BEE 271 Digital circuits and systems
Autumn 2020
Lab 0: Learn Tools, and simple IO

1 Objectives

This lab has three components:

- Download and install Quartus Lite
- Introduction to Verilog
- Create a simple program that reads switch inputs and displays results on matching LED outputs, including 7-segment hex displays
- Download the Verilog code to the DE1-SoC, and test it.

By the end of the lab, you should have demonstrated creating, downloading and testing the simple IO code.

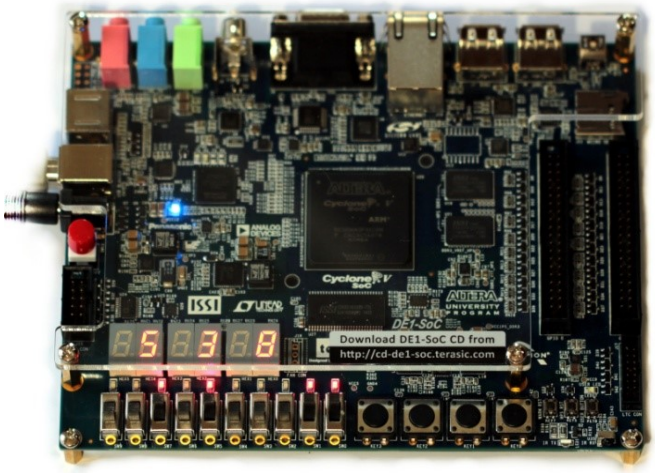

Figure 1, DE1-SoC Kit

2 Required: locate and install software tools

To do this lab, you will need a Terasic DE1-SoC board (provided in the lab kit) and a PC with the following installed:

1. Quartus Prime Lite Edition, v19.1 or v20.1
2. Altera's USB Blaster driver which is included.
3. Intel System Builder utility, from the install package (DE1-SoC CDROM, which should be installed in the C:\altera directory.
4. DE1-SoC User Manual (also on Canvas)

Quartus Lite is a Windows application. There is also a Linux native version.

It is NOT a direct MacOS Application. Choices if you are MacOS user:

- Quartus will run under Windows emulation on MacOS: BootCamp, VMWare, etc. These emulation programs need a Windows OS license; obtain one from UW Bothell IT dept.
- Quartus will run on a Windows notebook borrowed from UW Bothell IT dept.
- Find a lab partner (always a good idea) who has a Windows PC.

3 Preparation for this lab – time consuming

Locate the ~6GB compressed TAR file. Choices:

- Intel's web site (20.1): https://fpgasoftware.intel.com/
- My Google Drive (19.1): https://drive.google.com/file/d/1oyc_S8vMZ_-MXPwWQOje2h0KvYV9wbkN/view?usp=sharing

Note: either the Intel .tar file or the 19.1 .tar file on my google drive is 5.9 GB:

- It is too big to load onto Canvas (limit about 2GB)
- It takes at least 30 minutes to download, depending on your networks speed
- After download, you must unpack it with WinZip, which takes a long time
- You will need WinZip application. Use the eval package or consult UW IT.
- The WinZip unpacking process took over half an hour on my i7 PC
- The software installation process took a long time, too.

In other words, this process may take several hours.

Canvas will have these resources:

- Installing Quartus
- Intro_to_Quartus
- Tut_quartus_intro_verilog

For the Intel site, it says "Quartus Prime Lite Edition". Select edition: Lite. Accept 20.1

Choose Individual Files. Choose More, to see what the choices are:

1. Download Quartus Prime software, and any other software products you want to install, into a temporary directory.

2. Download device support files into the same directory as the Quartus Prime software installation file.

3. If you want to use add-on software, download the files from the **Additional Software** tab. Save the files to the same temporary directory as the Quartus Prime software installation file.

4. Run the **QuartusLiteSetup-20.1.0.177-windows.exe** file.

All software and components downloaded into the same temporary directory are automatically installed; however, stand-alone software must be installed separately. "

You may find the Intel FPGA Software v20.1 installation FAQ instructive. We will need support for the Intel (Altera) Cyclone V FPGA, which is at the heart of a DE1-SoC.

You must then launch the Quick Start Guide.

4 Quartus Demonstration Program

The first task is to familiarize students with the process of building and testing Verilog FPGA programs using Quartus from Intel (formerly Altera).

Find the following:

- The "USB Blaster device driver will be part of the installed software.
- The System Builder utility (DE1SoC_SystemBuilder.exe)
- Connect:
    - AC power to the DE1-SOC power adaptor
    - DC power to the J14 connector on the DE1-SoC
    - USB on lab computer to J13 on the DE1-SOC kit (note the number on the kit)

4.1 Demonstration Procedure Overview

1) Turn on the DE1-SoC by pushing the red button next to the J14 power connector. If the board is working, it will run a simple 'it works' procedure:
    - The 6 HEX LED displays will all cycle through 16 states: 0, 1, 2, … A, b, C, d, E, F,
    - The 10 LEDs with display: all OFF, alt LEDS on & off, all ON,
2) Use the DE1-SoC System Builder tool to create a skeleton project with the inputs and outputs you need: Switches, LEDs and HEX displays
3) Open it in Quartus and add the skeleton Verilog file to the project. (System Builder generates this file but doesn't add it to the project automatically.)
4) Compile your project. This code has been tested: warnings, but no errors.
5) Program the device using the USB Blaster.
6) Verify that everything works as expected.

4.2 Run the System Builder

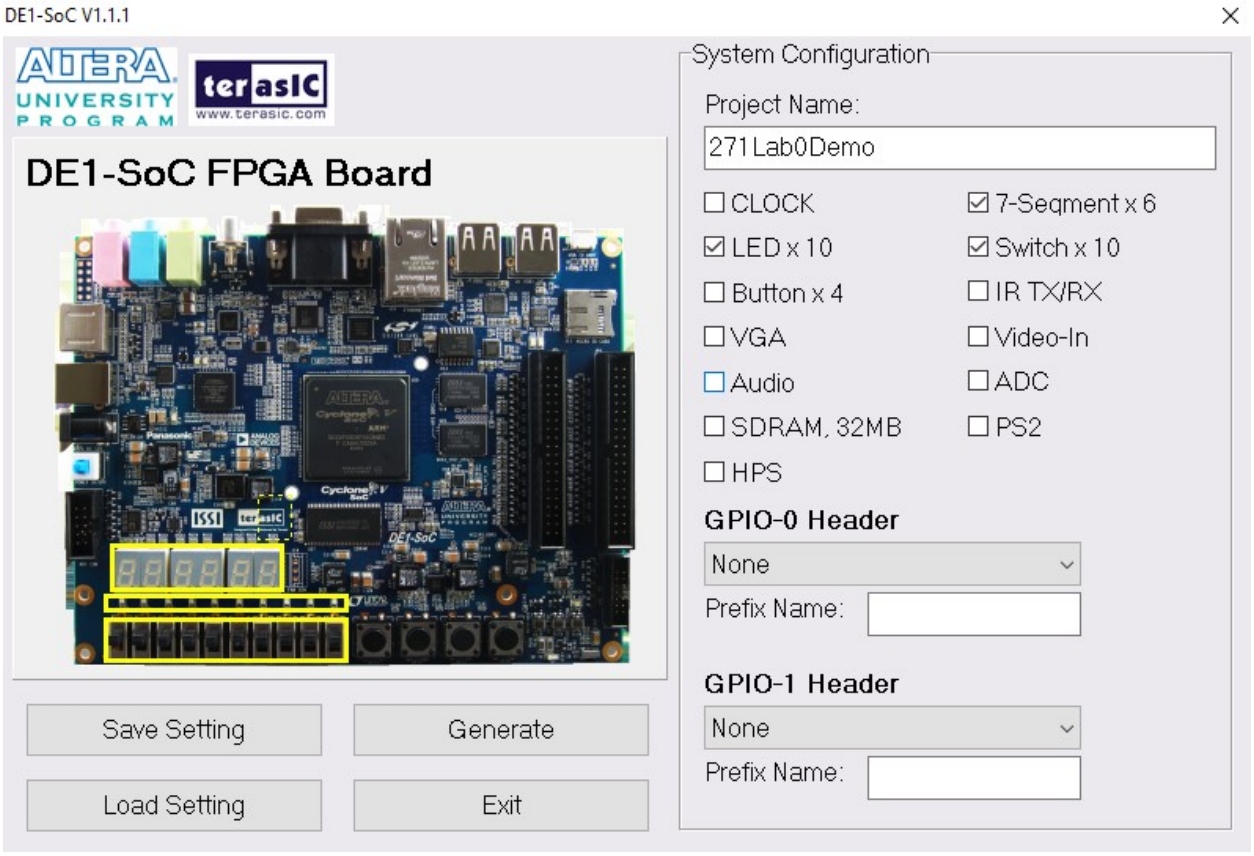

Figure 2 DE1-SoC System Builder window for selecting inputs and outputs.

Find and launch DE1SoC_SystemBuilder.exe

It will start up with all options boxes checked. Uncheck all of them except: 7-Segment x 6, LED x 10 and Switch x 10.

Select Generate. The program might generate error messages that you might not be authorized to save in some locations. I allowed it to save where it defaults to:

C:\ … \Tools\SystemBuilder\CodeGenerated\DE1_SOC\271Lab0Demo

4.3 Capture the resulting files

Located and copy the files to the directory you wanted to move them, or leave them where they are. I used: \..\Lab 0\BEE271L0Demo

The four essential files:

1. BEE271L0Demo.qpf file = Quartus Project File
2. BEE271L0Demo.qsf file
3. BEE271L0Demo.sdc file
4. BEE271L0Demo.v file = Verilog file

From a file explorer view, the icon for the QPF file will look like this:

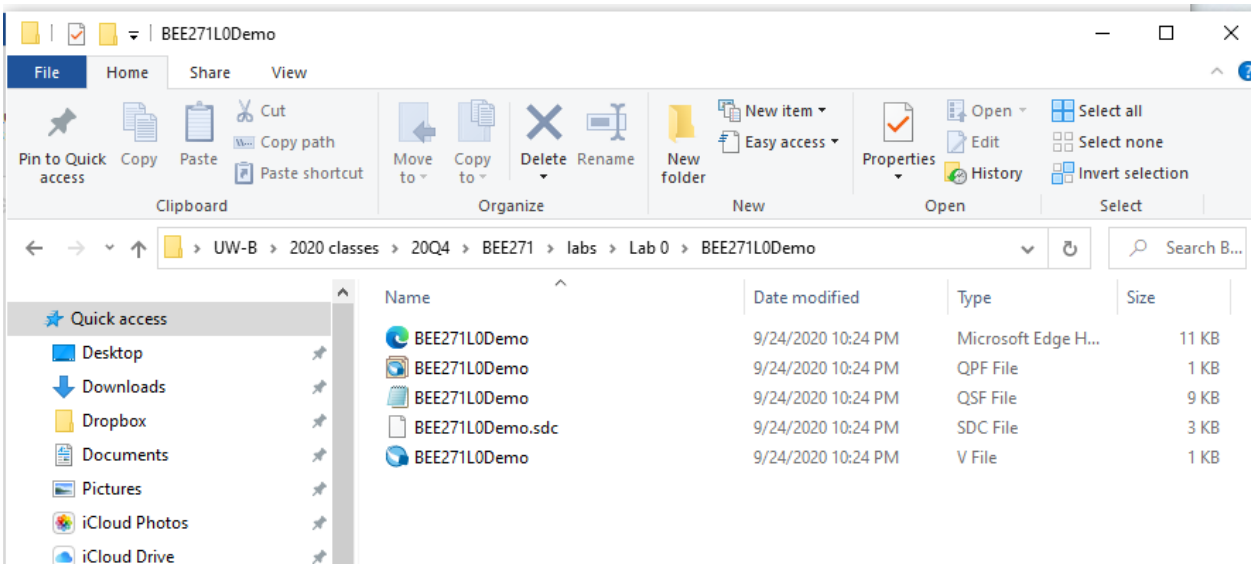

Figure 3, Windows file explorer view of a demonstration subdirectory

4.4 Launch Quartus

If Quartus is installed, double click on the QPF file.

Windows will take several 10s of seconds to open. It will produce a window inviting a software upgrade; it will spontaneously close that advertisement. Eventually it will show a screen like this:

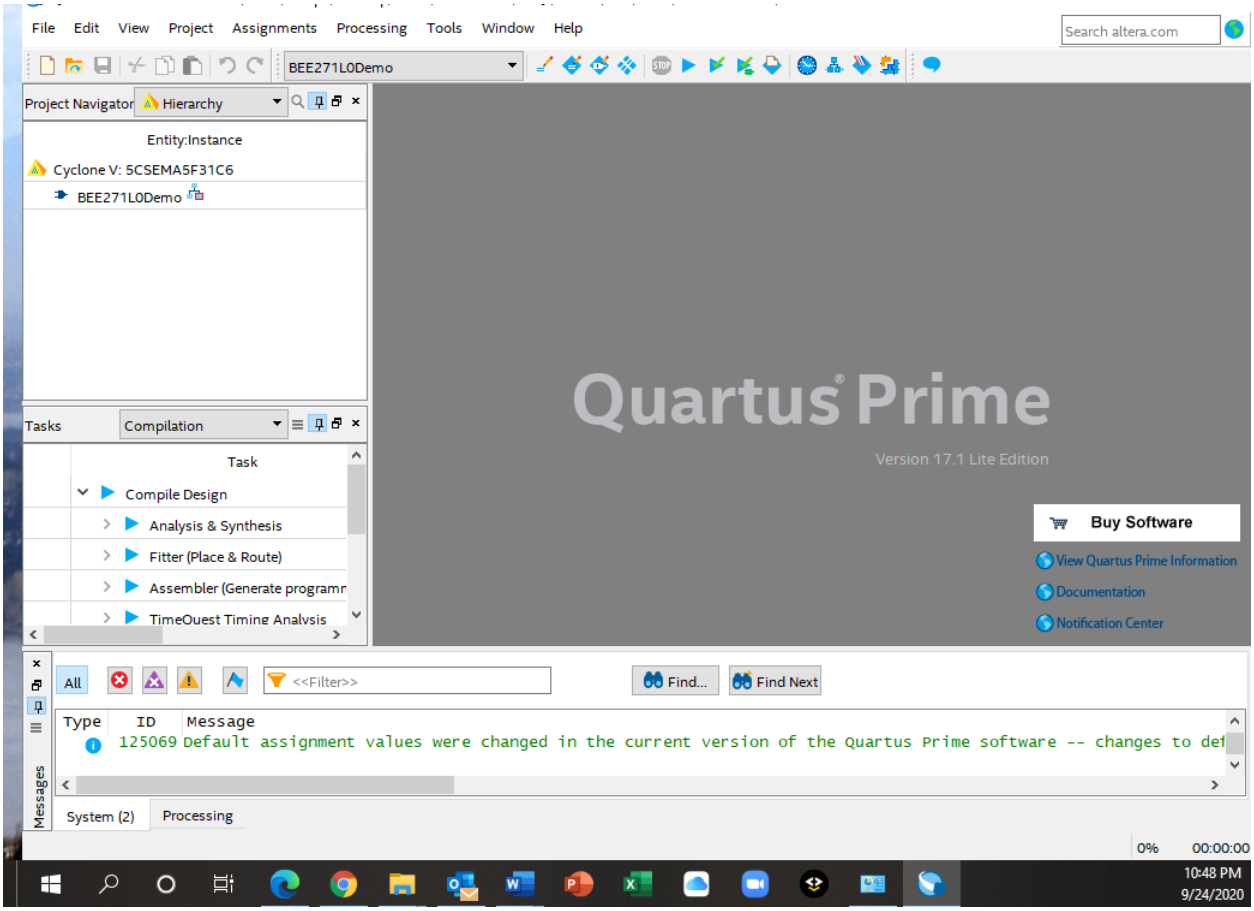

Figure 4, Opening Quartus Prime window.

Double Click on the "BEE271L0Demo" under Cyclone V: 5CSEMA5F31C6

That will open and display the simple Verilog (.v) code, replacing the grey "Quartus Prime" subwindow.

```
//=======================================================
//  This code is generated by Terasic System Builder
//=======================================================

module BEE271L0Demo(

	////////// SEG7 //////////
	output		[6:0]		HEX0,
	output		[6:0]		HEX1,
	output		[6:0]		HEX2,
	output		[6:0]		HEX3,
	output		[6:0]		HEX4,
	output		[6:0]		HEX5,

	////////// LED //////////
	output		[9:0]		LEDR,
```

```
        ////////////// SW ///////////
        input               [9:0]               SW
);

//=======================================================
//  REG/WIRE declarations
//=======================================================


//=======================================================
//  Structural coding
//=======================================================


endmodule
```

Note: this data is copied and pasted. Quartus will color code several elements:

- All comment lines begin with doubled // characters. They will be green.
- Crucial special works are in blue, including module, endmodule, input and output
- Recognized numbers will be red.

5 System Builder source code explanation

The top few lines are self-explanatory. Recommendation: you will replace them with comments that identify your work, e.g.

```
//BEE271 Fall 2020
//Lab 0 Demonstration and Test Verilog Program
//1) map switches to LEDs
//2) replicate switches to LED segments
//Joseph Decuir, 2020-09-25
```

The crucial lines begin with the module declaration. It is followed by the name of the module "BEE271L0Demo" and an opening parenthesis character (.

It is followed by a series of declarations of inputs and outputs. As is, all of these newly named objects are bit vectors. "[6:0]" means that there are 7 bits, numbered from [6] to [0]. "[9:0]" means that there are 10 bits, numbered from [9] to [0].

Each declaration is followed by a comma (,) character except the last one. The module declaration is terminated by a closing parenthesis) and a semicolon character.

(This module has 62 ports: 6 x 7 + 2 x 10 = 62.)

The next three lines are basically a reminder that traditional Verilog structure then has a set of internal variable definitions. In Verilog, these are: registers (REG), wires or parameters.

We will not need any definitions yet. We will do those in Lab 1.

The following three lines are reminders that there should be some structural code that defines behavior. We will specify that in section 6.

The module declaration is terminated by the endmodule statement.

The starting code would compile, but do nothing. The Quartus compiler might complain with warnings, but no errors.

6 Create simple code

6.1 Construct new structural code

This exercise has two coding steps:

- Copy the slide switch states directly to the LEDs
- Copy 7 of the slide switch states to each LED segment, in parallel

This code needs the assign command:

```
assign LEDR = SW;          // copy Switch states to LEDs

assign HEX0 = SW[6:0];     // this copies 7 switches to on 7-segment display
```

To test your installation, enter those two lines of code + your own header.

Then, save the file, either with File – Save, or the 'save file' icon.

6.2 Compile the code

There is a row of icons on the third row shown in Figure 4. The next step is to compile this code. A clue: click "Processing". The second choice is "Start Compilation". It illustrates that the icon is blue triangle pointing to the right. It also illustrates the keyboard shortcut Ctrl+L. Choose one of those methods to start compilation.

That process will open up a new window with a Compilation Report. I attach a completed screen in Figure 5.

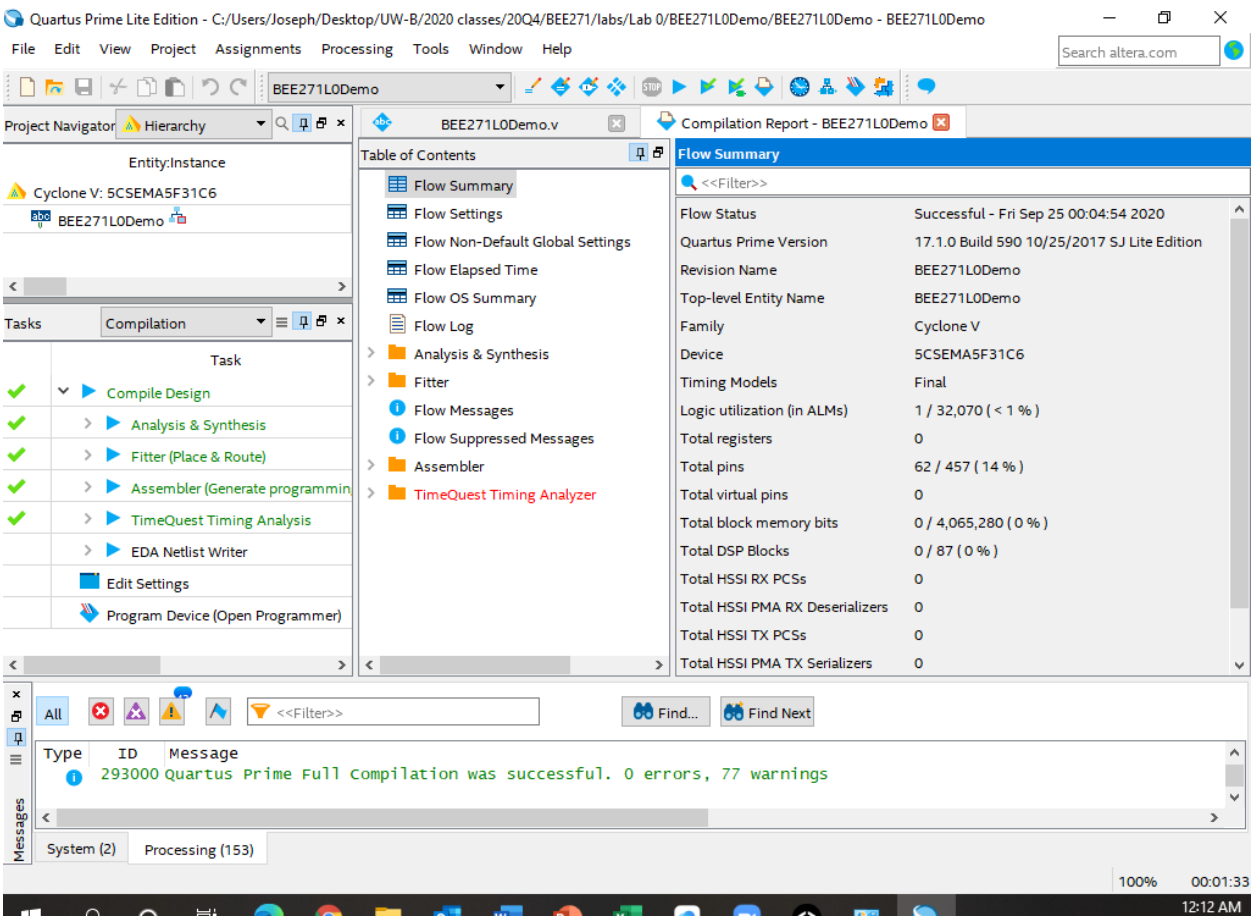

Figure 5, Quartus after successful compilation

Note that I had used the mouse to extend the Compilation window on the left higher, so you can see all the compilation steps completed, followed by: EDA Netlist Writer, Edit Settings and Program Device.

If compilation was unsuccessful a series of error messages and/or warnings would show up below. In fact, we have 0 errors but 77 Warnings! Example: there will be several "10034" Warnings that the output ports HEX1 through HEX5 have no drivers. Those are all true, but they can be ignored for the time being.

6.3 Download the code

In Quartus, open the Tools menu. It will show many choices. One, in the middle, is "Programmer". Note what the icon looks like.

Launch the programmer. The first, time you do, it will not yet be ready.

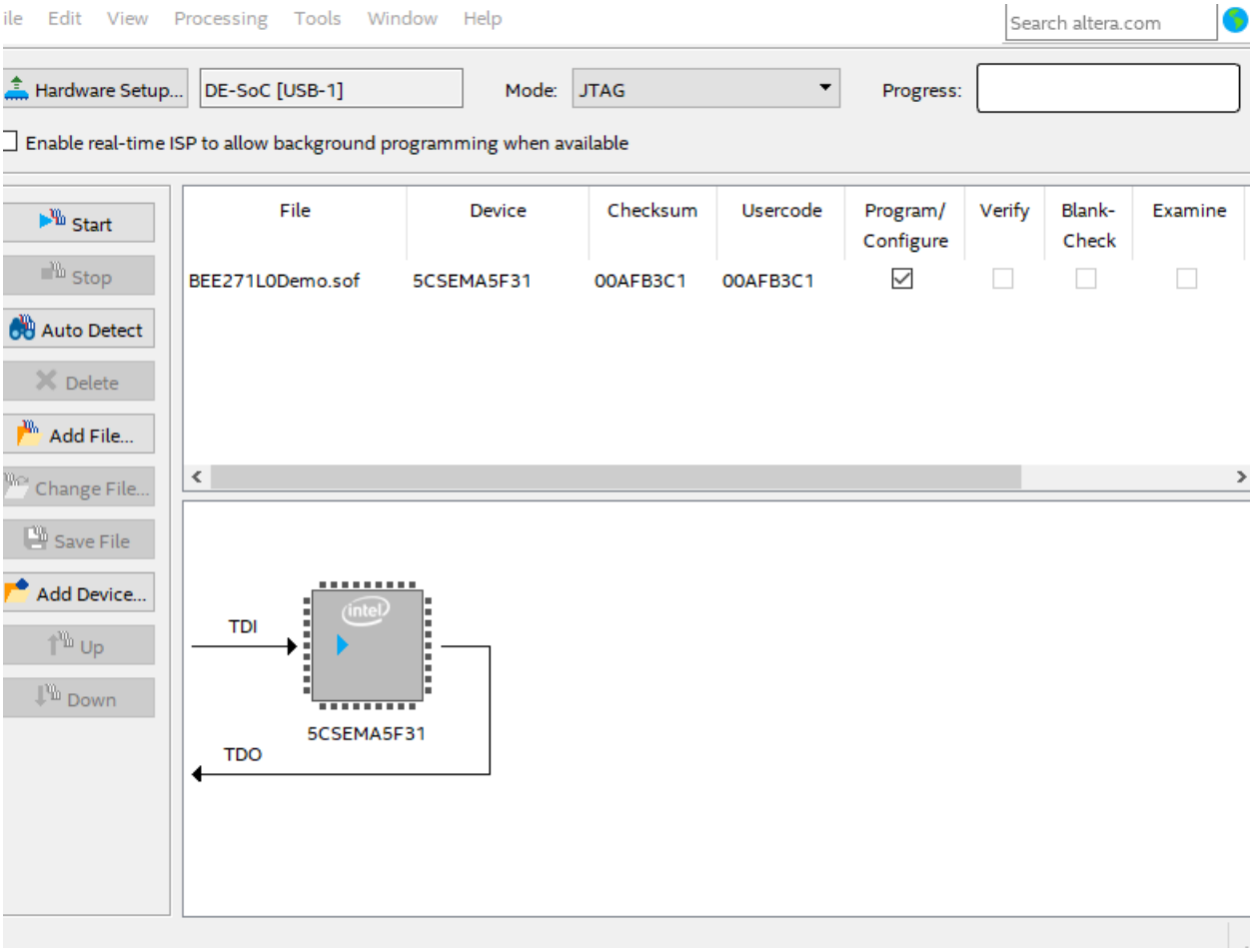

Figure 6 Initial programmer window.

It does correctly reference the BEE271L0Demo.sof file. It is missing a reference to the SOCVHPS (System on a Chip Hard Processor System); the Cyclone V has a pair of 32-bit ARM CPUs, which our code will not use. Choose Autodetect.

The programmer will note that it found several devices with a shared ID. Choose 5CSEMA5. That added the SOCVHPS and the 5CSEMAS5, but deleted the 5CSEMA5F31.

Next, Add file. Choose BEE271L0Demo.sof. That will restore the 5CSEMA5F31.

Last step: remove the 5CSEMA5. Click on it, and then choose **X** Delete.

Result should look like this:

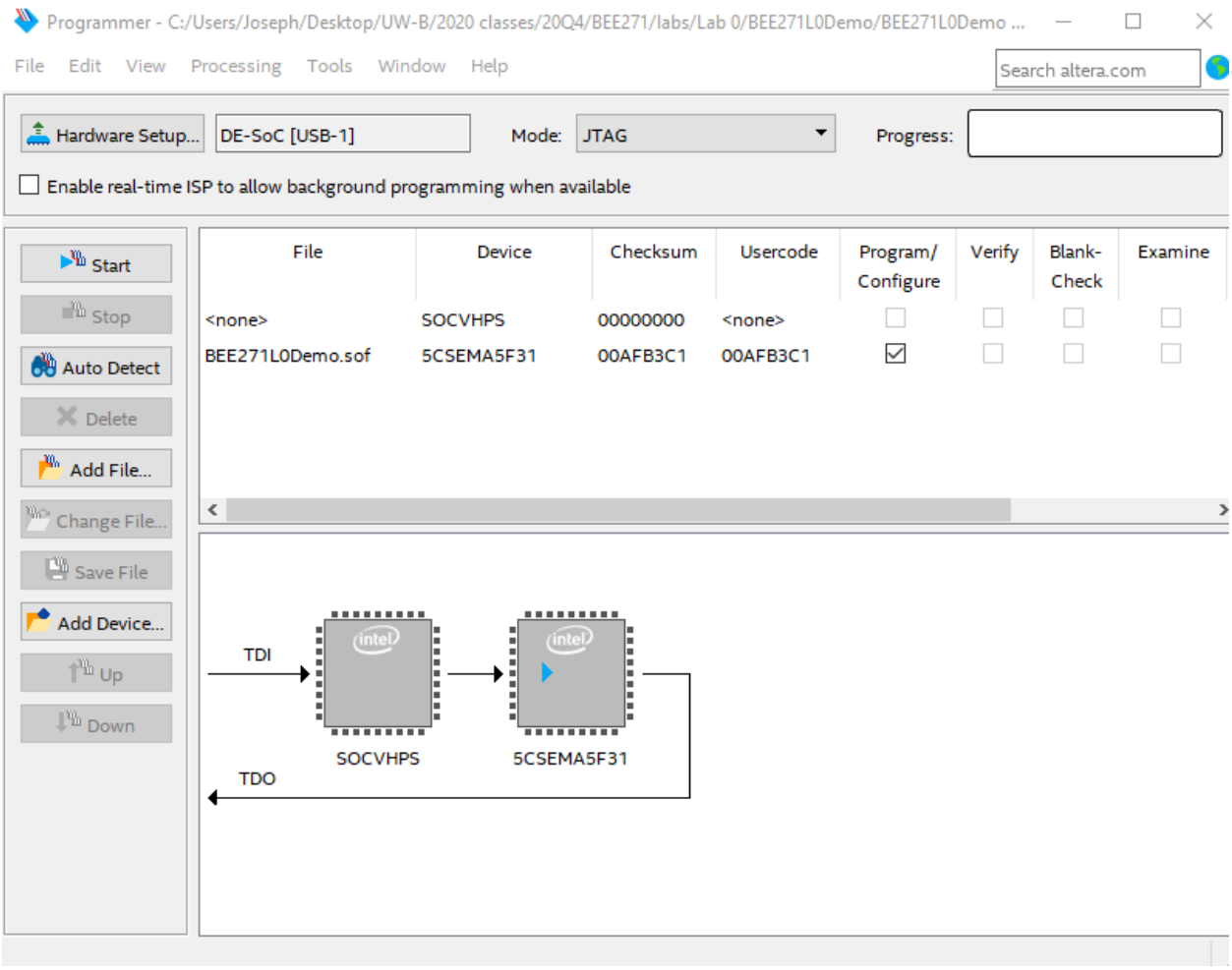

Figure 7 FPGA Programmer ready to download code

Check: at the top of this figure, after Hardware Setup, it says DE-SoC [USB1], Mode "JTAG". Progress will be empty.

Find the Start button, and launch it. Progress should turn green "100% (Successful)".

7 Test the code

Initialization: start with all the switches in the lower position. That should:

- Turn off all 10 LEDs
- Turn on all 7 segments on each of the 6 7-segment hex displays

First test: toggle the rightmost switch to the ON position:

- The rightmost switch (SW[0]) should turn on the rightmost LED (LEDR[0])
- It should also turn off the top segment on the rightmost hex display (HEX0[0])

Second test: toggle the second switch, SW[1], turning off the rightmost switch:

- The second LED will light up, LEDR[1]

- The hex display will look like a number 6, toggling HEX0[1].

Continue testing: toggle more switch bits, from right to left:
- The next 5 bits, SW[2] -> SW[6], will progressively toggle LEDR[2] -> LEDR[6]
- These will also toggle HEX0[2] -> HEX0[6].
- Example: if you assert only SW[4], you get a number 9
- Example: if you assert only SW[6], you get a number 0
- SW[7] -> SW[9] only toggle the LEDs

Experiment #1: find out which switch combinations generate each hex character:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E and F
You will use that information in Lab 1.

Experiment #2: change the code so that some combination of switches drives all the 7-segment displays. Recompile it, download it, and observe changes to those hex displays.

Last test: assert all 10 switches
- All 10 LEDs should light
- The rightmost hex display should be blank.

Notice that the LEDs are active high (they go on when the switch is on) but that the display segments are active low (they go on when the switch is off).

8 Report via Canvas assignment

Write a simple report that explains what you did and what you learned:

1. Identification of the team member(s) e.g. on the report and in the code
    - We encourage teamwork. IMO, business is a team sport
2. Abstract e.g. a paragraph
3. Introduction
4. Procedures
5. Results:
    - source code v file
    - screen shot of test results and/or
    - photo of a representative test case
6. Discussion and Conclusion
7. A bibliography of any resources you consulted, as well as citing any assistance you received from your classmates.
    - Example: you might have help from classmates, or you helped your classmates
8. Are there any problems to report or suggestions/hints for future students.
    - Example: any missing crucial steps to install the software
9. Please make one composite PDF document.
    - It is very hard when I have to unpack your zipped files, run your code to replicate the work, and then grade it. I have had other students give me work that takes 20 minutes to grade. This doesn't scale…