SmartFusion: DSP Coprocessor in FPGA Fabric

Table of Contents

Introduction

SmartFusion™ mixed signal FPGAs contain a hard embedded microcontroller subsystem (MSS), programmable analog circuitry, and FPGA fabric consisting of logic tiles, static random access memory (SRAM), and phase-locked loops (PLLs). The MSS consists of a 100 MHz ARM® Cortex™-M3 processor, advanced high-performance bus (AHB) matrix, system registers, Ethernet MAC, DMA engine, real-time clock (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), fabric interface controller (FIC), the Philips Inter-Integrated Circuit (I^2C), serial peripheral interface (SPI), and external memory controller (EMC).

The intent of this application note is to demonstrate the capability of DSP coprocessing engine in the FPGA fabric of SmartFusion devices. The DSP coprocessor example design can be used where the Cortex-M3 processor in the SmartFusion device needs to offload the DSP processing (audio/video/image) to the FPGA fabric to boost overall performance.

This example design uses Cortex-M3 in the SmartFusion MSS as master and the fast fourier transform processor (CoreFFT IP) in fabric as DSP coprocessor. The Cortex-M3 can provide data directly to the coprocessor for processing or it can use the peripheral direct memory access (PDMA) controller in the MSS for the data transfer and thus helps to free up Cortex-M3. This application note demonstrates both methodologies. The example design uses CoreFFT and the advanced peripheral bus interface (CoreAPB3). A custom-made APB3 interface has been developed to connect CoreFFT with the MSS via CoreAPB3.

A basic understanding of the SmartFusion design flow is assumed. Refer to *Using UART with SmartFusion - Actel Libero® Integrated Design Environment (IDE) and SoftConsole Flow Tutorial* to understand the SmartFusion design flow.

Design Example Overview

This design example demonstrates the capability of the DSP coprocessing engine in the FPGA fabric on the SmartFusion Evaluation Kit and SmartFusion Development Kit. The design example consists of a DSP coprocessor block and CoreAPB3 implemented in the FPGA fabric. CoreAPB3 connects the DSP coprocessor with MSS. The DSP coprocessor has CoreFFT IP, 64x16 FIFO, and a custom-made APB3 interface. Figure 1 illustrates how to interface the DSP coprocessor with the MSS. The GPIOs in the MSS are used to handle flow control in data transfer from the MSS to DSP coprocessor. The UART in the MSS is used for printing the FFT values on HyperTerminal.

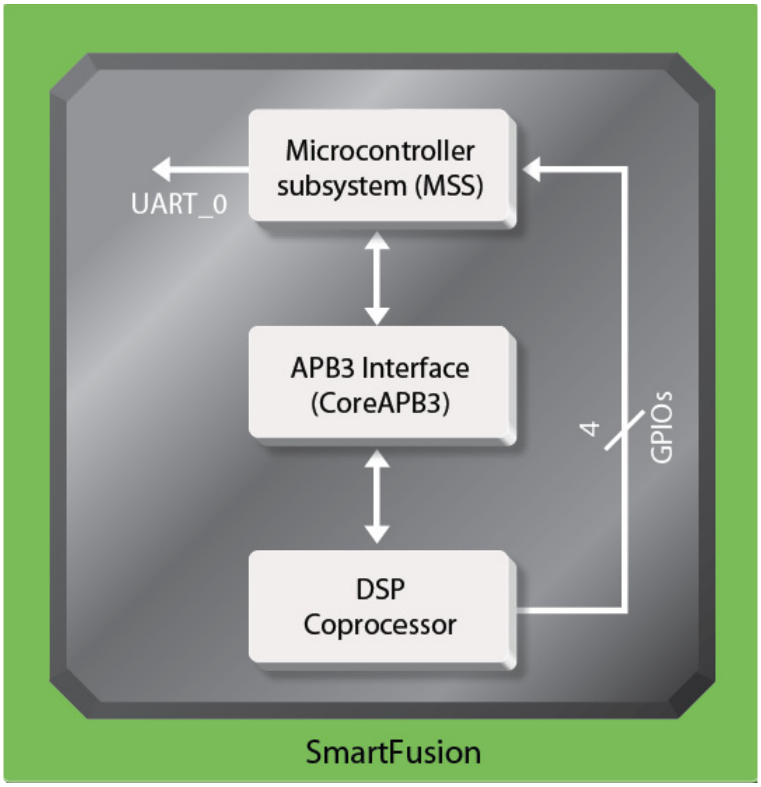


Figure 1 • **System Level Interface Signals**

Description of the Design Example

Description of DSP Coprocessor

The design uses CoreFFT as a DSP coprocessor. You can download the core generator for CoreFFT at www.actel.com/products/ip/search/evaluate.aspx?m=624&ev=60. The example design uses a 256-point and 8-bit FFT. A custom-made APB3 interface has been developed to connect CoreFFT with MSS. CoreFFT output data is stored in a 64x16 FIFO. The FIFO status signals are given in Table 1 on page 3. Status signals indicate that FFT is ready to receive data and data is available in the output of FIFO. These status signals are mapped to the GPIOs in the MSS. The Cortex-M3 can read GPIOs to handle flow control in data transfer from the MSS to the DSP coprocessor.

You can implement various methodologies based on the application to handle data transfer, such as polling of the general purpose input/output (GPIOs) for status, using GPIOs as interrupts to the MSS or using PDMA. Figure 2 on page 3 shows the block diagram of DSP coprocessor with custom-made APB3 bus.

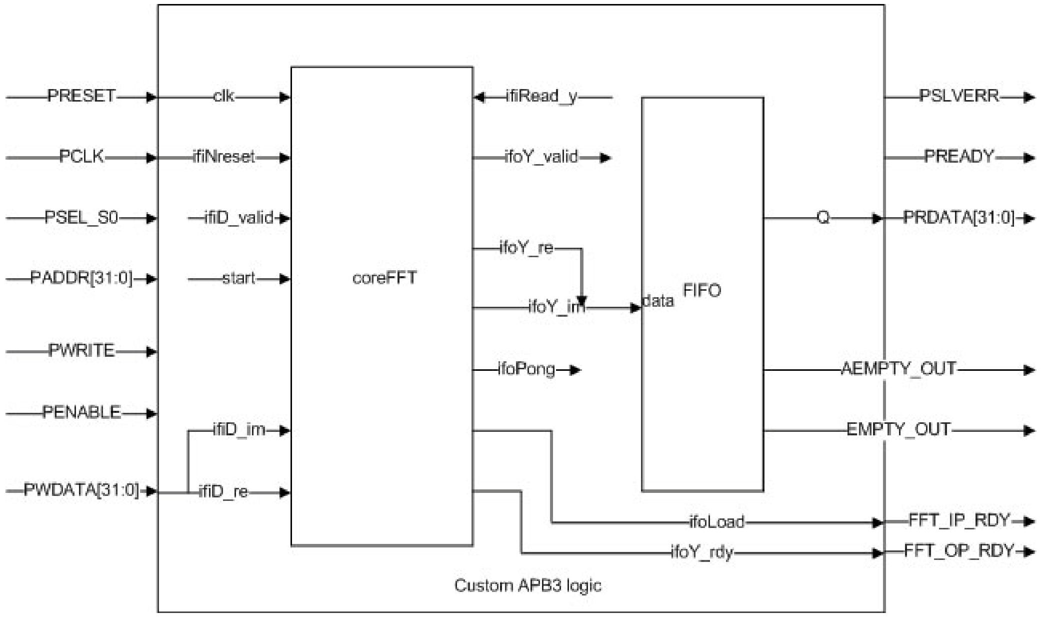


Figure 2 • **DSP Coprocessor with Custom-Made APB3 Bus**

The data valid signal is generated in custom logic whenever the master needs to write data into the input buffer of the FFT to process through the APB3 interface. The FFT_IP_RDY signal indicates the status of the input buffer of the FFT. If the input buffer is full, the FFT_IP_RDY goes low. The master can read the FFT_IP_RDY signal to get the FFT input buffer status. The FFT generates the processed data with a data valid signal. The processed data is stored in the FIFO. When FIFO is not ready to receive output data, it can stop the data fetching from the FFT by pulling down the ifiRead_y signal.

The status signal FFT_OP_RDY is used to indicate to the Master that processed data is available in the FIFO. FFT_OP_RDY goes High whenever processed data is available in the FFT output buffer. The Master can use AEMPTY_OUT or EMPTY_OUT to determine whether the FIFO is empty and all the processed data has been read. Refer to the CoreFFT Handbook for more details on architecture and interface signal descriptions.

Table 1 • **FIFO Status Signals with Descriptions**

Signal	Description
FFT_IP_RDY	FFT is ready to receive the Input from the master processor
FFT_OP_RDY	Processed data is ready in output buffer of FFT
AEMPTY_OUT	Output FIFO is almost empty
EMPTY_OUT	Output FIFO is empty

SmartDesign Implementations

The design example consists of the MSS, CoreAPB3, and the DSP coprocessor block. The DSP coprocessor block includes CoreFFT, 64x16 FIFO, and a custom-made APB3 interface. The CoreAPB3 connects the DSP coprocessor with the FIC in the MSS.

The MSS is configured with an FIC, clock conditioning circuit (CCC), GPIOs, and a UART. The CCC generates a 40 MHz clock, which acts as the clock source. The FIC is configured to use a Master interface with an AMBA APB3 interface. Four GPIOs in the MSS are configured as inputs that are used to handle flow control in data transfer from MSS to DSP coprocessor. The UART_0 is configured for printing the FFT values on HyperTerminal.

The CoreAPB3 acts as a bridge between the MSS and DSP coprocessor block. It provides AMBA3 and APB3 fabric supporting up to 16 APB slaves. This design example uses one slave slot (Slot 0) to interface with DSP coprocessor block and is configured with direct addressing mode. Refer to the *CoreAPB3 Handbook* for more details on CoreAPB3 IP.

Figure 3 illustrates the DSP Coprocessor Example Design in SmartDesign.

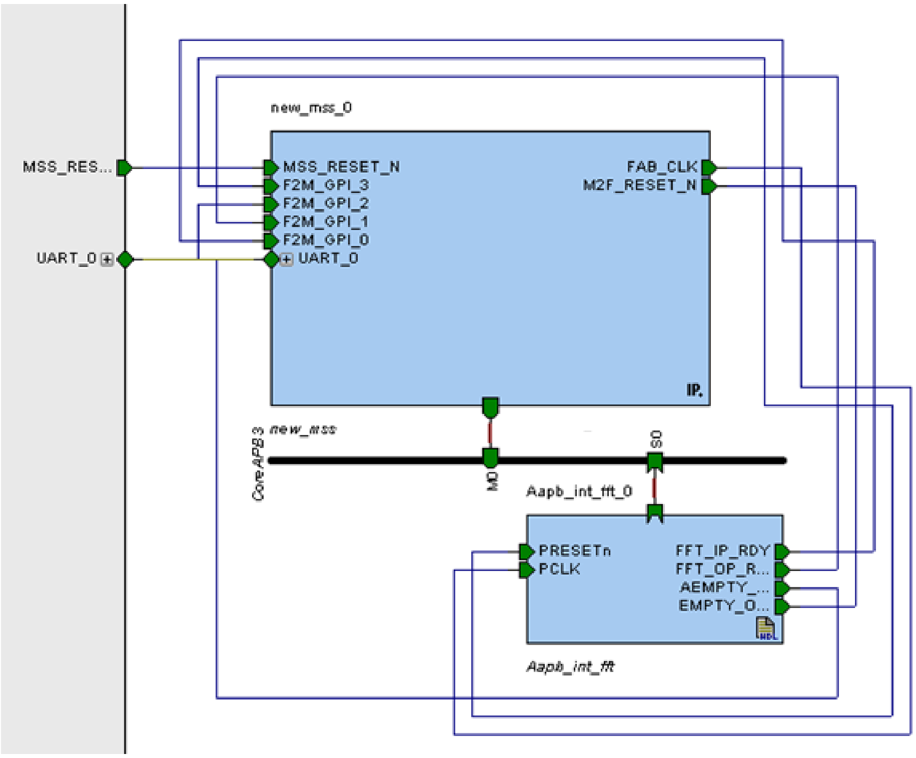


Figure 3 • **DSP Coprocessor Example Design in SmartDesign**

Both Verilog and VHDL Libero IDE projects and SoftConsole project are provided in the design files attached with this design example.

Software API

The CALL_FFT () API can be used whenever DSP coprocessor is required to transform the input data. This API configures the PDMA for data transfer from the Cortex-M3 processor to DSP coprocessor and from DSP coprocessor to Cortex-M3.

Without PDMA, the Cortex-M3 transfers data by polling the status signals.

Data Transfer with PDMA

The peripheral DMA (PDMA) in the MSS consists of eight instances of a single DMA channel design. Each channel can be configured to perform 8-/16-/32-bit transfers from the peripheral to memory, memory to peripheral, or between memory and memory. Channels can be assigned to peripherals or memory arbitrarily.

The Cortex-M3 can use PDMA to transfer data to the DSP coprocessor and thus free itself for other tasks. Custom APB3 logic will cause the PREADY signal to be low whenever FFT is not ready for the data transfer.

Data Transfer without PDMA

Cortex-M3 can transfer data from memory for processing or processed data to the memory. Cortex-M3 polls the status signals for the flow control with coprocessor. These status signals are mapped to the GPIOs in the MSS.

Simulation

System level simulation is done by modifying the user.bfm script available in the simulation folder of the MSS to verify the custom interface. Refer to the *DirectCore AMBA BFM User's Guide* for further details on system level simulation with SmartFusion. Figure 4 illustrates the FFT write operation. When data is written to FFT for processing, the data valid signal, if oY_valid, goes High.

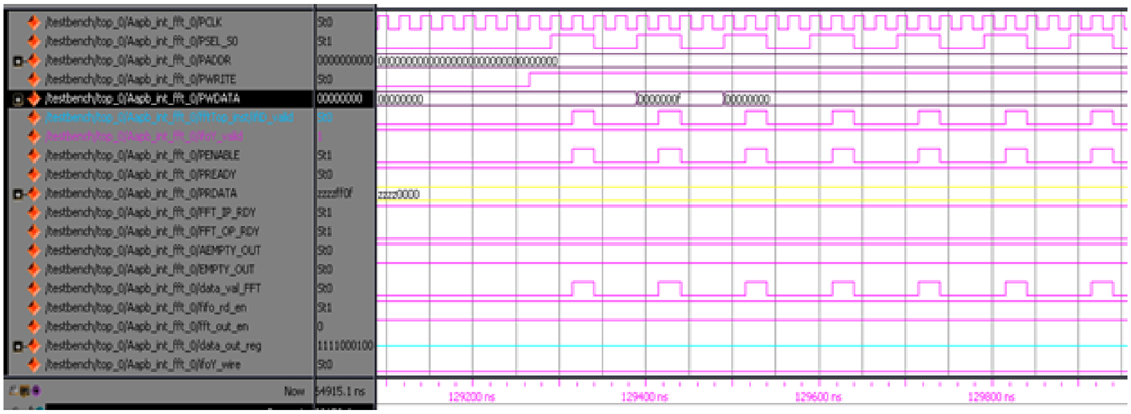
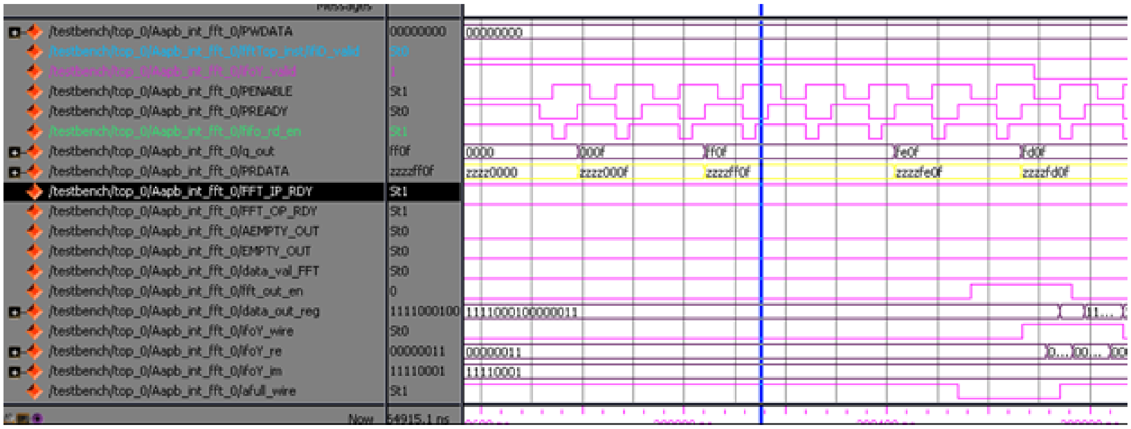

/testbench/top_0/Aapb_int_fft_0/PCLK St0
/testbench/top_0/Aapb_int_fft_0/PSEL_S0 St1
/testbench/top_0/Aapb_int_fft_0/PADDR 0000000000
/testbench/top_0/Aapb_int_fft_0/PWRITE St0
/testbench/top_0/Aapb_int_fft_0/PWDATA 00000000
/testbench/top_0/Aapb_int_fft_0/fftTop_inst/ifiD_valid St0
/testbench/top_0/Aapb_int_fft_0/ifoY_valid 1
/testbench/top_0/Aapb_int_fft_0/PENABLE St1
/testbench/top_0/Aapb_int_fft_0/PREADY St0
/testbench/top_0/Aapb_int_fft_0/PRDATA zzzzff0f
/testbench/top_0/Aapb_int_fft_0/FFT_IP_RDY St1
/testbench/top_0/Aapb_int_fft_0/FFT_OP_RDY St1
/testbench/top_0/Aapb_int_fft_0/AEMPTY_OUT St0
/testbench/top_0/Aapb_int_fft_0/EMPTY_OUT St0
/testbench/top_0/Aapb_int_fft_0/data_val_FFT St0
/testbench/top_0/Aapb_int_fft_0/fifo_rd_en St1
/testbench/top_0/Aapb_int_fft_0/fft_out_en 0
/testbench/top_0/Aapb_int_fft_0/data_out_reg 1111000100
/testbench/top_0/Aapb_int_fft_0/ifoY_wire St0
Now 54915.1 ns
00000000
0000000f
00000000
zzzz0000
129200 ns
129400 ns
129600 ns
129800 ns

Figure 4 • **CoreFFT Write Operation**

Figure 5 illustrates the FFT read operation. If a read request is received from the Cortex-M3 processor, the fifo_rd_en signal is asserted and data is transferred. Figure 5 shows fft_out_en signal is requesting the processed data from the FFT as shown.



Messages	
/testbench/top_0/Aapb_int_fft_0/PWDATA	00000000
/testbench/top_0/Aapb_int_fft_0/fftTop_inst/fifO_valid	St0
/testbench/top_0/Aapb_int_fft_0/ifoY_valid	1
/testbench/top_0/Aapb_int_fft_0/PENABLE	St1
/testbench/top_0/Aapb_int_fft_0/PREADY	St0
/testbench/top_0/Aapb_int_fft_0/fifo_rd_en	St1
/testbench/top_0/Aapb_int_fft_0/q_out	ff0f
/testbench/top_0/Aapb_int_fft_0/PRDATA	zzzzff0f
/testbench/top_0/Aapb_int_fft_0/FFT_IP_RDY	St1
/testbench/top_0/Aapb_int_fft_0/FFT_OP_RDY	St1
/testbench/top_0/Aapb_int_fft_0/AEMPTY_OUT	St0
/testbench/top_0/Aapb_int_fft_0/EMPTY_OUT	St0
/testbench/top_0/Aapb_int_fft_0/data_val_FFT	St0
/testbench/top_0/Aapb_int_fft_0/fft_out_en	0
/testbench/top_0/Aapb_int_fft_0/data_out_reg	1111000100
/testbench/top_0/Aapb_int_fft_0/ifoY_wire	St0
/testbench/top_0/Aapb_int_fft_0/ifoY_re	00000011
/testbench/top_0/Aapb_int_fft_0/ifoY_im	11110001
/testbench/top_0/Aapb_int_fft_0/afull_wire	St1
Now	54915.1 ns

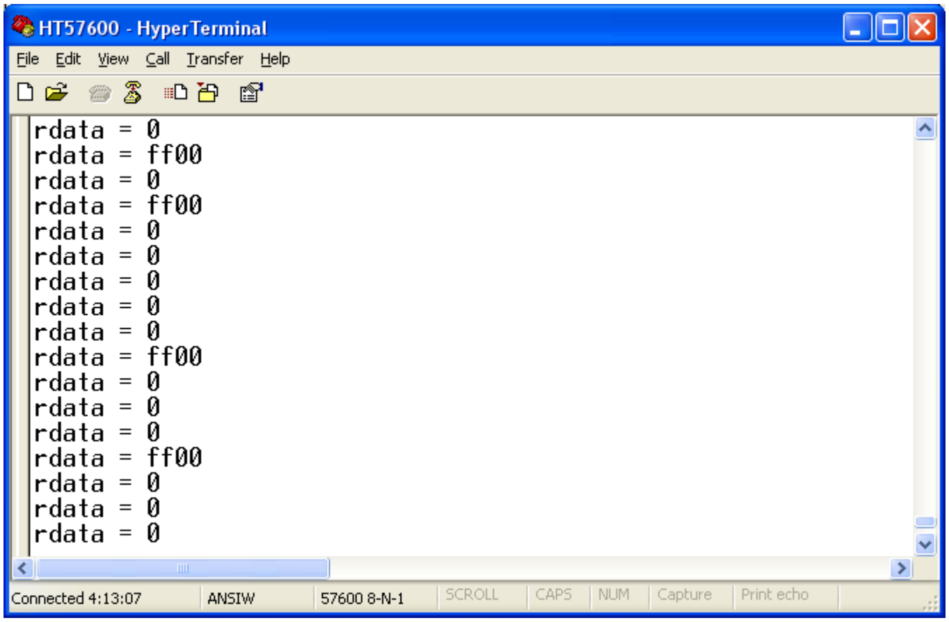
Figure 5 • **CoreFFT Read Operation**

Running the Design

Program the SmartFusion Evaluation Board or Development Board with the generated/provided PDB file (refer to "Appendix A - Design Files" on page 7) using FlashPro and then power cycle the board.

Invoke the SoftConsole IDE, browse for SoftConsole project folder (refer to "Appendix A - Design Files" on page 7)   and launch the debugger. Start a HyperTerminal with a baud rate of 57600, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have the HyperTerminal program, use any free serial terminal emulation program such as PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.

When you run the debugger in SoftConsole, the application starts printing the FFT values of a sine wave on HyperTerminal. Figure 6 shows an example of HyperTerminal with FFT values.



```
rdata = 0
rdata = ff00
rdata = 0
rdata = ff00
rdata = 0
rdata = 0
rdata = 0
rdata = 0
rdata = 0
rdata = ff00
rdata = 0
rdata = 0
rdata = 0
rdata = ff00
rdata = 0
rdata = 0
rdata = 0
```

Figure 6 • **Screenshot of HyperTerminal with FFT Values**

Conclusion

This application note described the capability of DSP coprocessing engine in FPGA fabric of SmartFusion devices. The DSP coprocessor design example demonstrates the usage where the ARM Cortex-M3 processor in the SmartFusion mixed signal FPGA needs to offload the DSP processing (audio/video/image) to FPGA fabric to boost the overall performance. This application note also describes the use of PDMA for further improvements in overall performance.

Appendix A - Design Files

The design files are available for download on www.actel.com/download/rsc/?f=A2F_AC355_DF.

The design file consists of Libero Verilog and VHDL projects, SoftConsole software project, and programming files (*.pdb). Refer to the Readme.txt file included in the design file for directory structure and description.

Microsemmi Corporate Headquarters
2381 Morse Avenue, Irvine, CA 92614
Phone: 949-221-7100 · Fax: 949-756-0308
www.microsemi.com