TSTE17 System Design, CDIO

Lab 1, Fast prototyping using Altera DSP Builder

v 0.4

2019-09-09

Kent Palmkvist, DA, Dept. EE, Linköpings Universitet

Table of Contents

1 Introduction

This lab manual is assuming that the student has some experience of VHDL synthesis and FPGA technology. Details about the specific technology used and the CAD tools included in the design flow is available online through the help function of each tool. Additional information is also available on the web pages for the CAD tool vendors.

The design entry is based on Simulink models. This tool is a simulation program integrated into MATLAB. The graphical user interface is straightforward, and this lab manual assumes the student already have experienced similar interface (e.g., HDL Designer), or is able to quickly learn. More information about Simulink is available in the help menu of the program. There are also tutorials and examples available in the help section.

1.1 Lab goals

Learn how to create a design in Simulink using the blocks from Altera DSP Builder, simulate it at different levels, and test the design in hardware including use of the built-in logic analyzer. The design example is also illustrating important design principles such as hierarchy and enable signals.

1.2 Design requirements

An up/down counter with reset shall be designed. It should be controlled by the pushbuttons SW4-7 of the FPGA board, and the counter value should be presented on the two 7-segment displays. The counting shall be synchronized to the 100 MHz system clock of the FPGA board. The decimal point of the first 7-segment display shall indicate when the next most significant digit will change value on the next press of the SW4 push button. The decimal point of the second 7-segment display should be controlled by SW7.

The count value should change once every time the SW4 push-button is pressed. The SW5 switch shall control the direction of the counter. SW6 shall be used to reset the counter. The counter shall count in decimal from 00 to 99.

1.3 Design description

The final design is shown in Figure 1 below. It consists of the support blocks for FPGA synthesis and logic analyzer (SignalCompiler, board configuration, SignalTap II), stimuli generation and analysis in Simulink (pulse generator, scope), I/O blocks (sw 4-7, display 0-1), and the functionality to be placed inside the FPGA (counters, flipflops, etc).

The complete design is a synchronous circuit using an implicit clock signal of 100 MHz (defined by the board configuration block). Flipflops are used to synchronize the pushbutton inputs to this clock signal.

All clocked elements in the design are clocked by the global 100 MHz clock, including the counters. The falling edge of the SW4 signal is instead detected and a one clock cycle long pulse is generated. This pulse is used as an enable signal that makes the counter change value only once when there is a falling edge on SW4.

Generation of the enable signal for the second counter (MSD) is more complicated. That part of the system has therefore been placed into a subsystem. There are two situations that generates an active enable signal, if LSD is going from 0->9 or 9->0. This is detected by checking the current value combined with the up/down signal and the enable input to the LSD counter. The implementation of this is shown in .Figure 2.

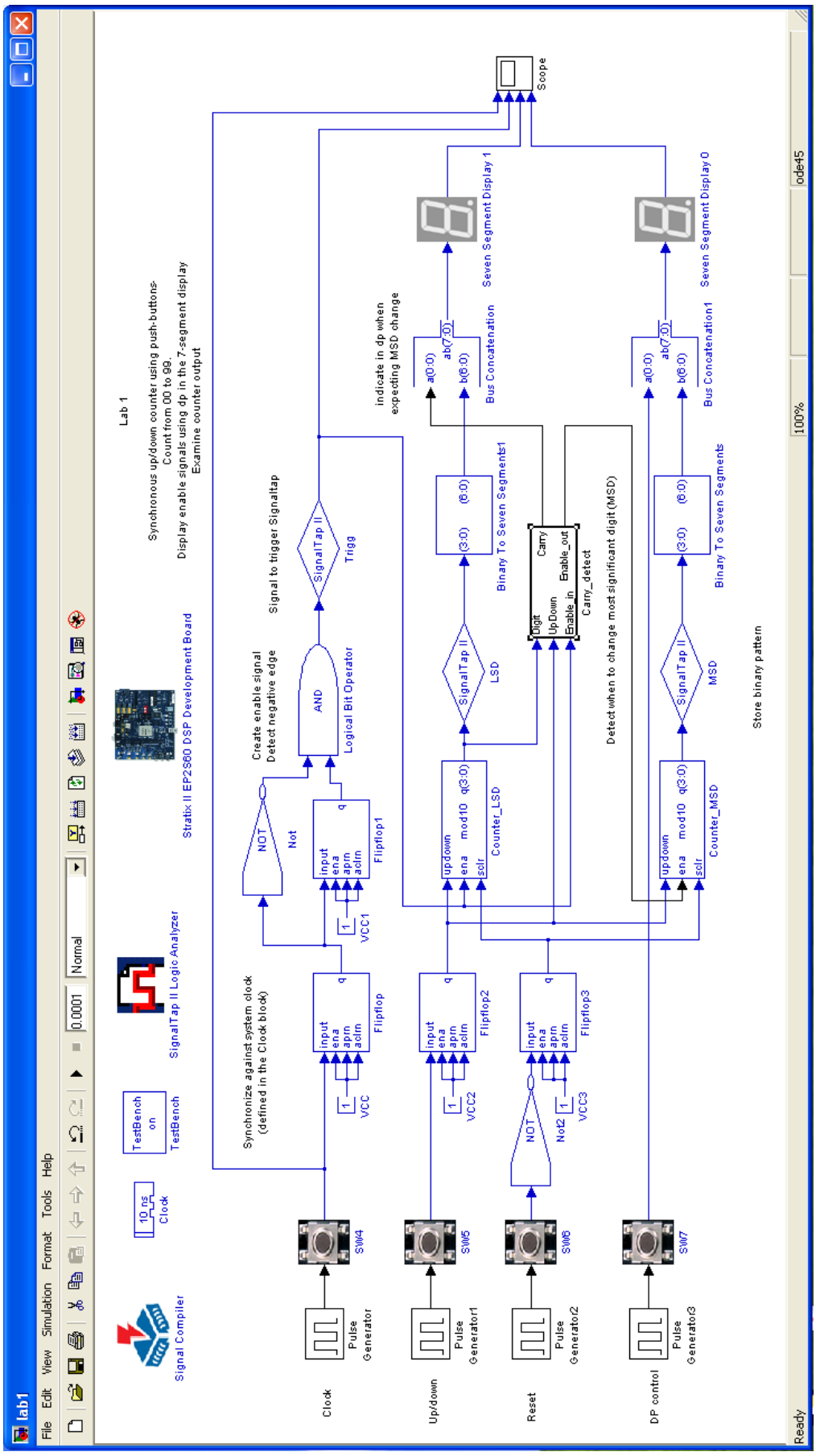

Figure 1: Complete system

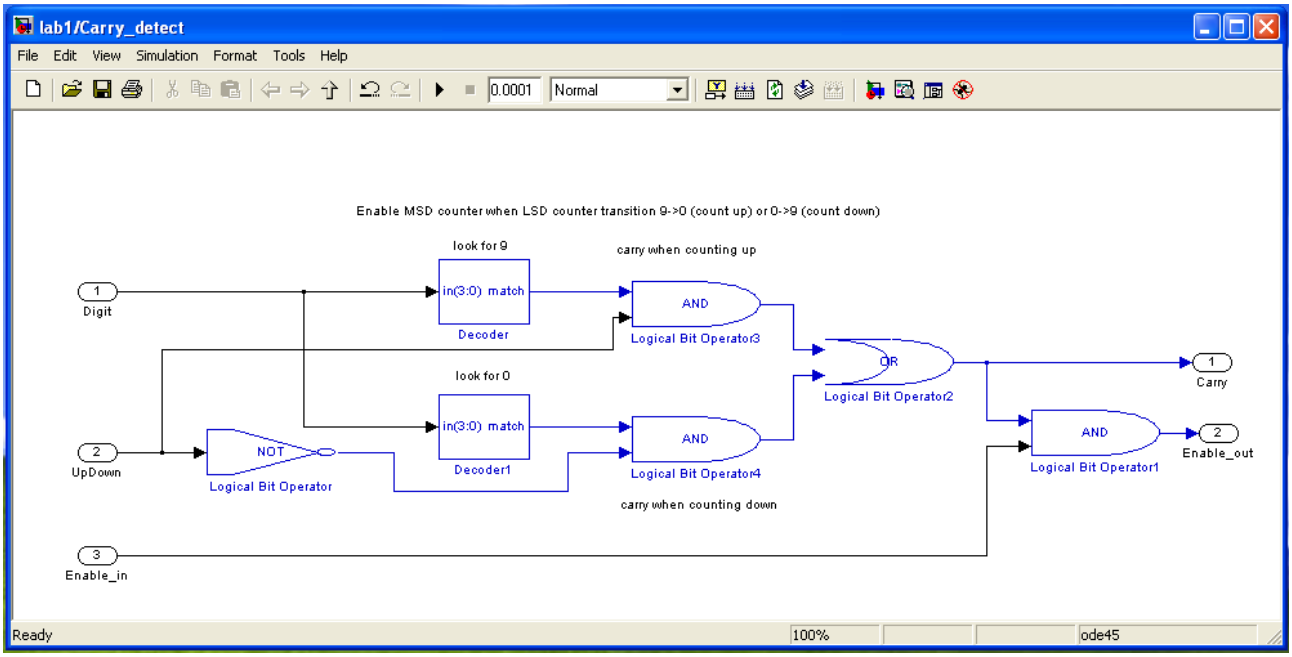

Figure 2: Carry detect subsystem

1.4 Overview of the design flow

The design flow used is divided into the following steps

- Initial setup of the computer and CAD system
- Enter the design
- Run a simulation in Simulink to verify functionality
- Create a VHDL model with support for the built-in logic analyzer
- Run a simulation of the VHDL model using ModelSim
- Synthesize the VHDL into FPGA configuration information
- Program and test the hardware
- Use the built-in logic analyzer to examine internal signals of the FPGA

2 Detailed descriptions of the design flow tasks

The following description tries to give a complete description of the steps taken. More details about individual steps and functionality of specific blocks and functions can be found in the documentation for Simulink, ModelSim, and Quartus software. All programs have online help, but you can also find some extra help regarding the Quartus DSP Builder at /courses/TSTE17/DSP_Builder_Documentation.

2.1 Simulink design entry

This section shows how to create the design shown in Figure 1. The description shows how to get a running model, even though the additional comments, formatting and similar is not described. There are also a lot of shortcuts that can be done (e.g., copy – paste) which is also not covered here.

Start by loading the TSTE17 module using *module load courses/TSTE17* followed by *TSTE17lab*. There is also a *TSTE17proj* command that not only starts the tool, but also changes the current directory to the project directory. For this lab, use only *TSTE17lab*.

Change the current directory to a directory within your home directory or in a group directory. Use the current directory field to the right in the matlab toolbar at the top of the window. It is also possible to use the commands *cd*, *dir*, and *pwd*. More alternatives can be found in the MATLAB help.

Important: Do not use a directory path that contains spaces anywhere! The design flow will not work if a directory name contains a space, such as "my documents"!

Start Simulink by pressing the Simulink library button or type *simulink* in the MATLAB window.

Hint: There is a lot of help available for the Simulink. This can be accessed by selecting *Help->Product Help*. Documentation can be found here for both MATLAB and Simulink, as well as the different blocksets used in Simulink. There is a separate document covering the Altera DSPBuilder block set. An additional DSP Builder tutorial can also be found in this chapter. Documentation for each individual block is available by right-clicking on the block and select help for the block.

Create a new design by *File->New->Model*.

Save the design using a new name, such as lab1. Important: Change the file type to .mdl, as the other available format is not supported by DSP Builder. Remember also that this file should be placed in your home directory (or some subdirectory there) where the path must not contain spaces.

Open the *Simulink* blockset, and select the *sources* block collection

Add the *Pulse Generator* block to the model by dragging it from the block collection and dropping it on the model. Drag three more pulse generators into the model.

Add the *Scope* block from the *sinks* block collection.

Open the "Altera DSP Builder Standard blockset", and select the AltLab block collection.

Add the SignalCompiler, SignalTap II Logic Analyzer, 3 units of SignalTap II Node, and a clock block to the new model by dragging the symbols from the library window into the model.

Open the Boards blockset and select the StratixIIEP2S60 block set. Add the Stratix II EP2S60 DSP Development Board, Seven Segment Display 0 , Seven Segment Display 1, and SW 4 to 7.

Open the Gate and Control block collection and add two Binary to Seven Segments blocks, four flipflop blocks and three logical bit operator blocks.

Open the IO and Bus block collection and add two bus concatenation blocks and four VCC blocks.

Finally open the Arithmetic block collection and add two counter blocks.

The names of the blocks may be changed to better illustrate their use. This is done by clicking on the name, and editing/entering the new name. Note that the names should follow VHDL syntax (start with A-z, contain only A-z, 0-9 and _). Additional free text may also be placed in the schematic. This is added by double click in some free area in the schematic. This free text is not used in the VHDL generation, and may thus contain any character.

Some of the blocks have the wrong number of inputs, or needs to be configured correctly. This is done by double-clicking on the blocks and then modifying the entries in the dialog. These dialog windows are also useful places to look for more information about some of the blocks.

Open the Stratix II EP2S60 DSP Development Board block. Set the clock pin in to Pin_AM17, the device to 2S60F1020C4, and the global reset input to PIN_AG19.

The busconcatenation is used to add the control of the dp element to the bus generated by the binary to 7-segment decoder. The dp is controlled by the most significant bit (MSB) of the bus, and bus A of the bus concatenation should therefore be set to 1, and bus B to 7. Make also sure that the output is defined as unsigned by unchecking the "Output is Signed" checkbox.

The two counters are counting unsigned integers, and should have 4 bits, counting modulo 10. The control inputs counter enable and synchronous clear are needed. Set the counter direction to “use direction port”.

The clock block should specify a Real-world clock period of 10, with a period unit of ns., and with a simulink sample time of 10e-9. This will make the simulation time equal to the real time behavior when the design is run on the FPGA board.

The testbench block in the altlab set should be added to the model. The system will then create a VHDL simulation model that will compare it with the simulation results from the simulink model.

The scope has initially only one input. Multiple inputs can sometimes be useful to simplify the comparison of signals. The number of inputs to the scope can be changed by opening the scope (double click on the block), and then select the parameter button (the button next to the printer button in the window, see Figure 3). Specify the number of axes to 4 in the parameter window, see Figure 4. The scope have a limited memory depth. The default is 5000 data points, which is too small in our case. Change this to ten times as many (50000) by selecting the Data history tab and modify the “limited to the last” entry. Press the OK button, this updates the block in the model to have 4 inputs and a larger history depth.

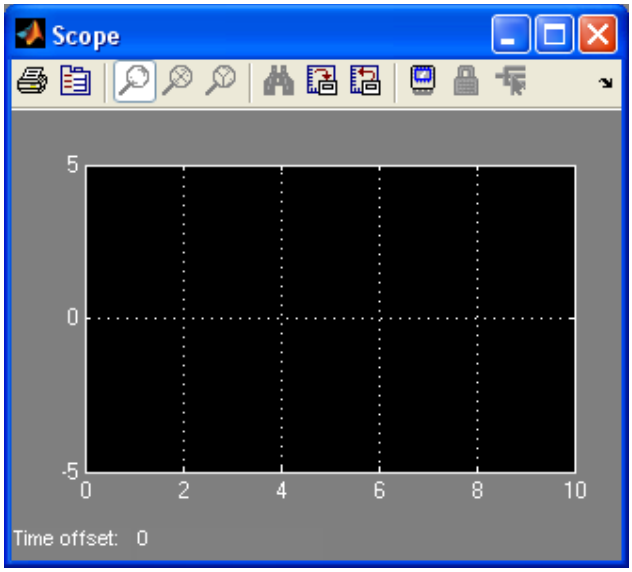

Figure 3 Scope window

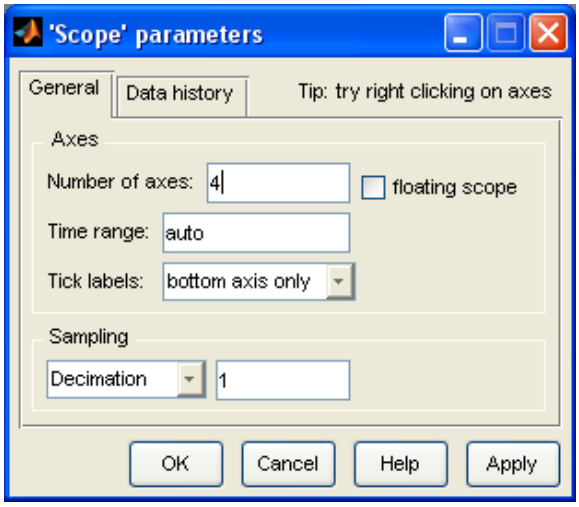

Figure 4 Scope parameter dialog

Next, the interconnections should be added. Wires are added by moving the cursor on top of an input or output to a block, then press the mouse button and dragging to the other port to connect to.

2.1.1 Adding a subsystem

Subsystems are used to reduce the complexity of the schematic by adding hierarchy to the design. There is support for a large number of different subsystems in the Simulink environment, but the DSPBuilder system only accepts the standard subsystem without any enable or trigger input.

Add the subsystem block from the Simulink library, the ports and subsystems set. The subsystem has initially one input and one output. Double-click on the subsystem to open it.

Additional inputs can now be added by adding in1 and out1 blocks from the Ports and Subsystems block collection in the Simulink blockset.

The subsystem should contain two decoder blocks detecting when the the least significant digit is zero and nine. Combined with the up/down control and the enable signals is the enable of the most significant digit calculated. See Figure 2 form more details.

2.2 Simulink simulation

The pulse generators in the model supplies the stimuli used in simulation. The parameters of the generators must be set to match the expected use of the design.

It is important to remember that the simulation takes time to perform. In this design is the system clock 100 MHz, and it is therefore time consuming to run simulation in the range of seconds or more. The simulation setup is therefore modified to catch the behavior of the system in less than 10 ms.

The setup for the clock pulse generator is shown in Figure 5 below. The clock is running at 2 MHz (well below 100 MHz, but still short enough to fit a large number of clock cycles within 10 ms).

The up/down pulse generator should have a period of 0.00003, and a pulse width of 80%.

The reset pulse generator should have a period of 0.00006, with a pulse width of 90%.

Finally, the dp pulse generator should have a period of 0.00009, with a pulse width of 50%.

All the settings of the pulse generators above are examples. Experiment yourself with these, and see what happens.

Set the simulation time to 0.0001 in the simulink toolbar at the top of the model window. The default value (10.0) is too large, as it will require a couple of hours to complete the simulation. It will also create gigabytes of testbench data if run for that long time.

Press the start simulation button to run the simulation. Check with the scope what happens (open it by double clicking on the symbol), and try to evaluate if the design works as expected.

2.3 VHDL model creation with support for the built-in logic analyzer

Open the SignalCompiler block. Make sure that the “Use board block to specify device” is selected.

Select the Signaltap II tab and enable the signaltap.

The design can be analyzed, synthesized and fitted into the FPGA either from the “simple” tab pressing the compile button, or taken step by step using the “advanced” tab with the individual buttons for each step.

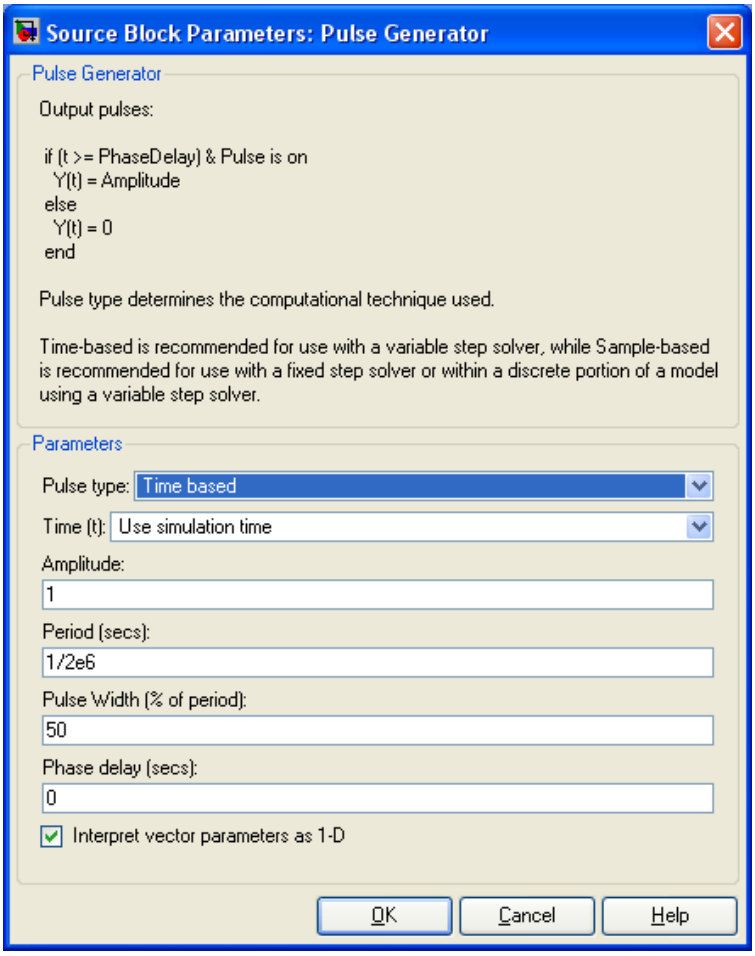

Figure 5 Clock pulse generator dialog

The individual steps are more useful when there is a problem in the model, as it will not try to run all steps at once.

Many of the steps are complicated and time consuming to compute. Always wait for task to complete before closing the signal compiler window.

Run all the steps from analysis to fitter. This will take some time, and a lot of information messages will be given.

2.4 Simulation of the VHDL model using ModelSim

The testbench block allows you to create a VHDL model and compare the simulation results of the simulink model with the generated VHDL model.

Open the testbench block. The advanced tab allows you to take individual steps in the process, such as VHDL generation, modelsim simulation, simulink simulation, and comparison. The simple tab allows all steps to be taken automatically.

Use the simpel tab and press “compare against HDL”. This will first generate the VHDL code including a testbench. Then is the simulink simulation run while the input and output values are logged. These values will be used as inputs to the VHDL model and for comparison with the VHDL model output. The is the VHDL model simulated using Modelsim. Finally is the result of the modelsim run compared with the simulation results from Simulink, and any differences are presented.

Note that the simulation may take some time.

It is also possible to create new test patterns, and to look at internal structure and nodes in the design using the advanced tab entries. If you select “launch gui” then the modelsim simulator will start with your design loaded and executed. This can be very useful if the comparison between the two models does not match.

2.5 Synthesis of the VHDL into FPGA configuration information

Open the Signal Compiler block and press the Compile button in the Simple tab. Wait for the compilation steps to complete. A lot of information about resource usage, clock speed etc. can be found in the transcripts. This step is one of the most time-consuming ones.

The synthesis step creates a directory called lab1_dspbuilder that contains the VHDL files, other configuration files, and the generated FPGA configuration file (.sof). Messages generated by the synthesis as well as stastics about resource usage etc can be found in various .rpt and .summary files in the directory. These files are very useful if you have problems with the synthesis of a design.

2.6 Programming and test of the hardware

In the setup here, the computer used to program the FPGA is a separate machine from the one you are running Simulink on. The connection to the FPGA board is done through a USB dongle to the programming machine, and then through the network to the machine running Simulink.

Quartus II is required to run once to make the computer running Simulink aware of the other hardware-connected computer. The following steps are therefore required:

Start Quartus II programmer using *module load prog/quartus/13.0.1* followed by *quartus_pgmw*. Press the *hardware setup* button. In the dialog window select the *jtag settings* page. Type in the computer name and password as given by the lab instructor.

Close and quit Quartus II Programmer. The machine now knows about the available hardware computer server, and this step is not needed to be rerun until the computer is restarted.

Before programming the device, make sure that no one else is using the board. Then press the button named “Scan jtag” to get the list of connected hardware. Select the proper connection. The entry to the right of the button should now show “2S60”, which is the name of the FPGA on the board. Then press the 2nd button called "program" to configure the FPGA on the board. The blue light in the USB JTAG dongle connected to the board should light up, and the new design will be downloaded within seconds.

Try out the hardware, pressing the pushbuttons. Do you get problems with multiple bounces when pressing the pushbutton?

2.7 Use of the built-in logic analyzer

The next step is to try and measure signals inside your design. This is accomplished through the use of the Signal tap block. Open the Signal tap block. Set the triggering information to trigger on falling edge of the enable pulse generation by first selecting the signal, then select falling edge in the list to the right, and finally pressing the change button. The other inputs should be set to don't care. Tell the system to start its wait for a falling transition on the enable pulse by pressing the acquire button. Go to the board, and press on the pushbuttons on the board. You should now see that there now is a new set of waveforms of the counters and the trigger signal presented on the screen. Note that the trigger only defines when the sampling of data should start, but is not used to define how often data is sampled. The sample rate is the global system clock (100 MHz), and it is therefore difficult to catch more than one transition of the SW4.

3 Troubleshooting

3.1 The SignalCompiler is unable to detect the jtag cable

The SignalCompiler window is used to define the hardware to program. See section 2.6 on how to run the Quartus II software and define the available hardware boards.

3.2 I can not open the DSP Builder block/Signal tap block

Sometimes the DSP builder block and the Signal Tap block does not seem to open when double-clicking in the Simulink schematic. The windows does not show in the taskbar.

The reason may be that the windows are already open, but are hidden behind some other window even if it is not shown in the taskbar.

The solution is to move the top windows down on the desktop to see if there is any windows behind them. Once the open DSP Builder or Signal tap window is found the taskbar is usually updated.

3.3 Modelsim complaines about missing ".salt" file when trying to run

When trying to execute the tcl script in modelsim, the simulation stops with error messages similar to 'Failed to open VHDL file "DSPBuilder_lab1/SW5.salt"'.

The reason is that the .salt files are only generated when running the Simulink model in Simulink, not when generating the vhdl code.

The solution is to open the Simulink model and run it once.

Please note that the .salt files are not automatically updated when you update the simulink model. Always remember to run the Simulink model before running the modelsim model.

3.4 I can not run signal tap (analyze gives error "...Analysis was unsuccessful")

Sometimes the signal tap application returns an error when trying to do the analysis.

The reason may be that you have a space character in your path to the design. The system does not accept spaces in the directory path.

Solve this by moving the design into a directory path without any space in it.

3.5 The hardware does not have the clock running/The reset seems to be active all the time

Sometimes any clocked element such as flipflop, registers, counters, etc. seems to be in a constant reset state. This occur even if there is no reset input on the device, and/or the reset signal is inactive.

This behavior may be caused by an active global reset. The default setting in the board definition block is to select "any" pin. The selected pin may then be any I/O pin on the board, and that may then be a pin that is always active.

The solution is to define the exact pin to use. Select the PIN_AG19 for the global reset. This makes the reset input connected to the pushbutton labeled "CPU Reset" located next to the FPGA.

3.6 Saving the model generates a lot of Java errors

The Simulink model can be saved in two different formats (.mdl and .slx). The DSP Builder tools only supports designs in .mdl format. Make sure the file is saved in this format.