ENGR 325 - Fall 2019
Lab 1: Building Systems with Quartus' Qsys Tool
Due Date: Beginning of Lab on Sept 12, 2019

Objectives

After completing this lab you should:

- be able to generate a complex system using the Altera Qsys tool and
- be able to use the Signal Tap Logic Analyzer tool.

Overview

In this lab, you will be building a system with input and output devices and the NIOS processor on the Altera DE2 development board. You will be learning to use one of the tools that comes with Quartus.

Background

Modern CAD tools have powerful tools or applications that enable engineers to build complex computer systems very quickly and easily. You will be learning to use the Qsys tool which can build such a system.

Pre-Lab Exercise:

Pre-Lab: Review how to write clocked processes in VHDL. Next, design a clock divider circuit in VHDL. Your design should take the 50 MHz clock and divide it by a factor of 10 to provide a 5 MHz clock in addition to the 50 MHz system clock. The 5 MHz clock is needed by a logic analyzer module that you will be adding to your system later in the lab. This clock divider should be written using a clocked process. The process can simply use a 0 to 4 counter that toggles the state of a pin like GPIO_0(0) every time the counter resets itself back to 0. This is just your basic VHDL clocked process that implements a counter, based on a clock and a reset signal. A template file called "ClockGenTemplate.vhd" is provided for you as a starting point. A modelsim "do" file ("do.do") is included for testing so that you can make sure the period of the slow clock is exactly 10 times the period of the 50 MHz clock. Screen capture the signals and measure the period of the input clock as well as the output clock. Is the output clock's frequency one-tenth of the input clock's frequency? Hand write your analysis on the screen capture printout and bring it at the beginning of lab to show your instructor.

Lab Exercises:

Part 1: Work through the Qsys tutorial provided in the Engr325 folder on the library drive (also on moodle) "Introduction_to_the_Altera_Qsys_Tool.pdf". You should only go through the Qsys tutorial once. The overall steps of the tutorial are exactly what you need to follow, but you will name your system Lab1 instead of lights and the details of the system are different as noted in the yellow comments. It is important that you download the pdf and open the file with Adobe Reader so that you will see several comments boxes provided on the side of the document

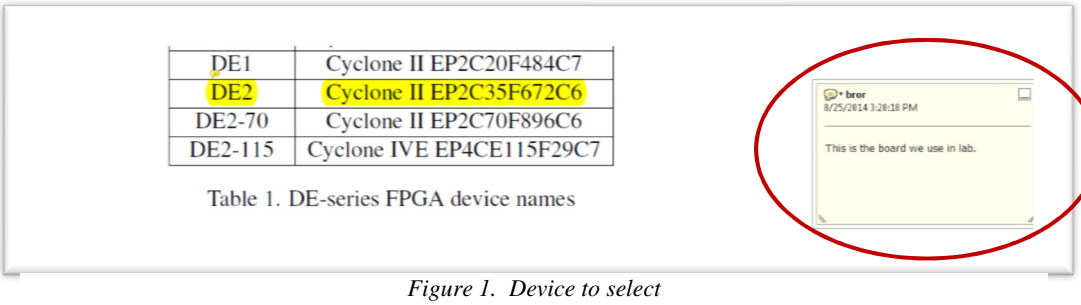


Figure 1. Device to select

that look similar to what is shown in Figure 1, found on Page 5 of the tutorial. If you don't pay attention to the comments, you system will be incorrect for the rest of the lab.

Your system will be slightly different from the one described in the tutorial. Create a new folder for the project

called “Lab1” and name the Quartus project and entity “Lab1”. As always, avoid any spaces or unusual characters in all folder and file names, including the folder names of those folders above the one you are creating. You will build a portion of the final system using the Altera Qsys tool. That sub-system designed in the Qsys tool should be called “nios_system.” Note that you will be asked for that name when you either save or generate within the Qsys tool. The “nios_system” that you build should include:

- the standard clock component which is pre-loaded into the system,
- a NIOS II/s CPU component
  - The middle option of the 3 CPU variants,
- 24 kBytes of on-chip RAM
  - Renamed “RAM” and having a base address of 0x0000,
- an RS232 UART component that will be used in the next lab
  - Found in the university program section (not the “JTAG UART,”) renamed “Serial,” configured for 115,200 baud, selected for interrupt #1, and having a base address of 0x8020,
- and a 16x2 character display LCD component
  - Found in the university program section, renamed “LCD,” and having a base address of 0x8000.

The RS232 and LCD components should have I/O signals exported via the “conduit” and these signals should be renamed as “rs232” and “lcd” respectively. Note that you may have to re-locate the JTAG Debug Module to another address. Your Qsys system should look similar to what is shown in Figure 2. Take a screen capture of the final Qsys configuration screen.

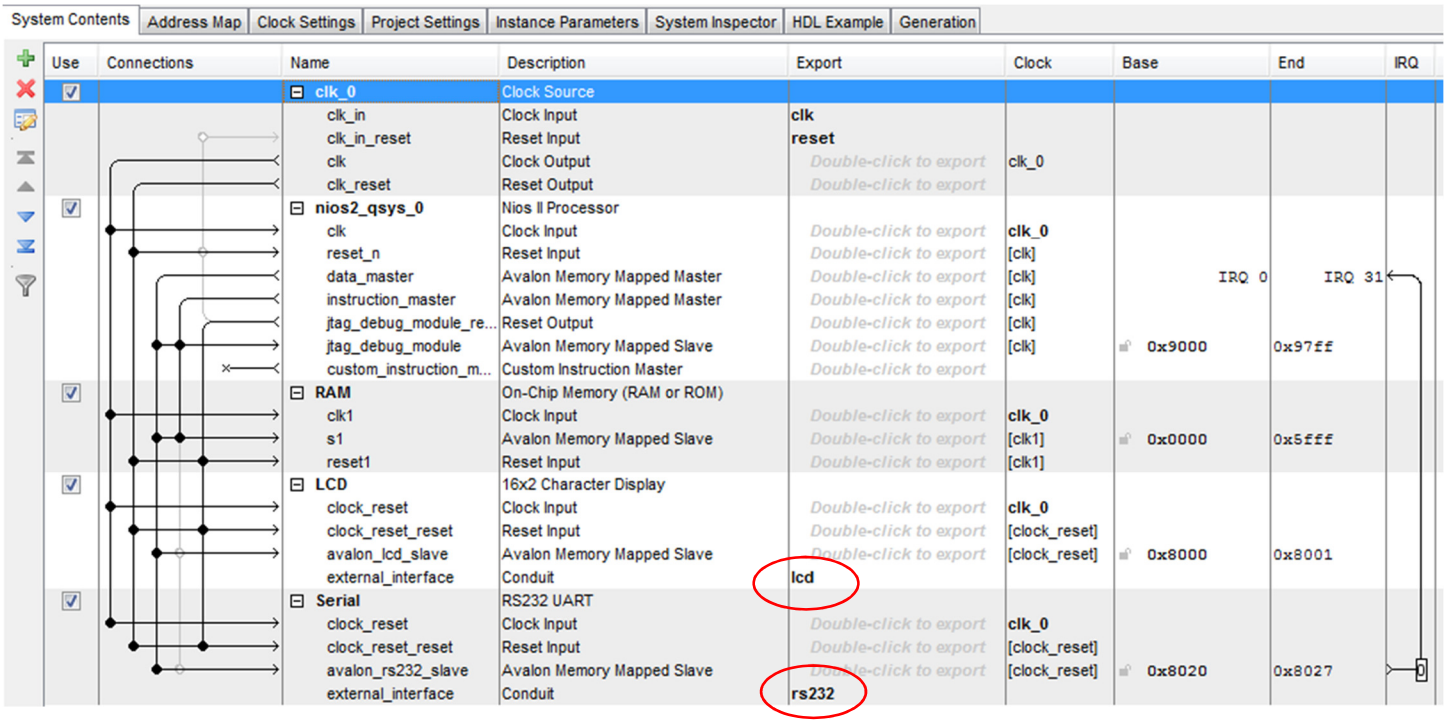


Figure 2. Qsys final configuration

The sample code provided by Qsys on the “HDL Example” tab is very similar to the code you will find on page 24 of the tutorial. The sample code is correct for your system compared to what is shown on page 24 since you have different inputs and outputs; however, the sample code does not show you where it fits in the top-level Lab1.vhd file – that is what page 24 shows.

Finally, after updating the top-level “Lab1” entity/architecture with appropriate references to the nios_system, run just the Quartus “analysis and synthesis” compilation steps on your overall project for now to make sure your VHDL code is correct.

Part 2: Add the clock divider module to your top-level VHDL file (instantiate the component in a way similar to how you instantiated the nios_system component) to divide the CLOCK_50 signal by 10 so that a logic analyzer has a 5 MHz clock available and can then sample at 5 MHz instead of 50 MHz. Connect this slow clock signal as an output port of the top-level entity and call it "GPIO_0(0)". Next, setup your pin assignments using the "DE2_pin_assignments_orig.csv" file as a starting point. There should be exactly one entry in your pin assignment file for every signal listed in the top-level port listing. Finally, run just the "analysis and synthesis" compilation steps for now so that the new clock signal is recognized.

Part 3: Review the "Getting Started with Signal Tap Logic Analyzer" document. Add a Signal Tap II logic analyzer with an 8k buffer to your system and configure it to watch the receive (RxD) and transmit (TxD) UART signals as well as the LCD interface signals (DATA, RS, EN, and RW). Set it up for basic OR triggering on the following signals: LCD_EN, UART_RXD, and UART_TXD. The logic analyzer should use the slower 5 MHz clock signal instead of CLOCK_50 as its clocking input. (If you clock it at 50 MHz, the window of time you capture would be so short that you wouldn't see much useful information.) The signal tap configuration should look like what is shown in Figure 3. After adding the signal tap unit to your design, run a complete compilation of the system, making sure all pin assignments on the FPGA are correct (review them in the Pin Planner tool). Create a screen capture of all warnings generated during the compilation process.

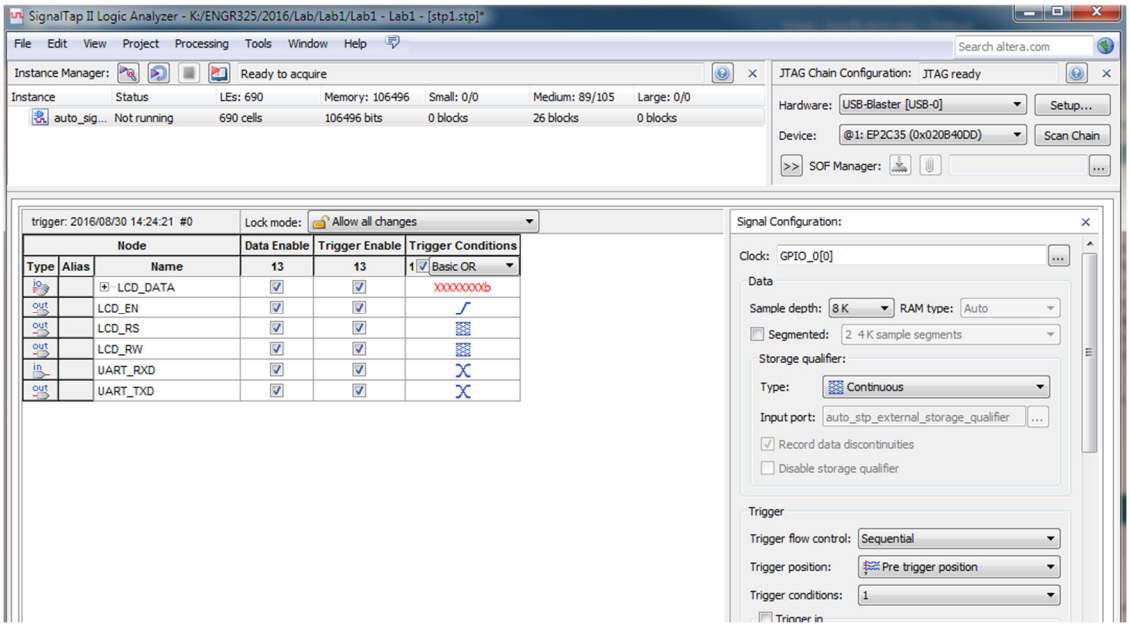
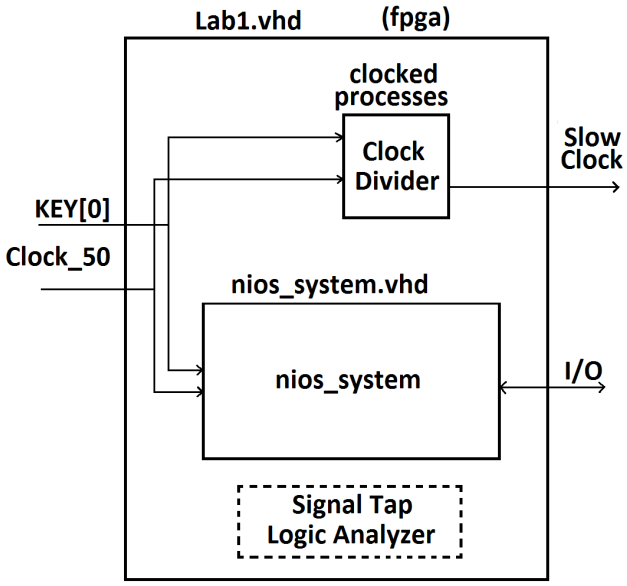


Figure 3. Signal Tap Logic Analyzer configuration

The signal tap component is added automatically during the compilation process once you have added it to your Quartus project; it is not specified in your Lab1.vhd file since it is not really a part of your top-level design. However, it does take up part of the FPGA's resources.

Below is a block diagram showing what the final system should look like.



Hand In:

Create an MS Word document (or other document tool) with the following items. Either embed the "hand-written" comments in the document or print the document and hand-write the comments on the printout.

Pre-Lab: (10 pts)
- Modelsim screen capture with notes and calculations that confirm the 5 MHz frequency of your ClockGen output
- Printout of your VHDL ClockGen code, following the guidelines in the "Formatting Software Printouts" document

Lab Part 1: (5 pts)
- Screen capture of the final Qsys configuration page

Lab Part 2: (5 pts)
- Printout of your top-level VHDL code, again following the guidelines for printouts

Lab Part 3: (5 pts)
- Screen-capture of all Quartus compilation warnings for your top-level design