# CSE P567 - Winter 2010

## Lab 1 – Introduction to FGPA CAD Tools

This is a tutorial introduction to the process of designing circuits using a set of modern design tools. While the tools we will be using (Altera's Quartus II CAD tools) are geared specifically to FPGAs, they are very similar to the tools used to design ASICs (Application-Specific Integrated Circuits).

This tutorial will proceed as follows:

- Typical CAD Design Flow
- Opening and viewing a Quartus design project
- Editing Verilog files
- Compiling the project
- Programming the FPGA (downloading the circuit to the FPGA board)
- Testing the circuit

This tutorial is derived from a longer and more detailed tutorial on the Altera Quartus II design tools, which is available on the class Resources Web page. Some of the screen shots may vary slightly based on the version of the tools being used.

## Typical CAD Design Flow

Computer Aided Design (CAD) software makes it easy to implement a desired logic circuit by using a pro- grammable logic device, such as a field-programmable gate array (FPGA) chip. A typical FPGA CAD flow is illustrated in Figure 1.

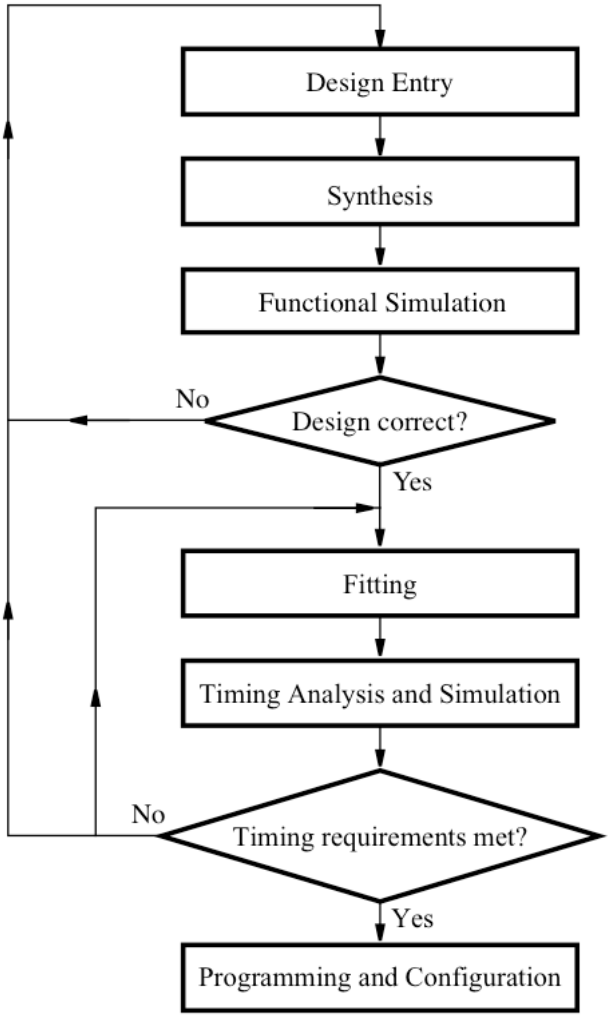



Figure 1 - Typical CAD Design Flow

The CAD flow involves the following steps:

**Design Entry** – the desired circuit is specified either by means of a schematic diagram, or by using a hardware description language, such as Verilog or VHDL. We will be using Verilog in this course.

**Simulation** – the Verilog/schematic version of the circuit is simulated to verify its functional correctness; this simulation does not take into account any timing or synthesis issues

**Logic Synthesis and Mapping** – the entered design is synthesized into a circuit comprised of combinational logic and registers. The final logic circuit must be mapped to the logic and register resources available on the FPGA. This design is optimized as much as possible to reduce the circuit size and increase performance. The resulting circuit is often called a "netlist".

**Functional Simulation** – the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues. This step is usually omitted under the assumption that the logic synthesis step is correct.

**Place and Route** – the CAD Fitter tool determines the placement of the logic elements (LEs) defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs

**Timing Analysis** – propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit. If the circuit does not meet performance goals, then the circuit must modified to reduce delay, or the tools re-run with a higher level of optimization (and correspondingly longer run time).

**Timing Simulation** – the fitted circuit, which includes logic and wire delays, is simulated to verify both its functional correctness and timing.

**Programming and Configuration** – the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections. This is done by downloading the "bit-file" generated by the CAD tools to the FPGAs.

The design process for ASICs is slightly different. The logic synthesis tools target the library of logic cells available for the ASIC technology, and placement and routing has more flexibility since the chip is being made from the ground up, not using an already pre-fabricated FPGA. But the biggest difference is that the resulting design "bit-file" is sent to a foundry to be made into a chip, which takes several months and millions of dollars, not the 1 second download into a $20-$200 FPGA.

In this tutorial, we will not cover simulation – this will be covered in a later lab assignment. We will also give you a complete circuit design for a camera pipeline to start with. Later you will be modifying this design by editing and/or adding Verilog files.

## Getting Started

Each logic circuit designed with Quartus II software is called a *project*. Like a software project, a Quartus project typically has several design files. In addition, the project also keeps track of tools setting, pin assignments, etc. The software works on one project at a time and keeps all information for that project in a single directory (folder) in the file system. In this tutorial, you will use a project that we have created. The process of creating a new project is described in the full Quartus II

Tutorial, but you won't need that for this class. (Creating a project involves making a new directory, importing the design files you need, importing pin assignment and setting the synthesis options.)

To begin, copy the cameraPipeline folder from the course directory to your directory. Start the Quartus II software. You should see a display similar to the one in Figure 2. A list of recent projects appear in the start window (after you have been using Quartus). To open a new project, you can either click the Open Existing Project button, or go to the File menu and use "Open Project". After selecting the "Open Existing Project" button, navigate to your copy of the camaraPipeline folder and open the .qpf file.

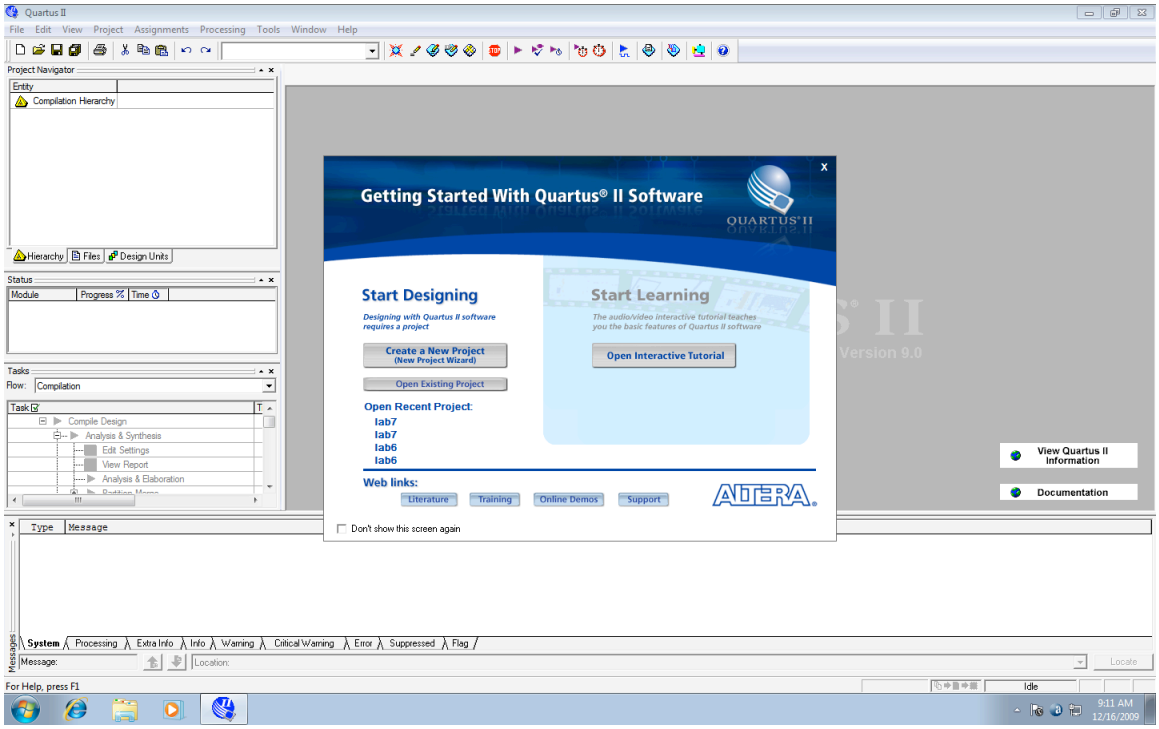



**Figure 2 - Project Start Window**

Quartus now reads in all the files associated with your project. Click on the Files tab in the Project Navigator to see a list of all the Verilog files in this project. (See Figure 3). As you will see, this is quite a large project, but don't let that worry you. We will be examining and modifying a few of these files as we are studying hardware design and understanding how Verilog works. We will be ignoring most of them; however, feel free to take a look at them if you are really interested in how the entire pipeline is constructed.

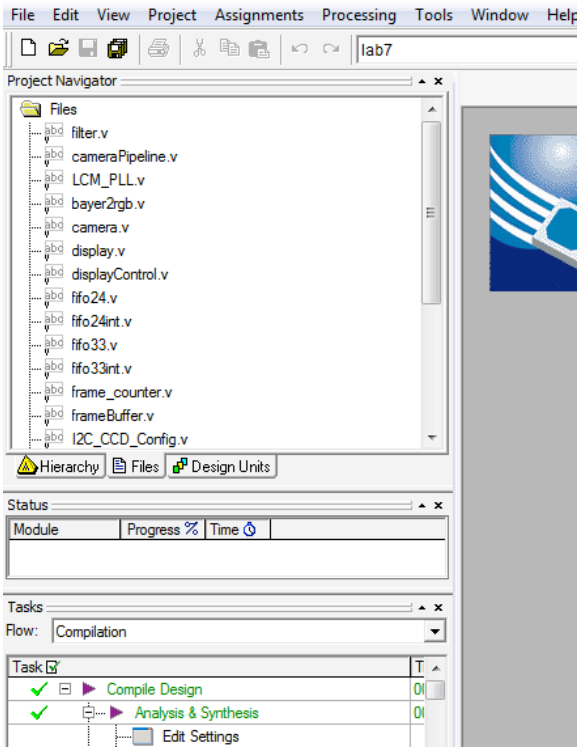



**Figure 3 - The Project Navigator Files tab lists all the design files**

We will not be using most of the tools and options available in Quartus. We will only be editing Verilog files, compiling them and downloading the results into the FPGA. Feel free to look around to see the commands and tools that are available.

Note that there is a simulator as part of Quartus, but we will not be using it. Instead, in an upcoming lab, we will be using ModelSim, an industry-standard simulator from Mentor.

Double-click on the filter.v file in the Project Navigator. This will open the Verilog file (see Figure 4). You don't have to understand how Verilog works yet, but it is pretty easy to see what this module does. It has red, green and blue inputs, which are passed along to the red, green and blue outputs, unless the enable input is set to 1. In this case, the outputs are all set to 0. Each of these values is an 8-bit unsigned number. It turns out that this enable signal is connected to switch 4 on the board in the cameraPipeline.v file, which is the top-level module in the circuit hierarchy. (You can see this if you search for the filter module in cameraPipeline.v – around line 300.)

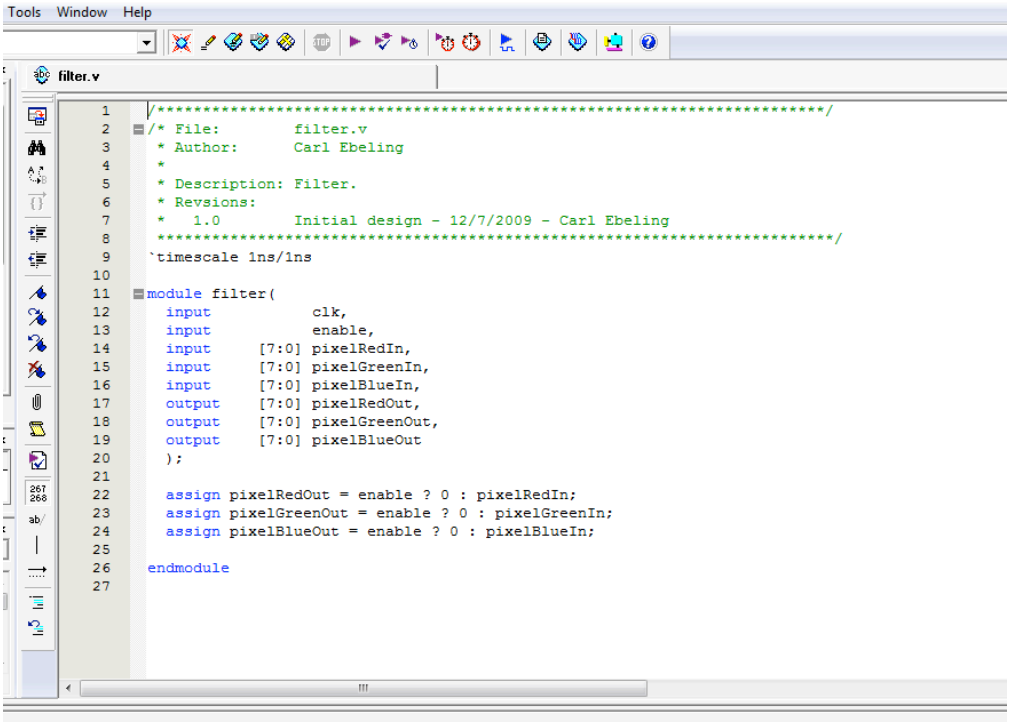



Figure 4 - Window with the filter.v file open

## Compiling the Project

Once you have finished entering a design (and have used simulation to be confident that it works), you compile it, which uses logic synthesis, mapping, placement and routing to generate a bit-file (like an executable) that we can download into the FPGA. These tools are listed in the Tasks window (Figure 5). You can execute the entire tool chain by double-clicking the Compile Design command. The Tasks window will show you what tools are being run and the progress.

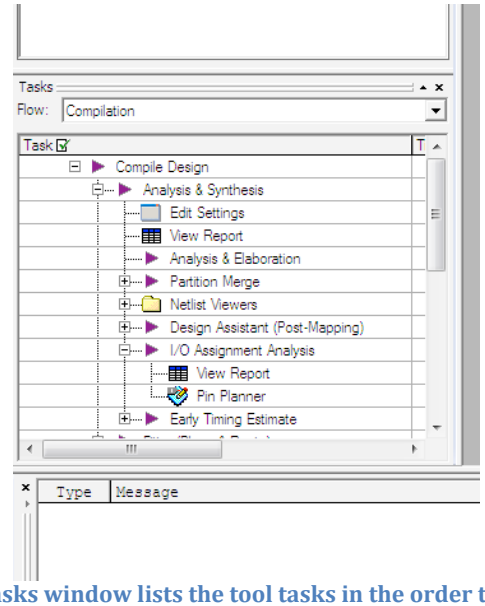
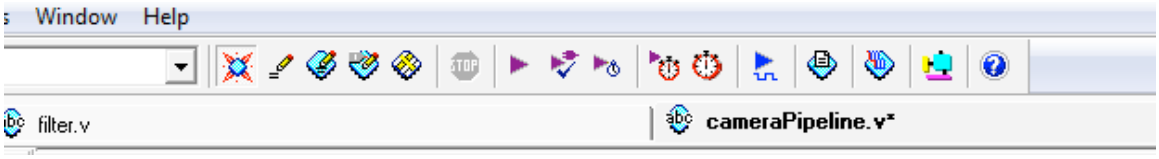



Figure 5 - The Tasks window lists the tool tasks in the order they are executed

This also appears at the top as the “Start Compilation” button (the purple “play” button – these buttons are “mouse-over” buttons which displays their names.)




The Messages window will display compilation messages along with Warnings and Errors. You may ignore Warnings (for now), but Errors will cause the compilation to abort. But there won’t be any Errors unless you happened to change something in the project. Compiling the circuit will take 5 to 10 minutes – be patient and take comfort in the fact that really big FPGAs can take many hours for this compilation process!

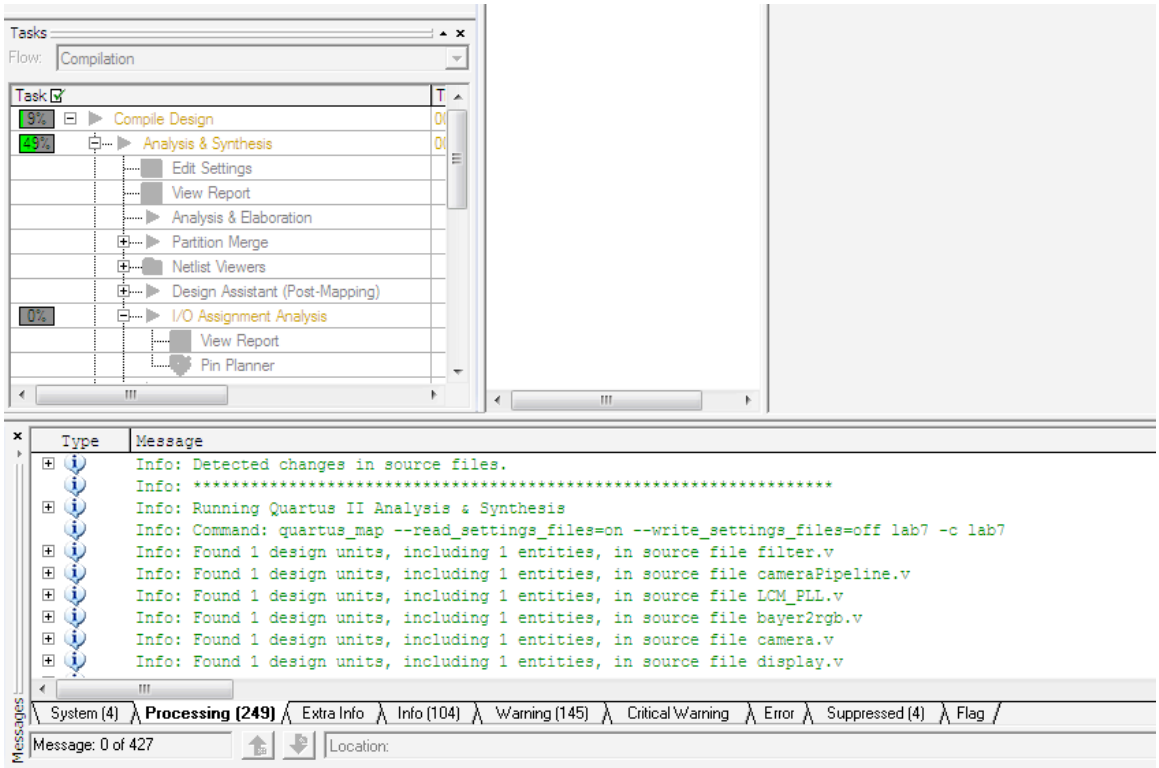



**Figure 6 - The messages window displays the compiler messages. Note the tabs.**

Before we can run the compiled circuit, we need something to run it on.

## The Demo Board

The Altera DE-1 demo board has an Altera Cyclone FPGA with about 20,000 LUTs (we’ll talk about these later). Also on the board are memories, clocks, lights, switches, buttons and interfaces to all sorts of devices. Our project is a camera pipeline, which connects to a camera on the input end and to an LCD display on the

output end. Since the camera and LCD display have different pixel formats, frame sizes and frame rates, we need to insert hardware that performs the appropriate conversions. We can also process the images as they pass through the pipeline. For example, in a later lab, we will design an adaptive thresholding filter that responds to the ambient light.

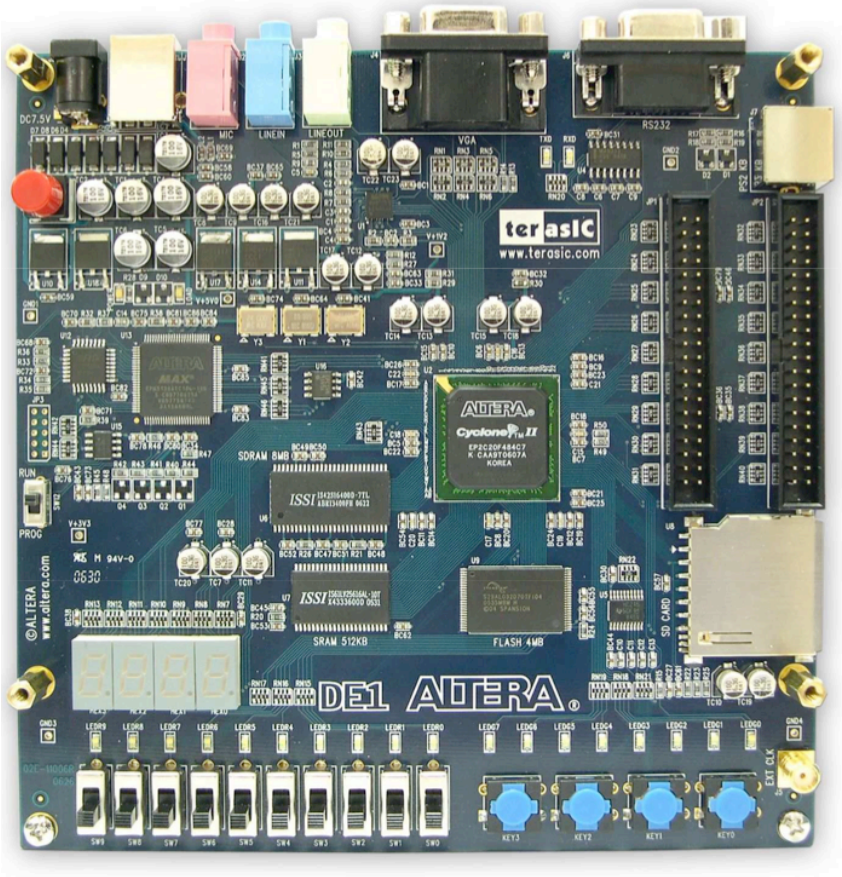



Figure 7 - Altera DE-1 Demo Board

To set up the board, plug in the power adaptor (Figure 7, top left) and connect it to your computer via the USB cable (top left). (The board will actually run off the USB power if the design doesn't draw too much power). There are 4 pushbuttons (blue lower right) and 10 switches (lower left). Pushbutton 0 is usually reserved for reset.

On the right side of the board, you will see two connectors labeled GPIO0 and GPIO1. The inside connector is used to connect the LCD display to the board, and the outside connector is used for the camera. (See Figure 8 and Figure 9.)

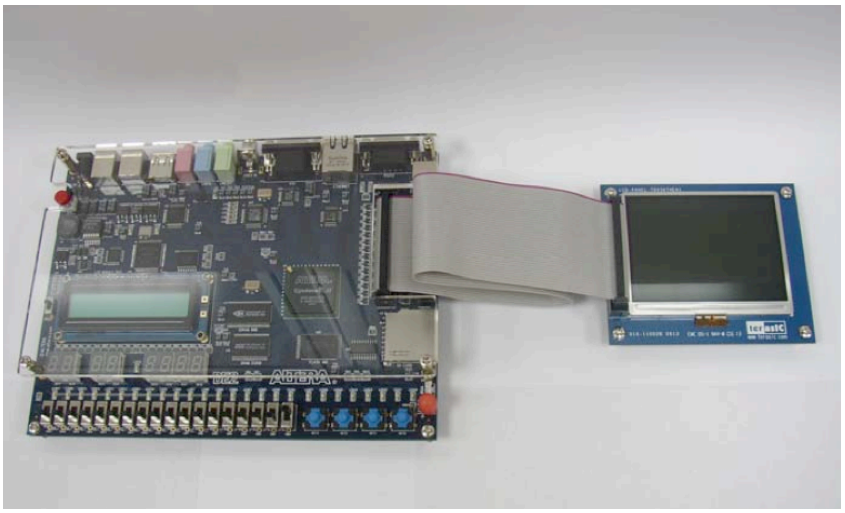


Figure 8 - LCD Display plugged into the inside connector

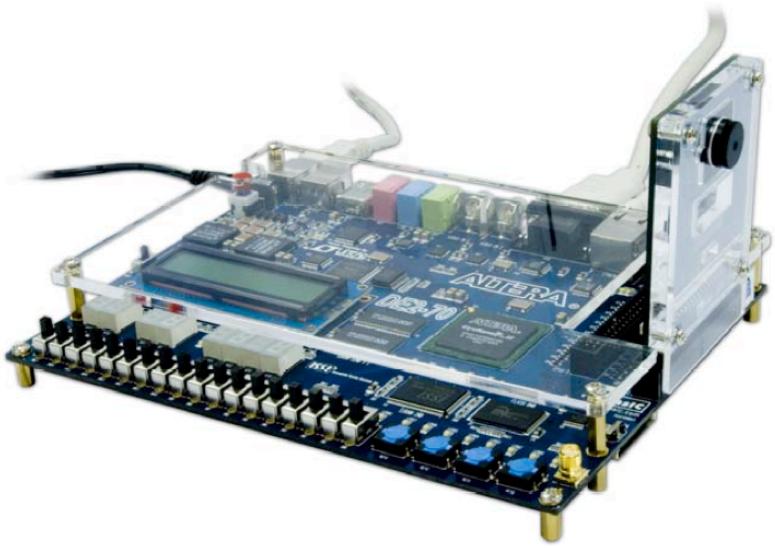


Figure 9 - Camera plugged into the outside connector

You are now ready to turn your board on. This executes a pre-loaded circuit that will flash lights and cycle numbers through the 7-segment displays.

## Downloading the Circuit

To download the circuit into the FPGA, click the Program Device command at the end of the Task window. (Figure 10) This also appears at the top as the "Programmer" button (3rd from right).

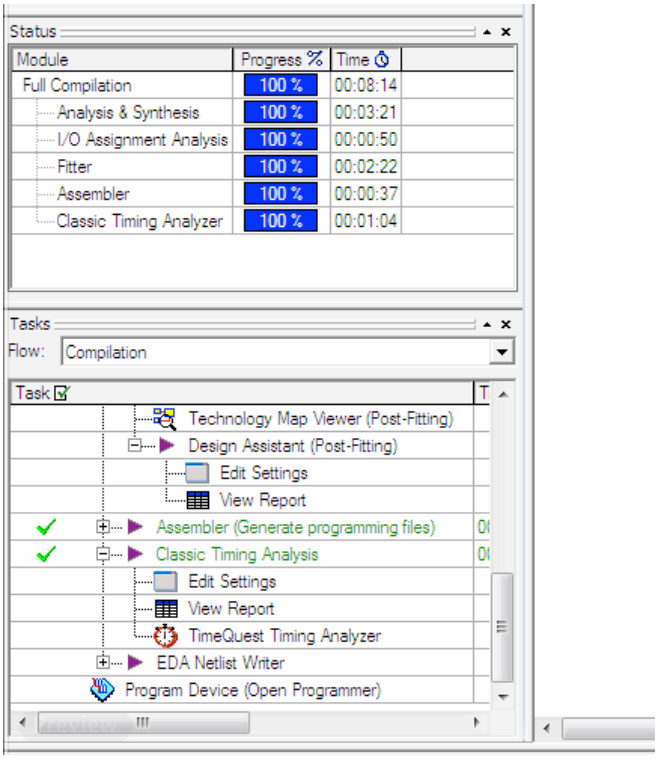
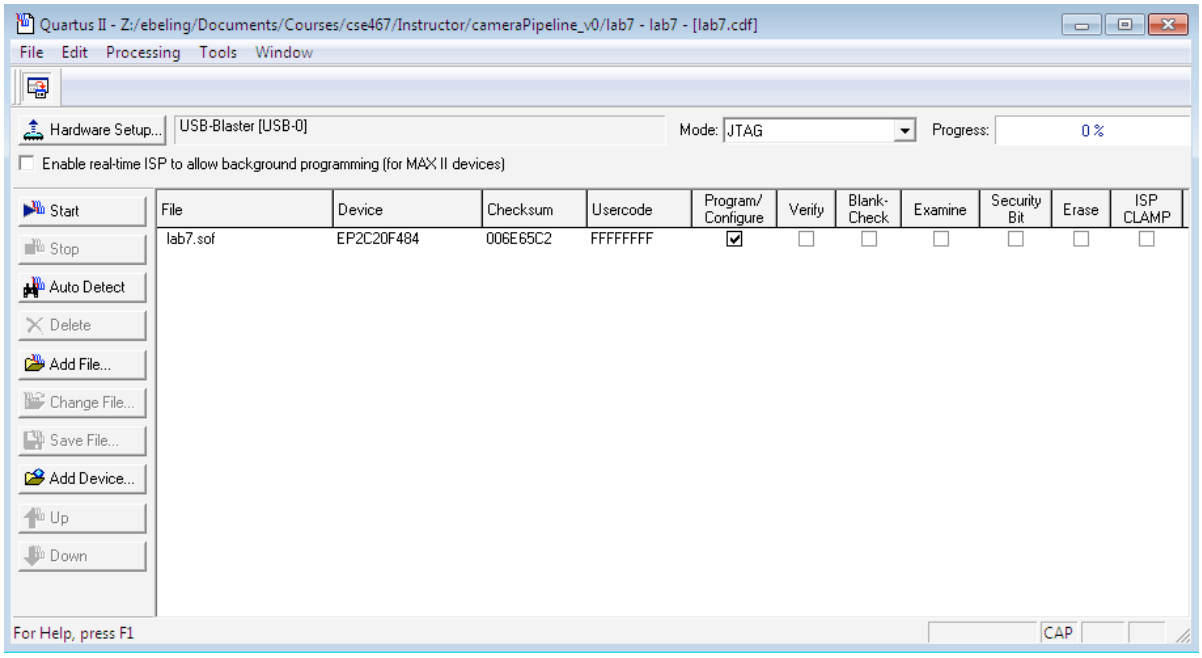



**Figure 10 - The Program Task command is the last task in the Task window**

This will bring up the following window:




(If the entry next to Hardware Setup says “No Hardware”, then either your computer is not connected to the board via the USB cable or you simply have to specify the hardware to use. To do this, click on the “Hardware Setup” button, and select “USB-Blaster” from the drop down menu.)

Now if you click the "Start" button, the bit-file created when you compiled the project is downloaded into the FPGA. The circuit will start running, but you should press Button 0 on the board, which is the reset button for the circuit. All the switches should be off (down).

Try toggling switch 7 – what happens?

Now toggle switch 6 – what happens? Remember what the filter.v module did?

## Changing and Re-Compiling Your Circuit

Now that you know how to load a project, edit files, compile and download the result to the board, you are ready to change the design. Open up the filter.v file, which describes one of the modules in the pipeline. The module as provided implements a simple on/off feature using switch 6.

Change this module to do something more interesting, for example:

- Display a gray-scale version of the image
- Display a false-color version of the image (permute the colors)
- Display a thresholded black/white version of the image
- Your own idea

I know that you don't know Verilog, but you can see that the syntax is similar to C and if you just stick to changing the assign statements, you don't need to know anything else. But if you want more, just ask us.

Once you have made the change, re-compile your circuit (fixing any Errors that arise) and download to the board. Demo your modified design to one of the instructors.