p (see Section 7.1). For example, with a supply voltage of V_{DD} = 5 V ± 10 % and $V_{OL(max)}$ = 0.4 V at 3 mA, $R_{p(min)}$ = (5.5 – 0.4) / 0.003 = 1.7 kΩ. As shown in Figure 42, this value of R_p limits the maximum bus capacitance to about 200 pF to meet the maximum t_r requirement of 300 ns. If the bus has a higher capacitance than this, a switched pull-up circuit (as shown in Figure 44) can be used.

Fig 44. Switched pull-up circuit

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

The switched pull-up circuit in Figure 44 is for a supply voltage of V_{DD} = 5 V ± 10 % and a maximum capacitive load of 400 pF. Since it is controlled by the bus levels, it needs no additional switching control signals. During the rising/falling edges, the bilateral switch in the HCT4066 switches pull-up resistor R_{p2} on/off at bus levels between 0.8 V and 2.0 V. Combined resistors R_{p1} and R_{p2} can pull up the bus line within the maximum specified rise time (t_r) of 300 ns.

Series resistors R_s are optional. They protect the I/O stages of the I²C-bus devices from high-voltage spikes on the bus lines, and minimize crosstalk and undershoot of the bus line signals. The maximum value of R_s is determined by the maximum permitted voltage drop across this resistor when the bus line is switched to the LOW level in order to switch off R_{p2}.

Additionally, some bus buffers contain integral rise time accelerators. Stand-alone rise time accelerators are also available.

7.3 Series protection resistors

As shown in Figure 45, series resistors (R_s) of, for example, 300 Ω can be used for protection against high-voltage spikes on the SDA and SCL lines (resulting from the flash-over of a TV picture tube, for example). If series resistors are used, designers must add the additional resistance into their calculations for R_p and allowable bus capacitance.

Fig 45. Series resistors (R_s) for protection against high-voltage spikes

The required noise margin of $0.1V_{DD}$ for the LOW level, limits the maximum value of R_s. $R_{s(max)}$ as a function of R_p is shown in Figure 46. Note that series resistors affect the output fall time.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

Fig 46. Maximum value of R_s as a function of the value of R_p with supply voltage as a parameter

7.4 Input leakage

The maximum HIGH level input current of each input/output connection has a specified maximum value of 10 μA. Due to the required noise margin of $0.2V_{DD}$ for the HIGH level, this input current limits the maximum value of R_p. This limit depends on V_{DD}. The total HIGH-level input current is shown as a function of $R_{p(max)}$ in Figure 47.

Fig 47. Total HIGH-level input current as a function of the maximum value of R_p with supply voltage as a parameter

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

7.5 Wiring pattern of the bus lines

In general, the wiring must be chosen so that crosstalk and interference to/from the bus lines is minimized. The bus lines are most susceptible to crosstalk and interference at the HIGH level because of the relatively high impedance of the pull-up devices.

If the length of the bus lines on a PCB or ribbon cable exceeds 10 cm and includes the V_{DD} and V_{SS} lines, the wiring pattern should be:

SDA _____

V_{DD} _____

V_{SS} _____

SCL _____

If only the V_{SS} line is included, the wiring pattern should be:

SDA _____

V_{SS} _____

SCL _____

These wiring patterns also result in identical capacitive loads for the SDA and SCL lines. If a PCB with a V_{SS} and/or V_{DD} layer is used, the V_{SS} and V_{DD} lines can be omitted.

If the bus lines are twisted-pairs, each bus line must be twisted with a V_{SS} return. Alternatively, the SCL line can be twisted with a V_{SS} return, and the SDA line twisted with a V_{DD} return. In the latter case, capacitors must be used to decouple the V_{DD} line to the V_{SS} line at both ends of the twisted pairs.

If the bus lines are shielded (shield connected to V_{SS}), interference is minimized. However, the shielded cable must have low capacitive coupling between the SDA and SCL lines to minimize crosstalk.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

8. Abbreviations

Table 15. Abbreviations

Acronym	Description
A/D	Analog-to-Digital
ATCA	Advanced Telecom Computing Architecture
BMC	Baseboard Management Controller
CMOS	Complementary Metal-Oxide Semiconductor
cPCI	compact Peripheral Component Interconnect
D/A	Digital-to-Analog
DIP	Dual In-line Package
EEPROM	Electrically Erasable Programmable Read Only Memory
HW	Hardware
I/O	Input/Output
I^2C-bus	Inter-Integrated Circuit bus
IC	Integrated Circuit
IPMI	Intelligent Platform Management Interface
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least Significant Bit
MCU	Microcontroller
MSB	Most Significant Bit
NMOS	Negative-channel Metal-Oxide Semiconductor
PCB	Printed-Circuit Board
PCI	Peripheral Component Interconnect
PMBus	Power Management Bus
RAM	Random Access Memory
ROM	Read-Only Memory
SMBus	System Management Bus
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

9. Legal information

9.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

9.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I²C-bus — logo is a trademark of NXP Semiconductors N.V.

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

10. Contents

1 Introduction . . . 3
2 I²C-bus features . . . 3
2.1 Designer benefits . . . 4
2.2 Manufacturer benefits . . . 5
2.3 IC designer benefits . . . 6
3 The I²C-bus protocol . . . 6
3.1 Standard-mode, Fast-mode and Fast-mode Plus I²C-bus protocols . . . 6
3.1.1 SDA and SCL signals . . . 8
3.1.2 SDA and SCL logic levels . . . 9
3.1.3 Data validity . . . 9
3.1.4 START and STOP conditions . . . 9
3.1.5 Byte format . . . 10
3.1.6 Acknowledge (ACK) and Not Acknowledge (NACK) . . . 10
3.1.7 Clock synchronization . . . 11
3.1.8 Arbitration . . . 11
3.1.9 Clock stretching . . . 13
3.1.10 The slave address and R/$\overline{W}$ bit . . . 13
3.1.11 10-bit addressing . . . 15
3.1.12 Reserved addresses . . . 17
3.1.13 General call address . . . 17
3.1.14 Software reset . . . 19
3.1.15 START byte . . . 19
3.1.16 Bus clear . . . 20
3.1.17 Device ID . . . 20
3.2 Ultra Fast-mode I²C-bus protocol . . . 23
3.2.1 USDA and USCL signals . . . 25
3.2.2 USDA and USCL logic levels . . . 25
3.2.3 Data validity . . . 25
3.2.4 START and STOP conditions . . . 25
3.2.5 Byte format . . . 26
3.2.6 Acknowledge (ACK) and Not Acknowledge (NACK) . . . 27
3.2.7 The slave address and R/$\overline{W}$ bit . . . 27
3.2.8 10-bit addressing . . . 28
3.2.9 Reserved addresses in UFm . . . 29
3.2.10 General call address . . . 30
3.2.11 Software reset . . . 30
3.2.12 START byte . . . 30
3.2.13 Unresponsive slave reset . . . 31
3.2.14 Device ID . . . 31
4 Other uses of the I²C-bus communications protocol . . . 32
4.1 CBUS compatibility . . . 32
4.2 SMBus - System Management Bus . . . 32
4.2.1 I²C/SMBus compliancy . . . 32
4.2.2 Time-out feature . . . 33
4.2.3 Differences between SMBus 1.0 and SMBus 2.0 . . . 33
4.3 PMBus - Power Management Bus . . . 34
4.4 Intelligent Platform Management Interface (IPMI) . . . 34
4.5 Advanced Telecom Computing Architecture (ATCA) . . . 35
4.6 Display Data Channel (DDC) . . . 35
5 Bus speeds . . . 35
5.1 Fast-mode . . . 36
5.2 Fast-mode Plus . . . 36
5.3 Hs-mode . . . 37
5.3.1 High speed transfer . . . 37
5.3.2 Serial data format in Hs-mode . . . 38
5.3.3 Switching from F/S-mode to Hs-mode and back . . . 40
5.3.4 Hs-mode devices at lower speed modes . . . 41
5.3.5 Mixed speed modes on one serial bus system . . . 42
5.3.6 Standard, Fast-mode and Fast-mode Plus transfer in a mixed-speed bus system . . . 44
5.3.7 Hs-mode transfer in a mixed-speed bus system . . . 44
5.3.8 Timing requirements for the bridge in a mixed-speed bus system . . . 45
5.4 Ultra Fast-mode . . . 46
6 Electrical specifications and timing for I/O stages and bus lines . . . 46
6.1 Standard-, Fast-, and Fast-mode Plus devices . . . 46
6.2 Hs-mode devices . . . 50
6.3 Ultra Fast-mode devices . . . 53
7 Electrical connections of I²C-bus devices to the bus lines . . . 55
7.1 Pull-up resistor sizing . . . 55
7.2 Operating above the maximum allowable bus capacitance . . . 56
7.2.1 Reduced f_{SCL} . . . 56
7.2.2 Higher drive outputs . . . 56
7.2.3 Bus buffers, multiplexers and switches . . . 56
7.2.4 Switched pull-up circuit . . . 57
7.3 Series protection resistors . . . 58
7.4 Input leakage . . . 59
7.5 Wiring pattern of the bus lines . . . 60
8 Abbreviations . . . 61

continued >>

All information provided in this document is subject to legal disclaimers.

© NXP Semiconductors N.V. 2014. All rights reserved.

9 Legal information 62
9.1 Definitions 62
9.2 Disclaimers 62
9.3 Trademarks 62
10 Contents 63

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP Semiconductors N.V. 2014. All rights reserved.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 4 April 2014
Document identifier: UM10204

I²S bus specification

1.0 INTRODUCTION

Many digital audio systems are being introduced into the consumer audio market, including compact disc, digital audio tape, digital sound processors, and digital TV-sound. The digital audio signals in these systems are being processed by a number of (V)LSI ICs, such as:

- A/D and D/A converters;
- digital signal processors;
- error correction for compact disc and digital recording;
- digital filters;
- digital input/output interfaces.

Standardized communication structures are vital for both the equipment and the IC manufacturer, because they increase system flexibility. To this end, we have developed the inter-IC sound (I^2S) bus – a serial link especially for digital audio.

2.0 BASIC SERIAL BUS REQUIREMENTS

The bus has only to handle audio data, while the other signals, such as sub-coding and control, are transferred separately. To minimize the number of pins required and to keep wiring simple, a 3-line serial bus is used consisting of a line for two time-multiplexed data channels, a word select line and a clock line.

Since the transmitter and receiver have the same clock signal for data transmission, the transmitter as the master, has to generate the bit clock, word-select signal and data. In complex systems however, there may be several transmitters and receivers, which makes it difficult to define the master. In such systems, there is usually a system master controlling digital audio data-flow between the various ICs. Transmitters then, have to generate data under the control of an external clock, and so act as a slave. Figure 1 illustrates some simple system configurations and the basic interface timing. Note that the system master can be combined with a transmitter or receiver, and it may be enabled or disabled under software control or by pin programming.

Figure 1. Simple System Configurations and Basic Interface Timing

3.0 THE I²S BUS

As shown in Figure 1, the bus has three lines:

- continuous serial clock (SCK);
- word select (WS);
- serial data (SD);

and the device generating SCK and WS is the master.

3.1 Serial Data

Serial data is transmitted in two's complement with the MSB first. The MSB is transmitted first because the transmitter and receiver may have different word lengths. It isn't necessary for the transmitter to know how many bits the receiver can handle, nor does the receiver need to know how many bits are being transmitted.

When the system word length is greater than the transmitter word length, the word is truncated (least significant data bits are set to '0') for data transmission. If the receiver is sent more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver is sent fewer bits than its word length, the missing bits are set to zero internally. And so, the MSB has a fixed position, whereas the position of the LSB depends on the word length. The transmitter always sends the MSB of the next word one clock period after the WS changes.

Serial data sent by the transmitter may be synchronized with either the trailing (HIGH-to-LOW) or the leading (LOW-to-HIGH) edge of the clock signal. However, the serial data must be latched into the receiver on the leading edge of the serial clock signal, and so there are some restrictions when transmitting data that is synchronized with the leading edge (see Figure 2 and Table 1).

3.2 Word Select

The word select line indicates the channel being transmitted:

- WS = 0; channel 1 (left);
- WS = 1; channel 2 (right).

WS may change either on a trailing or leading edge of the serial clock, but it doesn't need to be symmetrical. In the slave, this signal is latched on the leading edge of the clock signal. The WS line changes one clock period before the MSB is transmitted. This allows the slave transmitter to derive synchronous timing of the serial data that will be set up for transmission. Furthermore, it enables the receiver to store the previous word and clear the input for the next word (see Figure 1).

4.0 TIMING

In the I²S format, any device can act as the system master by providing the necessary clock signals. A slave will usually derive its internal clock signal from an external clock input. This means, taking into account the propagation delays between master clock and the data and/or word-select signals, that the total delay is simply the sum of:

- the delay between the external (master) clock and the slave's internal clock; and
- the delay between the internal clock and the data and/or word-select signals.

For data and word-select inputs, the external to internal clock delay is of no consequence because it only lengthens the effective set-up time (see Figure 2). The major part of the time margin is to accommodate the difference between the propagation delay of the transmitter, and the time required to set up the receiver.

All timing requirements are specified relative to the clock period or to the minimum allowed clock period of a device. This means that higher data rates can be used in the future.

T = clock period
T_{tr} = minimum allowed clock period for transmitter
T > T_{tr}
* t_{RC} is only relevant for transmitters in slave mode.

SN00120

Figure 2. Timing for I²S Transmitter

T = clock period
T_r = minimum allowed clock period for transmitter
T > T_r

SN00121

Figure 3. Timing for I²S Receiver

Note that the times given in both Figures 2 and 3 are defined by the transmitter speed. The specification of the receiver has to be able to match the performance of the transmitter

Example: Master transmitter with data rate of 2.5MHz (±10%) (all values in ns)

	MIN	TYP	MAX	CONDITION
clock period T	360	400	440	T_{tr} = 360
clock HIGH t_{HC}	160			min > 0.35T = 140 (at typical data rate)
clock LOW t_{LC}	160			min > 0.35T = 140 (at typical data rate)
delay t_{dtr}			300	max < 0.80T = 320 (at typical data rate)
hold time t_{htr}	100			min > 0
clock rise-time t_{RC}			60	max > 0.15T_{tr} = 54 (only relevant in slave mode)

Example: Slave receiver with data rate of 2.5MHz (±10%) (all values in ns)

	MIN	TYP	MAX	CONDITION
clock period T	360	400	440	T_{tr} = 360
clock HIGH t_{HC}	110			min < 0.35T = 126
clock LOW t_{LC}	110			min < 0.35T = 126
set-up time t_{sr}	60			min < 0.20T = 72
hold time t_{htr}	0			min < 0

Table 1. Timing for I²S transmitters and receivers

	TRANSMITTER				RECEIVER				
	LOWER LIMIT		UPPER LIMIT		LOWER LIMIT		UPPER LIMIT		
	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	NOTES
Clock period T	T_{tr}				T_r				1
MASTER MODE:									
clock generated by transmitter or receiver:									
HIGH t_{HC}	$0.35T_{tr}$				$0.35T_{tr}$				2a
LOW t_{LC}	$0.35T_{tr}$				$0.35T_{tr}$				2a
SLAVE MODE:									
clock accepted by transmitter or receiver:									
HIGH t_{HC}		$0.35T_{tr}$				$0.35T_r$			2b
LOW t_{LC}		$0.35T_{tr}$				$0.35T_r$			2b
rise-time t_{RC}			$0.15T_{tr}$						3
TRANSMITTER:									
delay t_{dtr}				0.8T					4
hold time t_{htr}	0								3
RECEIVER:									
set-up time t_{sr}						$0.2T_r$			5
hold time t_{hr}						0			5

All timing values are specified with respect to high and low threshold levels.

NOTES:

1. The system clock period T must be greater than T_{tr} and T_r because both the transmitter and receiver have to be able to handle the data transfer rate.

2a. At all data rates in the master mode, the transmitter or receiver generates a clock signal with a fixed mark/space ratio. For this reason t_{HC} and t_{LC} are specified with respect to T.

2b. In the slave mode, the transmitter and receiver need a clock signal with minimum HIGH and LOW periods so that they can detect the signal. So long as the minimum periods are greater than $0.35T_r$, any clock that meets the requirements can be used (see Figure 3).

3. Because the delay (t_{dtr}) and the maximum transmitter speed (defined by T_{tr}) are related, a fast transmitter driven by a slow clock edge can result in t_{dtr} not exceeding t_{RC} which means t_{htr} becomes zero or negative. Therefore, the transmitter has to guarantee that t_{htr} is greater than or equal to zero, so long as the clock rise-time t_{RC} is not more than t_{RCmax}, where t_{RCmax} is not less than $0.15T_{tr}$.

4. To allow data to be clocked out on a falling edge, the delay is specified with respect to the rising edge of the clock signal and T, always giving the receiver sufficient set-up time.

5. The data set-up and hold time must not be less than the specified receiver set-up and hold time.

SN00122

Figure 4. Clock rise-time definition with respect to the voltage levels

5.0 VOLTAGE LEVEL SPECIFICATION

5.1 Output Levels

V_L < 0.4V
V_H > 2.4V both levels able to drive one standard TTL input (I_{IL} = –1.6mA and
I_{IH} = 0.04mA).

5.2 Input Levels

V_{IL} = 0.8V
V_{IH} = 2.0V

Note: At present, TTL is considered a standard for logic levels. As other IC (LSI) technologies become popular, other levels will also be supported.

6.0 POSSIBLE HARDWARE CONFIGURATIONS

6.1 Transmitter (see Figure 5)

At each WS-level change, a pulse WSP is derived for synchronously parallel-loading the shift register. The output of one of the data latches is then enabled depending on the WS signal. Since the serial data input is zero, all the bits after the LSB will also be zero.

6.2 Receiver (see Figure 6)

Following the first WS-level change, WSP will reset the counter on the falling edge of SCK. After decoding the counter value in a “1 out of n” decoder, the MSB latch (B1) is enabled (EN1 = 1), and the first serial data bit (the MSB) is latched into B1 on the rising edge of SCK. As the counter increases by one every clock pulse, subsequent data bits are latched into B2 to Bn.

On the next WS-level change, the contents of the n latches are written in parallel, depending on WSD, into either the left or the right data-word latch. After this, latches B2 to Bn are cleared and the counter reset. If there are more than n serial data bits to be latched, the counter is inhibited after Bn (the receiver’s LSB) is filled and subsequent bits are ignored.

Note: The counter and decoder can be replaced by an n-bit shift-register (see Figure 7) in which a single ‘1’ is loaded into the MSB position when WSP occurs. On every subsequent clock pulse, this ‘1’ shifts one place, enabling the N latches. This configuration may prove useful if the layout has to be taken into account.

Figure 5. Possible transmitter configuration

Figure 6. Possible receiver configuration. The latches and the counter use synchronous set, reset and enable inputs, where set overrules the reset input, and reset overrules the enable input.

Figure 7. Possible receiver configuration, using an n-bit shift-register to enable control of data input register.