Wiki »

EXERCISE: ST BUS RTL Implementation

Exercise Goals

This exercise will through implementation ST-bus to IIS interface converter implementation and the use of a generated FIR filter introduce you to:

- The ST-bus protocol
- Wizard generated modules
- Configurations as a means of testing in the ModelSim IDE

Prerequisites

- Quartus and ModelSim software must be installed and working.
- The FIR Compiler II MegaCore Function User Guide
- Avalon Interface Specification chapter 6
- Skeleton code checked out from the repo (exercise_iis2st_rtl)

Introduction

The ST-bus is used in IntelFPGA SoC systems for low-latency data transfer. It is particularly well-suited for continuous (streaming) data.

In this exercise we'll create a bridge between the digital audio interface (IIS) and the ST-bus. Se the I2S-ST Functional Test Bench exercise for detail on how the interfaces are wired together.

The IIS bus, it's a 1-bit serial bus with two channels, left and right:

WM8731 / WM8731L Production Data

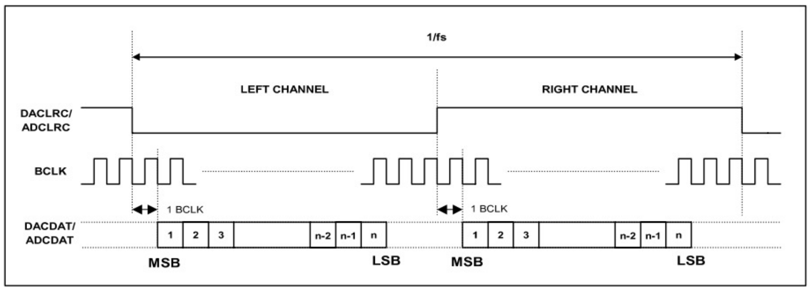
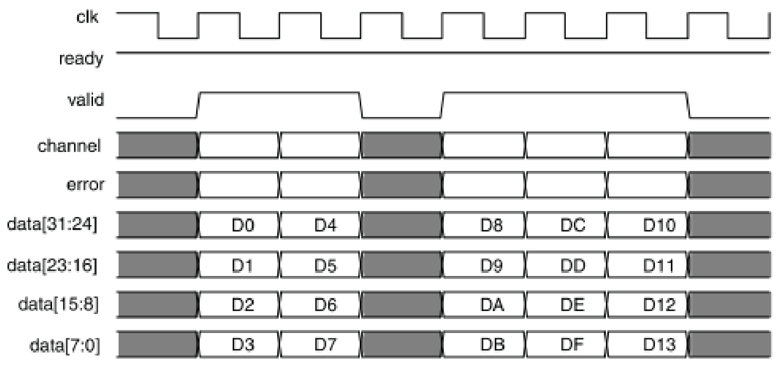


Figure 27 I²S Mode

Similar, but different is the ST-bus:



This example of a 32-bit/beat interface shows a sequential bus, yet data are transferred ats parallel bits. The IIS bus is only 1-bit/beat, but 24-bits per package (compared to the ST-bus)

Exercise Steps

Open and check the exercise_iis2st_rtl project.

Open the project in Quartus, and verify that it compiles.

Copy your iis2st_tb from last exercise to the folder, create a Modelsim project and verify that simulation also works.

1 Create the RTL version of the IIS2ST Converter

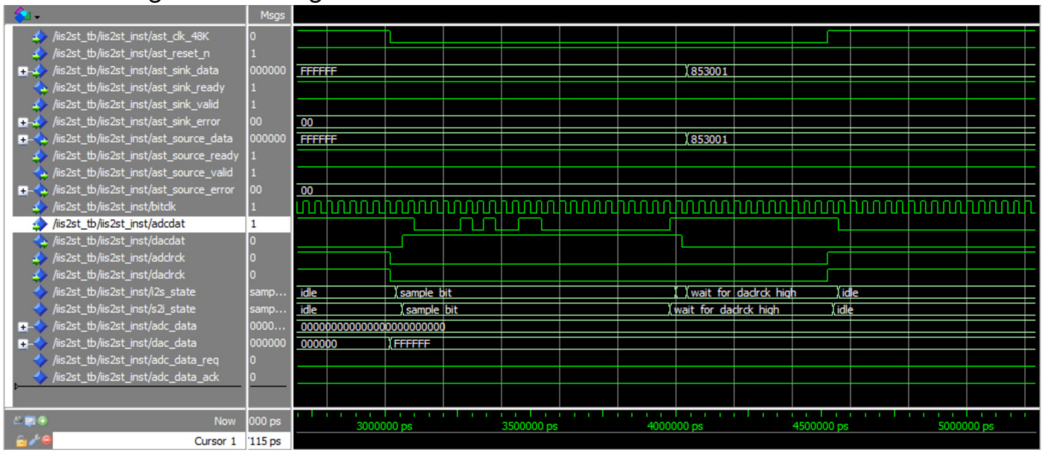
Open Modelsim and the project from the "Sequential TB exercise". The IIS2ST module contains two architectures, functional and RTL. You must implement the RTL version.

Start out by implementing the IIS to ST converter. Use a clocked state machine (see SimpleFSM code example here). Wait for ADCLRCK to go low, wait one clock cycle, shift the IIS data in, and latch the result when all 24-bits, has been recieved. See the IIS data sheet waveform above. You could use states like: "Waiting_for_adclrc_to_go_low", "Sample_iis_data" and "Latch_data"

Verify by simulating, but this time use the "rtl_sim_cfg" configuration in ModelSim.

When it works in one direction, implement the other in a similar fashion.

You should get something like this:


Msgs
/iis2st_tb/iis2st_inst/ast_clk_48K 0
/iis2st_tb/iis2st_inst/ast_reset_n 1
/iis2st_tb/iis2st_inst/ast_sink_data 000000 FFFFFF 853001
/iis2st_tb/iis2st_inst/ast_sink_ready 1
/iis2st_tb/iis2st_inst/ast_sink_valid 1
/iis2st_tb/iis2st_inst/ast_sink_error 00 00
/iis2st_tb/iis2st_inst/ast_source_data 000000 FFFFFF 853001
/iis2st_tb/iis2st_inst/ast_source_ready 1
/iis2st_tb/iis2st_inst/ast_source_valid 1
/iis2st_tb/iis2st_inst/ast_source_error 00 00
/iis2st_tb/iis2st_inst/bitclk 1
/iis2st_tb/iis2st_inst/adcdat 1
/iis2st_tb/iis2st_inst/dacdat 0
/iis2st_tb/iis2st_inst/adclrck 0
/iis2st_tb/iis2st_inst/daclrck 0
/iis2st_tb/iis2st_inst/i2s_state samp... idle sample_bit wait_for_daclrck_high idle
/iis2st_tb/iis2st_inst/s2i_state samp... idle sample_bit wait_for_daclrck_high idle
/iis2st_tb/iis2st_inst/adc_data 0000... 000000000000000000000000
/iis2st_tb/iis2st_inst/dac_data 000000 000000 FFFFFF
/iis2st_tb/iis2st_inst/adc_data_req 0
/iis2st_tb/iis2st_inst/adc_data_ack 0
Now 000 ps
Cursor 1 '115 ps
3000000 ps
3500000 ps
4000000 ps
4500000 ps
5000000 ps

2 Implement on DE1-SoC

Open the project in Quartus.

Investigate the top-level file in the project "exercise_iis2st_rtl".

Verify that your iis2st component is integrated correctly into de1soc_audio_no_hps_top. The module should be wired together as in the test bench, with a loopback on the ST-bus side.

Build the project in Quartus. You should be able to download it to DE1SOC and verify that audio is looped back from line-in to line-out.

3 Create a Fir Filter

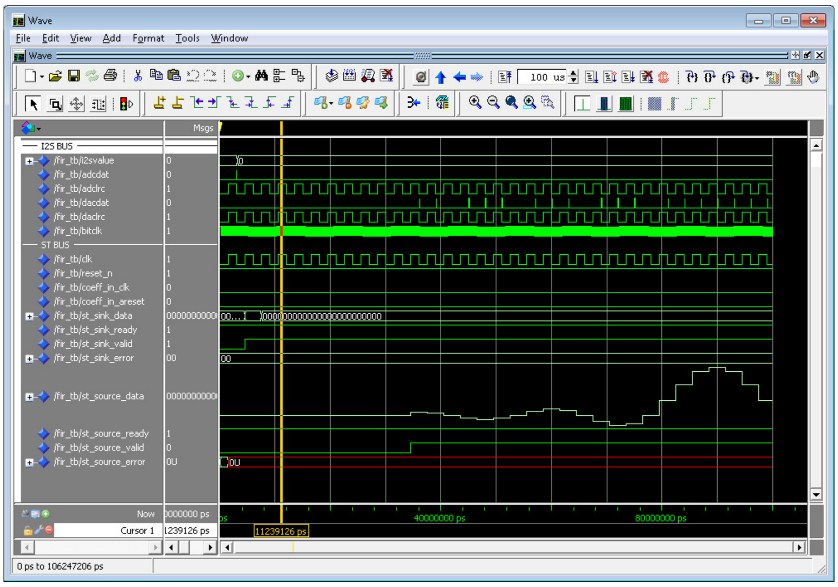
Create a FIR filter with the Quartus' built-in Fir filter generator. Follow this Tutorial to create a FIR filter with Quartus II FIR compiler II .

4 Insert the FIR filter into iis2st_tb

A) In your ModelSim project add the following (presuming you have called your fir filter for "fir_filter"):

- fir_filter.vhd
- iis2st.vhd
- iis2st_tb.vhd
- Plus all files generated by the FIR compiler, if you have used it

B) Simulate! The test bench generates an impulse signal on the IIS interface and passes it through the IIS2ST converter through the fir filter and back. Pick the "rtl_sim_cfg" configuration and you should get something like this:



5 Integrate FIR into Quartus Project

Insert the FIR filter into the Quartus top-level file and wire codec adc signals to the FIR input and codec DAC input to the FIR output. Use the 48KHz clock to clock the FIR filter.

Build and download to DE1SOC, you should now be able to hear the FIR filter in effect.