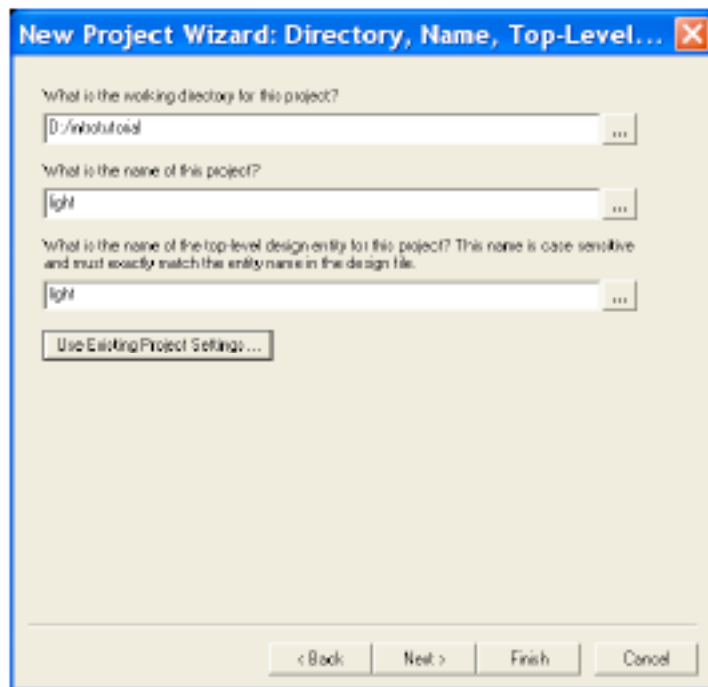


Quartus II Abbreviated Manual *

Creating a new project

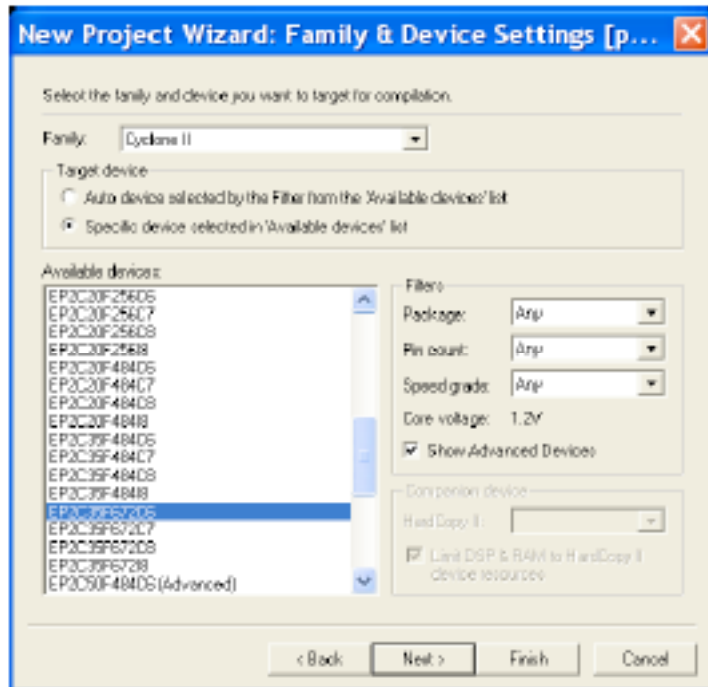
To start working on a new design we first have to define a new design project. Quartus II software makes the designer's task easy by providing support in the form of a wizard. Create a new project as follows:

1. Select *File* → *New Project Wizard*. Press *Next* to get the window shown bellow.



2. Set the working directory to any directory of your choice. The project must have a name, which is usually the same as the top-level design entity that will be included in the project. Press *Next*. If the directory you selected doesn't yet exist, Quartus II software will display a pop-up box asking if it should create the desired directory. Click *Yes*.
3. The wizard makes it easy to specify which existing files (if any) should be included in the project. Assuming that we do not have any existing files, click *Next*, which leads to the window shown bellow.

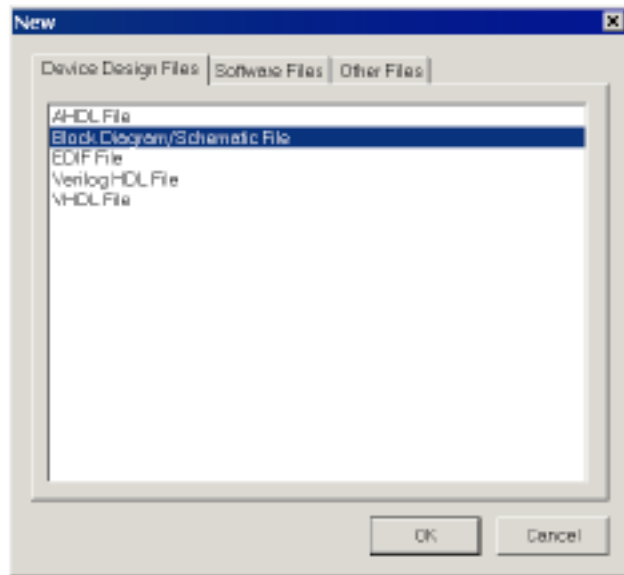
* Summarized from Altera's tutorials and manuals by Tali Moreshet.



4. We have to specify the type of device in which the designed circuit will be implemented. Choose *Cyclone II* as the target device family. From the list of available devices, choose the device called *EP2C35F672C6*, which is the FPGA used on Altera's DE2 board. Press *Next*.
5. The user can specify any third-party tools that should be used. Since we will rely solely on Quartus II tools, we will not choose any other tools. Press *Next*.
6. A summary of the chosen settings appears in the screen. Press *Finish*, which returns to the main Quartus II window.

Design using Schematic Capture

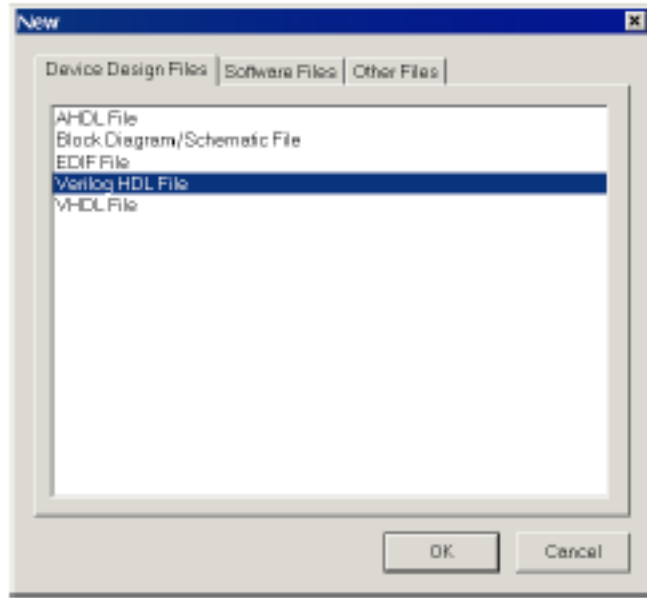
1. Select *File* → *New* to get the window bellow, choose *Block Diagram/Schematic File* and click OK. This opens the Graphic Editor window.



2. The first step is to specify a name for the file that will be created. Select *File* → *Save As* to open a pop-up box. In the box labeled *Save as type* choose *Block Diagram/Schematic File (*.bdf)*. In the box labeled *File name* type the top-level design entity you specified when the project was created (or another file name for any other entity). Put a checkmark in the box *Add file to current project*. Click *Save*, which puts the file into the project directory.
3. The Graphic Editor provides a number of libraries, which include circuit elements that can be imported into a schematic. Double-click on the blank space in the Graphic Editor window, or click on the icon in the toolbar that looks like an AND gate. A pop-up box will appear. Expand the hierarchy in the Libraries box. *Primitives* → *logic* contains logic gates, and *primitives* → *pin* contains input and output pins. Select the gates and pins required, one at a time, and click OK. The symbols can be moved, rotated, and also duplicated.
4. To assign names to the input and output symbols, double click *pin_name* on the top input symbol. Type the pin name and click OK.
5. To connect the gates and pins by wires, click on the icon in the toolbar to activate the *Orthogonal Node Tool*. Position the mouse pointer over the right edge of a gate or pin, click and hold the mouse button and drag the mouse to the right until the drawn line reaches the target gate or pin to be connected. Release the mouse button.

Design using Verilog

1. Select *File* → *New* to get the window bellow, choose *Verilog HDL File*, and click OK. This opens the Text Editor window.




2. The first step is to specify a name for the file that will be created. Select *File* → *Save As* to open a pop-up box. In the box labeled *Save as type* choose *Verilog HDL File (*.v)*. In the box labeled *File name* type the top-level design entity you specified when the project was created (or another file name for any other entity). Put a checkmark in the box *Add file to current project*. Click *Save*, which puts the file into the project directory.
3. The syntax of Verilog code is sometimes difficult for a designer to remember. To help with this issue, the Text Editor provides a collection of Verilog templates. The templates provide examples of various types of Verilog statements, such as a module declaration, an always block, and assignment statements. It is worthwhile to browse through the templates by selecting *Edit* → *Insert Template* → *Verilog HDL* to become familiar with this resource.

Compiling and Simulating

1. To compile and synthesize your design, run the Compiler by selecting *Processing* → *Start Compilation*, or by clicking on the toolbar icon that looks like a purple triangle. As the compilation moves through various stages, its progress is reported in a window on the left side of the Quartus II display. Successful (or unsuccessful) compilation is indicated in a pop-up box. Acknowledge it by clicking OK.
2. To simulate your design, open the Waveform Editor window by selecting *File* → *New*. Click on the *Other Files* tab, choose *Vector Waveform File* and click OK. Save the file as <project name>.vwf.
3. Set the desired simulation to run from 0 to 200 us (or any other duration of your

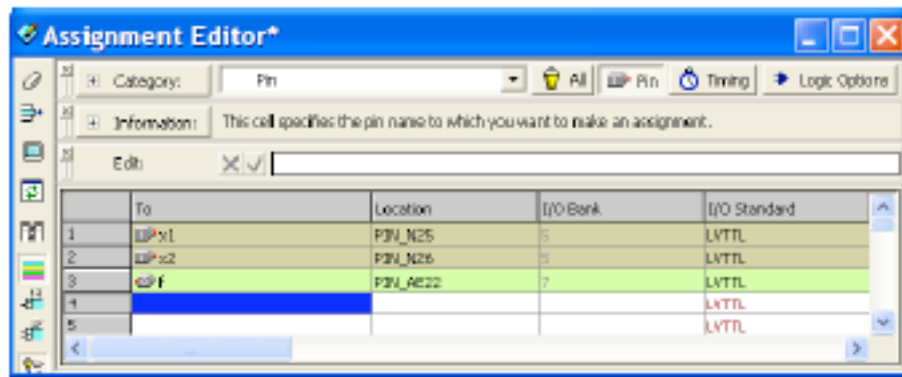
- choice) by selecting *Edit* → *End Time* and entering 200 us in the dialog box that pops up. Selecting *View* → *Fit in Window* displays the entire simulation range of 0 to 200 us in the window.
4. Make sure your end time, grid size, and clock (if using one) are all at least 1usec.
 5. Next, we want to include the input and output nodes of the circuit to be simulated. Click *Edit* → *Insert Node or Bus* and click on the button labeled *Node Finder*. The Node Finder utility has a filter used to indicate which type of nodes are to be found. Since we are interested in input and output pins, set the filter to *Pins: all*. Click the *List* button, mark the input and output nodes, and then click the > sign to add the nodes to the Selected Nodes box. Click OK to close the Node Finder window, then OK again.
 6. We will now specify the logic values to be used for the input signals during simulation. The logic values at the outputs will be generated automatically by the simulator. To generate the desired input waveforms click on the waveform name for the first input. Once a waveform is selected, the editing commands in the Waveform Editor can be used to draw the desired waveforms. Commands are available for setting a selected signal to 0, 1, unknown (X), high impedance (Z), don't care (DC), inverting its existing value (INV), or defining a clock or a counter waveform. Each command can be activated by using the *Edit* → *Value* command, or via the toolbar for the Waveform Editor. The Edit menu can also be opened by right-clicking on a waveform name. To set a signal to 1 for the time interval 0 to 100ns, press the mouse at the start of the interval and drag it to its end, which highlights the selected interval, and choose the logic value 1 in the toolbar.

Tip: Grouping signals together to form a bus allows to assign a value (such as counter) to all signals together.
 7. To start a simulation, select *Processing* → *Start Simulation*, or click the icon . At the end of the simulation, Quartus II software indicates its successful completion and displays a Simulation Report. If your report window does not show the entire simulation time range, click on the report window to select it and choose *View* → *Fit in Window*.

FPGA PIN Assignment

1. To assign FPGA pins to the inputs and outputs of your design, you will be using the Assignment Editor. Select *Assignments* → *assignment editor*. Under *Category* select *Pin*. Double-click on the entry <<new>>.
2. Choose a pin from the drop-down menu that includes all the pins in your design. Follow this by double-clicking on the box to the right of this new entry, in the column labeled *Location*. Choose a pin from the drop-down menu that includes all

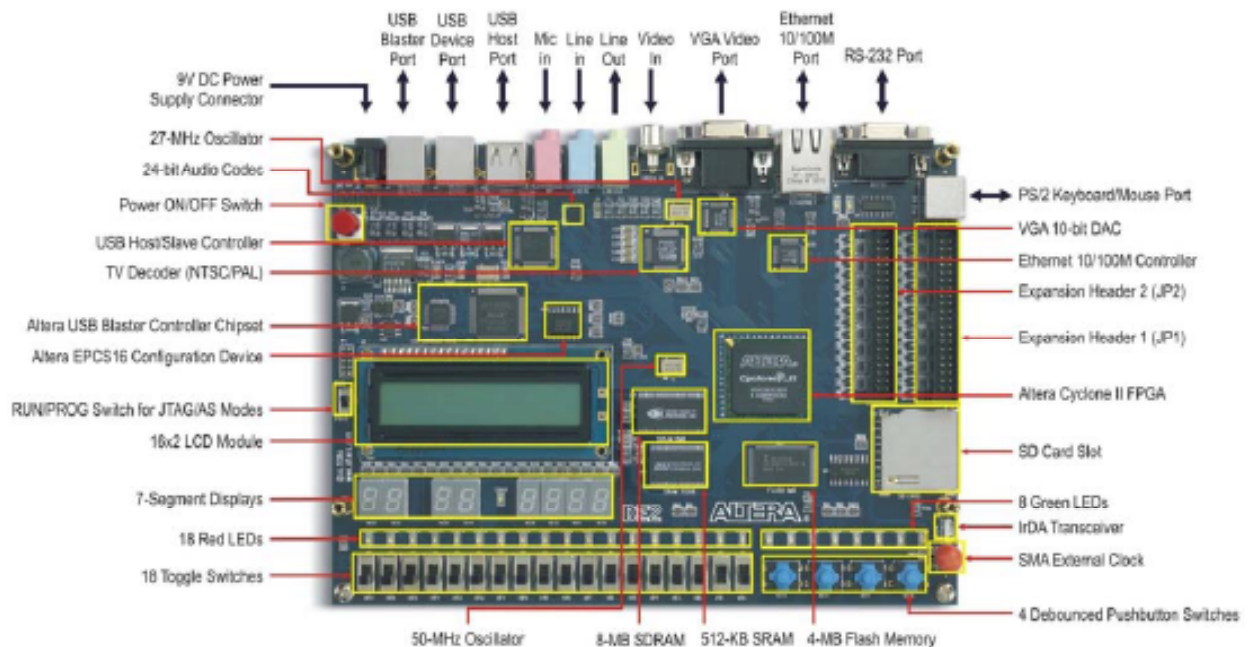
the FPGA pins.



3. To save the assignments made, choose *File* → *Save*. Compile your design with the new pin assignments.

Altera DE2 FPGA Board

This is a short overview of the FPGA board:



We will use the following FPGA pins:

- Toggle switches – level sensitive
- Pushbutton switches – active low
- LEDs – activated on high
- 7-segment display – activated on low
- Clock inputs

Pin assignments for the toggle switches (level sensitive):

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_N25	Toggle Switch[0]
SW[1]	PIN_N26	Toggle Switch[1]
SW[2]	PIN_P25	Toggle Switch[2]
SW[3]	PIN_AE14	Toggle Switch[3]
SW[4]	PIN_AF14	Toggle Switch[4]
SW[5]	PIN_AD13	Toggle Switch[5]
SW[6]	PIN_AC13	Toggle Switch[6]
SW[7]	PIN_C13	Toggle Switch[7]
SW[8]	PIN_B13	Toggle Switch[8]
SW[9]	PIN_A13	Toggle Switch[9]
SW[10]	PIN_N1	Toggle Switch[10]
SW[11]	PIN_P1	Toggle Switch[11]
SW[12]	PIN_P2	Toggle Switch[12]
SW[13]	PIN_T7	Toggle Switch[13]
SW[14]	PIN_U3	Toggle Switch[14]
SW[15]	PIN_U4	Toggle Switch[15]
SW[16]	PIN_V1	Toggle Switch[16]
SW[17]	PIN_V2	Toggle Switch[17]

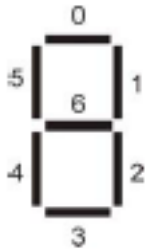
Pin assignments for the pushbutton switches (active low):

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_G26	Pushbutton[0]
KEY[1]	PIN_N23	Pushbutton[1]
KEY[2]	PIN_P23	Pushbutton[2]
KEY[3]	PIN_W26	Pushbutton[3]

Pin assignments for the LEDs (activated on high):

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_AE23	LED Red[0]
LEDR[1]	PIN_AF23	LED Red[1]
LEDR[2]	PIN_AB21	LED Red[2]
LEDR[3]	PIN_AC22	LED Red[3]
LEDR[4]	PIN_AD22	LED Red[4]
LEDR[5]	PIN_AD23	LED Red[5]
LEDR[6]	PIN_AD21	LED Red[6]
LEDR[7]	PIN_AC21	LED Red[7]
LEDR[8]	PIN_AA14	LED Red[8]
LEDR[9]	PIN_Y13	LED Red[9]
LEDR[10]	PIN_AA13	LED Red[10]
LEDR[11]	PIN_AC14	LED Red[11]
LEDR[12]	PIN_AD15	LED Red[12]
LEDR[13]	PIN_AE15	LED Red[13]
LEDR[14]	PIN_AF13	LED Red[14]
LEDR[15]	PIN_AE13	LED Red[15]
LEDR[16]	PIN_AE12	LED Red[16]
LEDR[17]	PIN_AD12	LED Red[17]
LEDG[0]	PIN_AE22	LED Green[0]
LEDG[1]	PIN_AF22	LED Green[1]
LEDG[2]	PIN_W10	LED Green[2]
LEDG[3]	PIN_V18	LED Green[3]
LEDG[4]	PIN_U18	LED Green[4]
LEDG[5]	PIN_U17	LED Green[5]
LEDG[6]	PIN_AA20	LED Green[6]
LEDG[7]	PIN_Y18	LED Green[7]
LEDG[8]	PIN_Y12	LED Green[8]

Pin assignments for the 7-segment display (activated on low):



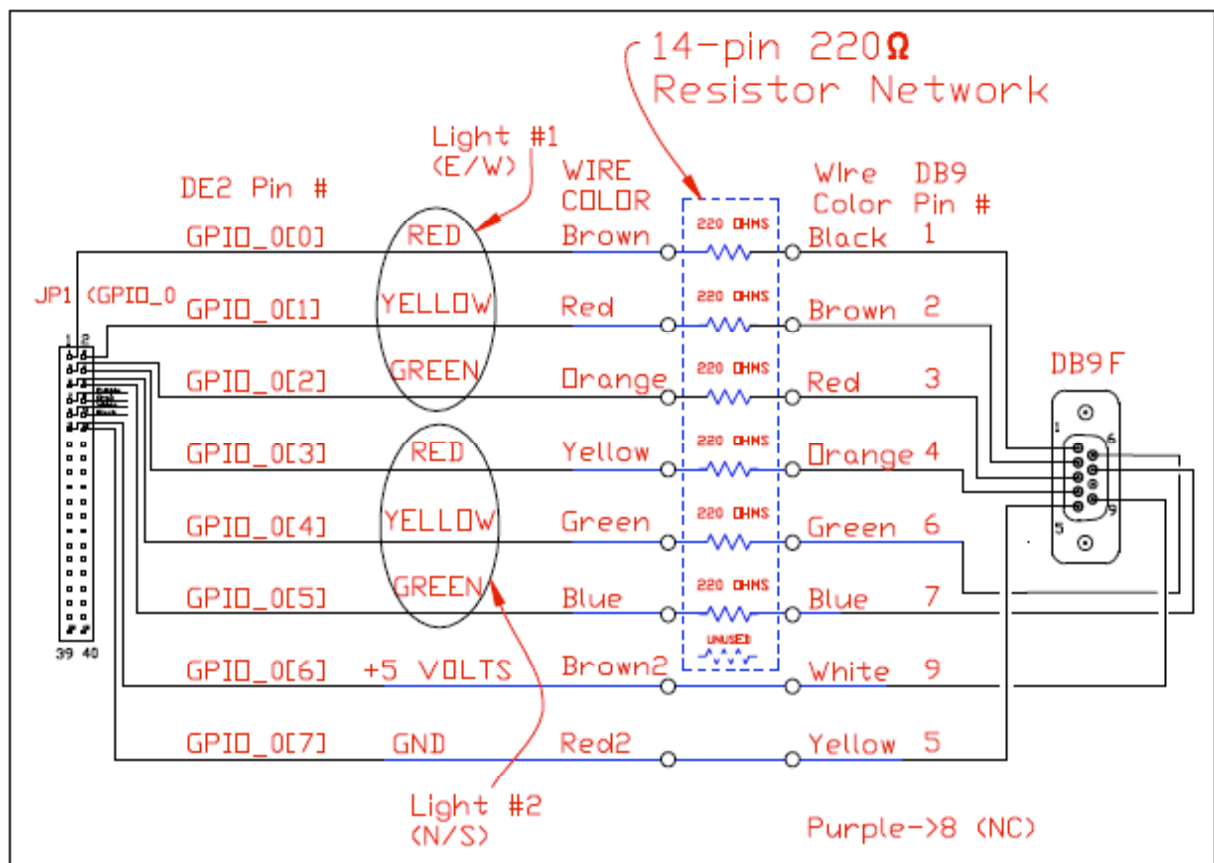
Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_AF10	Seven Segment Digit 0[0]
HEX0[1]	PIN_AB12	Seven Segment Digit 0[1]
HEX0[2]	PIN_AC12	Seven Segment Digit 0[2]
HEX0[3]	PIN_AD11	Seven Segment Digit 0[3]
HEX0[4]	PIN_AE11	Seven Segment Digit 0[4]
HEX0[5]	PIN_V14	Seven Segment Digit 0[5]
HEX0[6]	PIN_V13	Seven Segment Digit 0[6]
HEX1[0]	PIN_V20	Seven Segment Digit 1[0]
HEX1[1]	PIN_V21	Seven Segment Digit 1[1]
HEX1[2]	PIN_W21	Seven Segment Digit 1[2]
HEX1[3]	PIN_Y22	Seven Segment Digit 1[3]
HEX1[4]	PIN_AA24	Seven Segment Digit 1[4]
HEX1[5]	PIN_AA23	Seven Segment Digit 1[5]
HEX1[6]	PIN_AB24	Seven Segment Digit 1[6]
HEX2[0]	PIN_AB23	Seven Segment Digit 2[0]
HEX2[1]	PIN_V22	Seven Segment Digit 2[1]
HEX2[2]	PIN_AC25	Seven Segment Digit 2[2]
HEX2[3]	PIN_AC26	Seven Segment Digit 2[3]
HEX2[4]	PIN_AB26	Seven Segment Digit 2[4]
HEX2[5]	PIN_AB25	Seven Segment Digit 2[5]
HEX2[6]	PIN_Y24	Seven Segment Digit 2[6]
HEX3[0]	PIN_Y23	Seven Segment Digit 3[0]
HEX3[1]	PIN_AA25	Seven Segment Digit 3[1]
HEX3[2]	PIN_AA26	Seven Segment Digit 3[2]

HEX3[3]	PIN_Y26	Seven Segment Digit 3[3]
HEX3[4]	PIN_Y25	Seven Segment Digit 3[4]
HEX3[5]	PIN_U22	Seven Segment Digit 3[5]
HEX3[6]	PIN_W24	Seven Segment Digit 3[6]
HEX4[0]	PIN_U9	Seven Segment Digit 4[0]
HEX4[1]	PIN_U1	Seven Segment Digit 4[1]
HEX4[2]	PIN_U2	Seven Segment Digit 4[2]
HEX4[3]	PIN_T4	Seven Segment Digit 4[3]
HEX4[4]	PIN_R7	Seven Segment Digit 4[4]
HEX4[5]	PIN_R6	Seven Segment Digit 4[5]
HEX4[6]	PIN_T3	Seven Segment Digit 4[6]
HEX5[0]	PIN_T2	Seven Segment Digit 5[0]
HEX5[1]	PIN_P6	Seven Segment Digit 5[1]
HEX5[2]	PIN_P7	Seven Segment Digit 5[2]
HEX5[3]	PIN_T9	Seven Segment Digit 5[3]
HEX5[4]	PIN_R5	Seven Segment Digit 5[4]
HEX5[5]	PIN_R4	Seven Segment Digit 5[5]
HEX5[6]	PIN_R3	Seven Segment Digit 5[6]
HEX6[0]	PIN_R2	Seven Segment Digit 6[0]
HEX6[1]	PIN_P4	Seven Segment Digit 6[1]
HEX6[2]	PIN_P3	Seven Segment Digit 6[2]
HEX6[3]	PIN_M2	Seven Segment Digit 6[3]
HEX6[4]	PIN_M3	Seven Segment Digit 6[4]
HEX6[5]	PIN_M5	Seven Segment Digit 6[5]
HEX6[6]	PIN_M4	Seven Segment Digit 6[6]
HEX7[0]	PIN_L3	Seven Segment Digit 7[0]
HEX7[1]	PIN_L2	Seven Segment Digit 7[1]
HEX7[2]	PIN_L9	Seven Segment Digit 7[2]
HEX7[3]	PIN_L6	Seven Segment Digit 7[3]
HEX7[4]	PIN_L7	Seven Segment Digit 7[4]
HEX7[5]	PIN_P9	Seven Segment Digit 7[5]
HEX7[6]	PIN_N9	Seven Segment Digit 7[6]

Pin assignments for the clock inputs:

Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D13	27 MHz clock input
CLOCK_50	PIN_N2	50 MHz clock input
EXT_CLOCK	PIN_P26	External (SMA) clock input

Pin assignments for the traffic light cable (which is activated on low):



Pin assignments for the expansion headers, to which the traffic light cable is connected:

Signal Name	FPGA Pin No.	Description
GPIO_0[0]	PIN_D25	GPIO Connection 0[0]
GPIO_0[1]	PIN_J22	GPIO Connection 0[1]
GPIO_0[2]	PIN_E26	GPIO Connection 0[2]
GPIO_0[3]	PIN_E25	GPIO Connection 0[3]
GPIO_0[4]	PIN_F24	GPIO Connection 0[4]
GPIO_0[5]	PIN_F23	GPIO Connection 0[5]
GPIO_0[6]	PIN_J21	GPIO Connection 0[6]
GPIO_0[7]	PIN_J20	GPIO Connection 0[7]
GPIO_0[8]	PIN_F25	GPIO Connection 0[8]
GPIO_0[9]	PIN_F26	GPIO Connection 0[9]

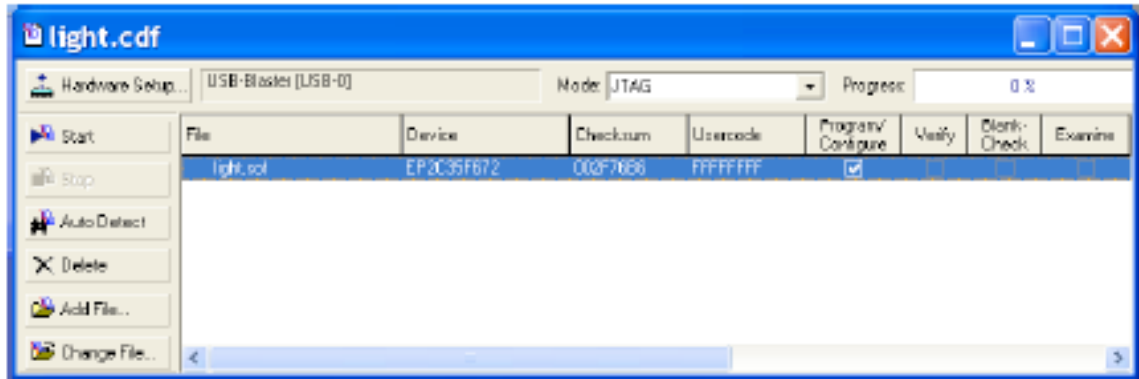
Programming and Configuring the FPGA Device

Altera's DE2 board allows the configuration to be done in two different ways, known as JTAG and AS modes. The configuration data is transferred from the host computer (which runs the Quartus II software) to the board by means of a cable that connects a USB port on the host computer to the leftmost USB connector on the board.

In the JTAG mode, the configuration data is loaded directly into the FPGA device. The acronym JTAG stands for Joint Test Action Group. This group defined a simple way for testing digital circuits and loading data into them, which became an IEEE standard. If the FPGA is configured in this manner, it will retain its configuration as long as the power remains turned on. The configuration information is lost when the power is turned off. The second possibility is to use the Active Serial (AS) mode. In this case, a configuration device that includes some flash memory is used to store the configuration data. Quartus II software places the configuration data into the configuration device on the DE2 board. Then, this data is loaded into the FPGA upon power-up or reconfiguration.

JTAG Programming:

1. Flip the RUN/PROG switch into the RUN position.
2. Select *Tools* → *Programmer* (or the programmer icon) to reach the window in the Figure bellow.
3. Specify the programming hardware and the mode that should be used. If not already chosen by default, select *JTAG* in the *Mode* box. Also, if the USB-Blaster is not chosen by default, press the *Hardware Setup...* button and select the USB-Blaster in the window that pops up.



4. Observe that the configuration file *<project name>.sof* is listed in the window. If the file is not already listed then click *Add File...* and select it. This is a binary file produced by the Compiler's Assembler module, which contains the data needed to configure the FPGA device. The extension *.sof* stands for SRAM Object File. Note also that the device selected is EP2C35F672, which is the FPGA device used on the DE2 board. Click on the *Program/Configure* check box, as shown in the Figure above.
5. Press *Start* in the window above. The *Progress* box will indicate when the configuration and programming process is completed.

Active Serial Mode Programming:

(Useful to keep the program when FPGA is disconnected from the power supply)

1. In this case, the configuration data has to be loaded into the configuration device on the DE2 board, which is identified by the name *EPCS16*. To specify the required configuration device select *Assignments → Device*, click on the *Device & Pin Options* button, and click on the *Configuration* tab. In the *Configuration device* box (which may be set to Auto) choose *EPCS16* and click OK. Click OK.
2. Recompile your design.
3. Flip the RUN/PROG switch into the PROG position.
4. Select *Tools → Programmer*. In the Mode box select *Active Serial Programming*. If you are changing the mode from the previously used JTAG mode, a pop-up box will appear, asking if you want to clear all devices. Click *Yes*. Make sure that the Hardware Setup indicates the USB-Blaster. If the configuration file is not already listed in the window, press *Add File*. Select the file *<project name>.pof* in your project directory and click *Open*. This is a binary file produced by the Compiler's Assembler module, which contains the data to be loaded into the EPCS16 configuration device. The extension *.pof* stands for Programmer Object File. Upon returning to the Programmer window, click on the *Program/Configure* check box.
5. Press *Start*. The *Progress* box will indicate when the configuration and programming process is completed.
6. You may now disconnect the board, move it, then reconnect it to the power

supply.

7. Flip the RUN/PROG switch back into the RUN position and reset the board by pressing the red button.