Wiki »

EXERCISE: I2S/ST FUNCTIONAL TEST BENCH

Exercise Goals

This exercise will through simulation of realistic interfaces introduce you to:

- Creation of sequential test benches in the ModelSim IDE
- Creation of stimuli procedures and response monitors
- Create of non-syntheziable VHDL code
- Serial data streaming
- working with configurations in simulation

Prerequisites

- Quartus and ModelSim Altera software must be installed and working.
- Knowledge about response monitors, stimuli procedures and configurations
- Template code from the repository (exercise_seq_tb, see tab)

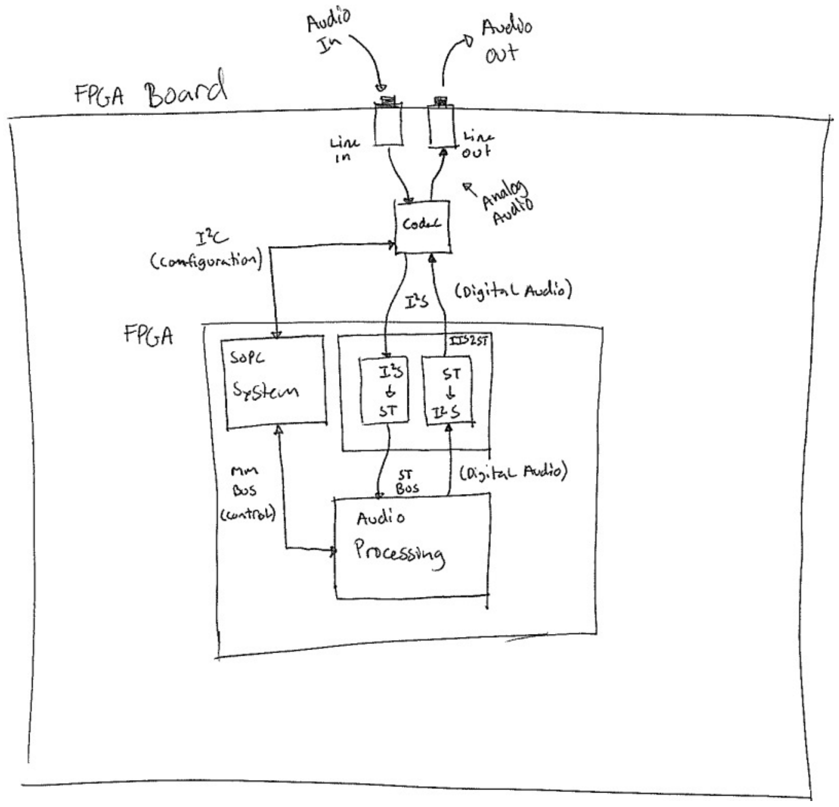
Introduction

This exercise is about sequential testbenches, but we'll use it for a real interface, so heres a breif introduction...

The DE1SoC board used in DSPC provides a codec for sampling and outputting analogue audio. The codec interfaces to the FPGA by means of two serial interfaces:

- IIC (or I2C) for configuring the codec's sampling rate, attenuation and a lot of other stuff. (See XXX for more detail).

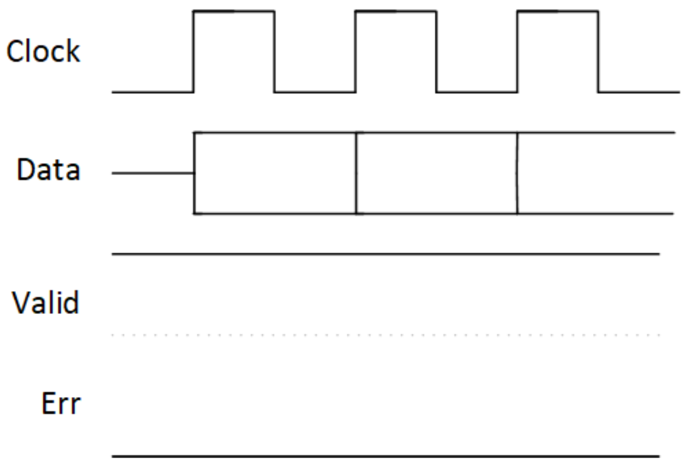
- IIS (or I2S) for audio data. This 1-bit data stream sends the digital audio values between the codec and FPGA.



Altera uses a streaming bus interface, ST-bus, for streaming data and provides FIR, IIR, and a lot of other signal processing blocks that are compliant with the ST-bus. Adapting our IIS audio format to Altera ST, will thus provide us with a large toolbox for manipulating audio.

We'll go into detail with the ST-bus in a later lesson, but here is a breif introduction:

The ST-bus interface is a serial interface, that sends data every clock cycle. Data can be one or multiple parallel bits (requires more connections for more bits). Data is latched out one the rising clock edge, and should be sampled on the falling clock edge. Clock and data is accompanied by two signals "Valid" and "Error". The "Valid" signal indicates the the current data i valid. This can be useful in cases where data can sometimes be invalid. Ex. You can have a valid data sample every 10th clock cycle, for a system sampling at a 10th of the system clock speed. Error indicates to the next block if an error has occurred and is a way to propagate errors through a chain of streaming bus connected blocks. The basic waveform looks like this:



The IIS interface is a streaming interface commonly used by stereo Codecs:

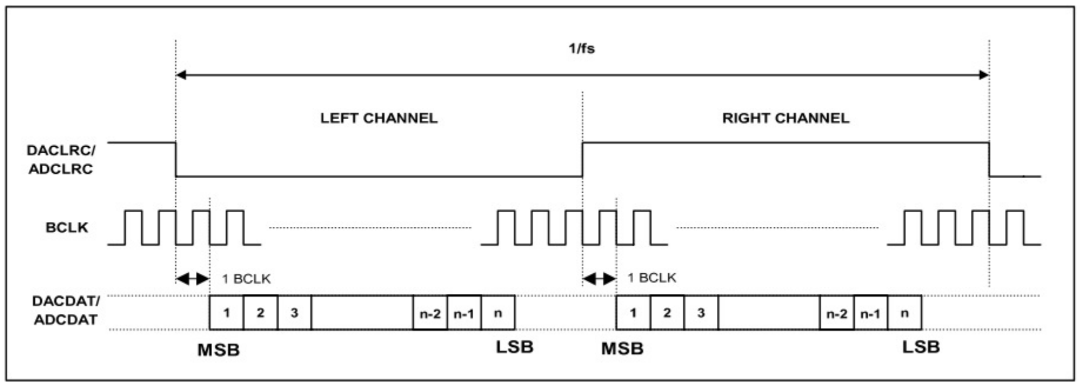
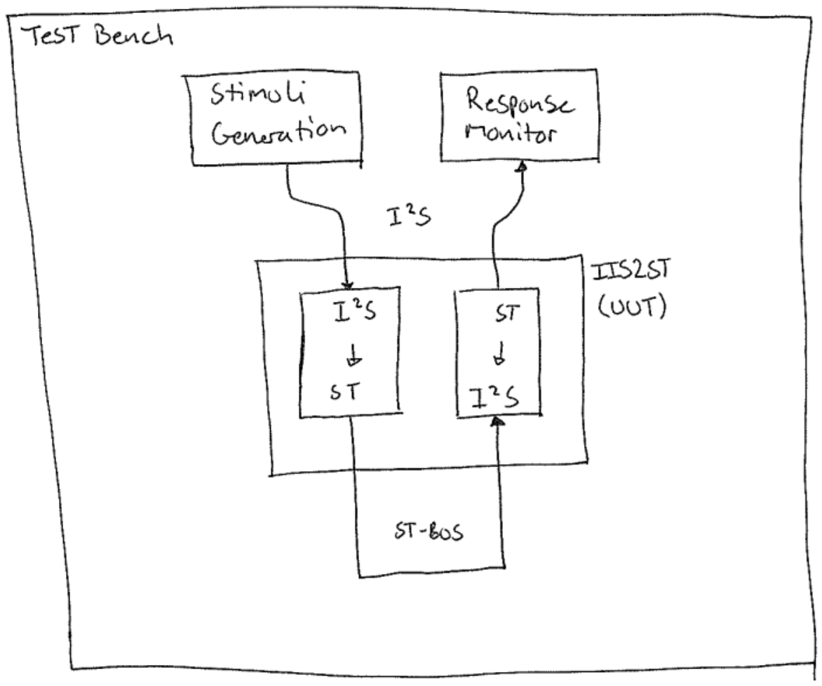


Figure 27 I^2S Mode

The daclrck/adclrck signal indicates the audio channel (left or right), and the beginning of the serial data sequence (dacdat/adcdat). bclk is the bit clock and is used to indicate when to sample the data bits. The codec typically runs with 24-bit resolution, and 24-bits are therefore shifted out per channel. This cycle repeats with the audio sampling frequency, which in this case is 48 KHz.

For this exercise, a functional model of IIS <-> ST-bus interface converter has been made, your task is to verify its functionality. For this you'll have to create a functional test bench.



In a later exercise you'll have to implement the actual rtl code and use the converter in conjunction with an SOPC system on the DE1SoC board.

NOTE! In the following work we will ONLY USE THE LEFT CHANNEL of the IIS Interface

Exercise Steps

1 Inspect the existing code

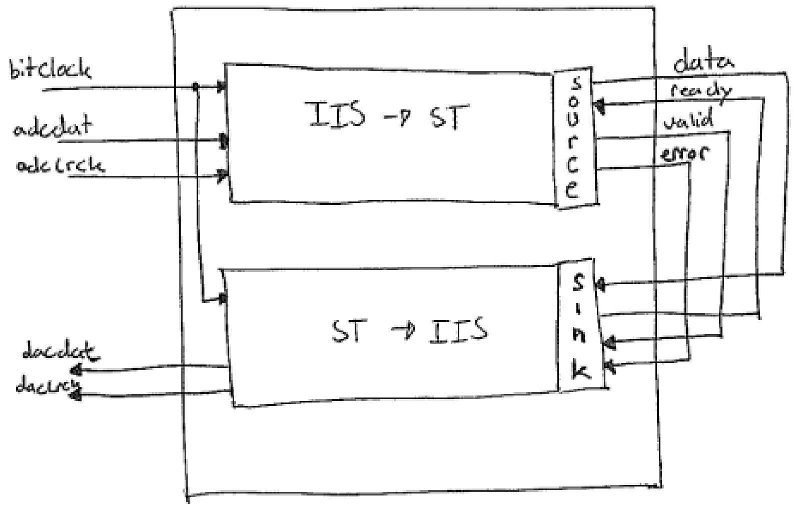
A) Create a project in ModelSim and include the files supplied.

B) Inspect the iis2st file, note how two architectures are present, though only one is filled in. Its code cannot be compiled to actual gates in the FPGA, as it uses ex. wait for xx ns (how would you want the compiler to implement that?)

C) Inspect the iis2st_tb file and note the configurations at the bottom. These let us select which architecture in iis2st to use when simulating.

2 Create Stimuli Procedure

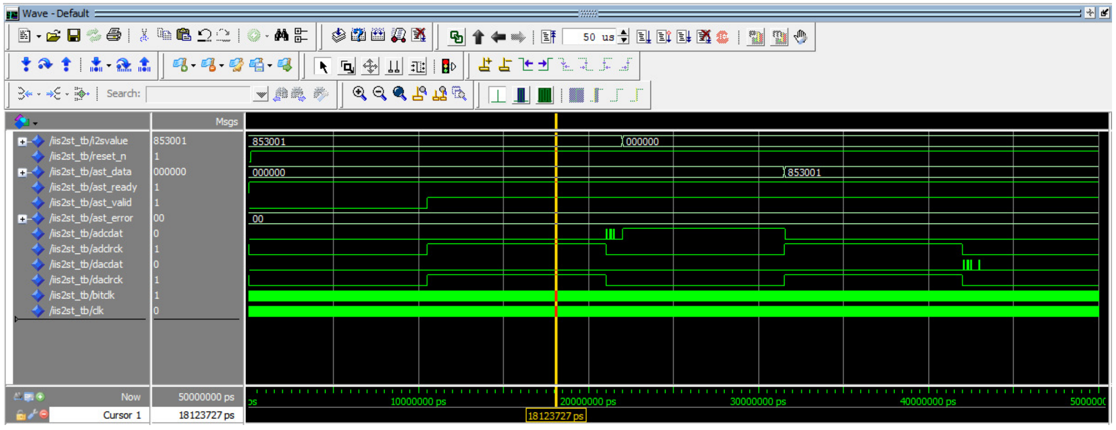
In the test bench note how source and sink is looped-back on the IIS2ST component instantiation. Connections are made according to the figure below.



A) Set the IIS input datastream (adcdat) to '0' and perform a simulation (run for 100 us). The output (dacdat) should remain zero. Set the input to '1' and try again. You should have the most basic test bench running now.

B) Create a stimuli procedure that can create an impulse function (|__) of a certain amplitude. The procedure shall emit a 24-bit constant value according to the IIS format followed by zeros: <value>______. Look carefully at the IIS format shown above. Use "Wait" statements to time when to transmit data, timed from the deassertion of ADCLRCK. Since it is a value shifted out, you should try and recall how to create a shift register. The value itself is the amplitude of the impulse. We'll use the impulse to test filters attached to the ST-bus later on.

C) Simulate that the impulse/step actually returns on the dacdat output (Just by viewing the waveform) The waveform below shows the adcdat data and how it ends up on the ST-bus (by means of the iis2st module):



```
                      Msgs
/iis2st_tb/i2svalue   853001
/iis2st_tb/reset_n    1
/iis2st_tb/ast_data   000000
/iis2st_tb/ast_ready  1
/iis2st_tb/ast_valid  1
/iis2st_tb/ast_error  00
/iis2st_tb/adcdat     0
/iis2st_tb/addrck     1
/iis2st_tb/dacdat     0
/iis2st_tb/dadrck     1
/iis2st_tb/bitclk     1
/iis2st_tb/clk        0

Now        50000000 ps
Cursor 1   18123727 ps
```

D) (Optional - But but good to do) Create a response monitor that can decode the dacdat sequence and report the value to ModelSim (only when it differs from zero or you'll get spammed)

E) (Optional - And harder to do) Extend the response monitor to estimate the latency (time from input to output) for a given input token. Report the latency.

Dspc_ex_seq_tb_de_block.png (36,5 KB) Peter Høgh Mikkelsen, 2017-01-04 10:19