

Digital System Design Lab Manual- 3700

COMPREHENSIVE TUTORIAL FOR SETTING UP DE10-LITE HW/SW
PLATFORM

Instructor – Chris Myers
Document Author - Vikas Rao

DE10-Lite reference manual - <https://www.terasic.com.tw/DE10-lite-user-manual>

Objective -

This tutorial provides comprehensive information that will help you understand how to create a FPGA design and run it on your DE10-Lite development board. The following sections provide a quick overview of the design flow, explain what you need to get started, and describe what you will learn.

The standard FPGA design flow starts with design entry using schematics or a hardware description language (HDL), such as Verilog HDL or VHDL. In this step, you can create a digital circuit that is implemented inside the FPGA. The flow then proceeds through compilation, simulation, programming, and verification in the FPGA hardware.

This tutorial will not make you an expert, but at the end, you will understand basic concepts about Quartus Lite projects, entering design using HDL, compiling your design, and downloading it into the FPGA on your DE10-Lite development board.

Terasic DE10-Lite is a cost-effective Altera MAX 10 based FPGA board. The board utilizes the maximum capacity MAX 10 FPGA, which has around 50K logic elements(LEs) and on-die analog-to-digital converter (ADC). It features on-board USB-Blaster, SDRAM, accelerometer, VGA output, 2x20 GPIO expansion connector, and an Arduino UNO R3 expansion connector in a compact size. The kit provides the perfect system-level prototyping solution for industrial, automotive, consumer, and many other market applications.

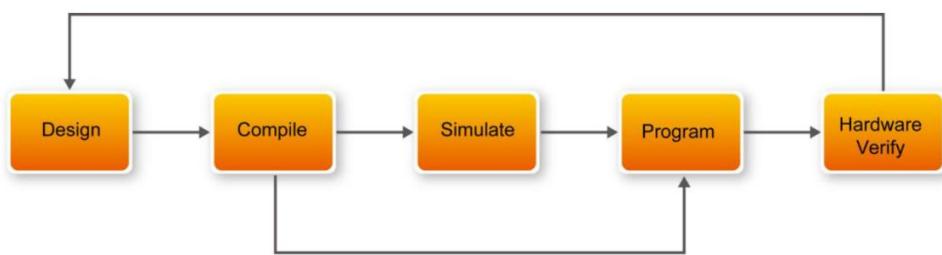


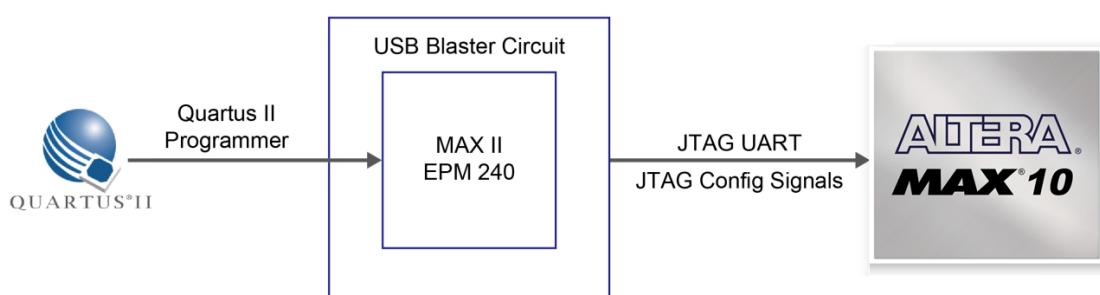
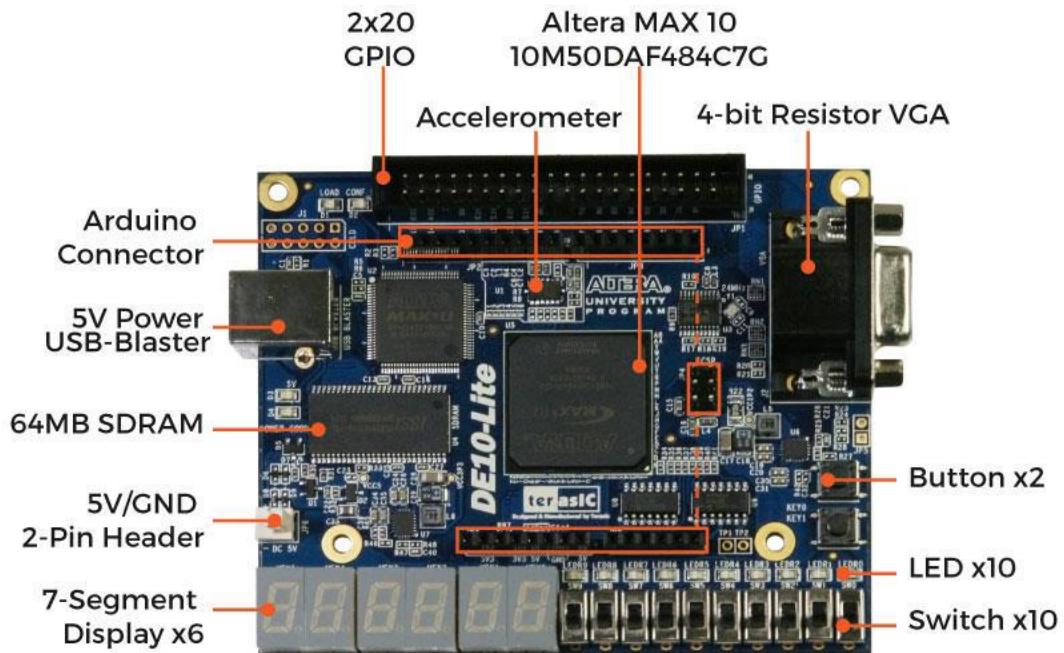
Figure 1-1 Design Flow

Index -

- Section 1 – Download and install Altera Quartus Lite Software on Windows/Linux
- Section 2 – Setup a new project in Quartus Lite Software
- Section 3 – Implement a function using Verilog HDL
- Section 4 - Simulate the Verilog circuit using ModelSim + Verilog test bench
- Section 5 – Pin assignment and Constraint generation
- Section 6 - Synthesize, Implement, Generate, and Program DE10-Lite board

Board Overview –

- Go over the chapters 1 and 2 in DE10-user manual to understand the board components and the control panel setup.



Section 1 - Download and install Altera Quartus Lite Software

1. Browse to the hyperlink provided below and make sure that Lite edition and 18.1 release are selected. Also, check the appropriate OS box(Windows/Linux). Use any download method (Direct Download works faster) and make sure to download only the individual files marked in the next bullet. Create a basic account on Intel using UnID if prompted -> http://fpgasoftware.intel.com/18.1/quartus_lite

Download Center for FPGAs

The screenshot shows the 'Design Software' section of the Intel Quartus Prime download page. It includes a sidebar with categories like Design Software, Embedded Software, Archives, Licensing (which is highlighted in orange), Programming Software, Drivers, Board System Design, Board Layout and Test, and Legacy Software. The main content area displays 'Quartus Prime Lite Edition' with a release date of September, 2018, and the latest release v18.1. It features dropdown menus for 'Select edition' (set to Lite) and 'Select release' (set to 18.1). Below this are sections for 'Operating System' (Windows and Linux) and 'Download Method' (Akamai DLM3 Download Manager and Direct Download).

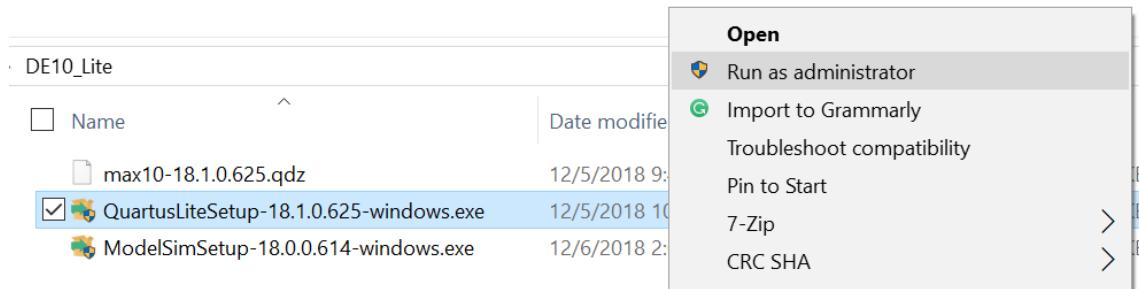
2. Navigate to the Individual files tab and download the Quartus prime lite edition, ModelSim FPGA edition, along with MAX 10 FPGA device support.

The screenshot shows the 'Individual Files' tab of the download center. It lists several software components:

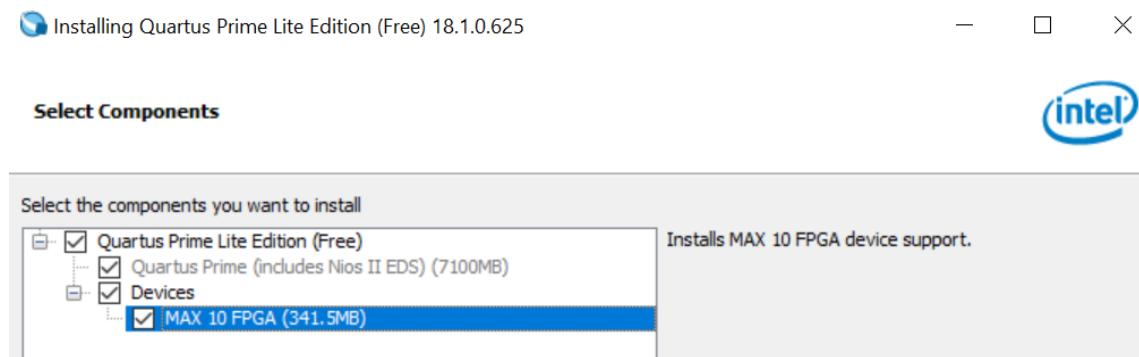
- Combined Files:** Quartus Prime Lite Edition (Free), Quartus Prime (includes Nios II EDS), ModelSim-Intel FPGA Edition (includes Starter Edition).
- Devices:** Arria II device support, Cyclone IV device support, Cyclone 10 LP device support, Cyclone V device support, MAX II, MAX V device support, MAX 10 FPGA device support.

Each item has a checkbox and a link to its details. At the bottom is a 'Download Selected Files' button.

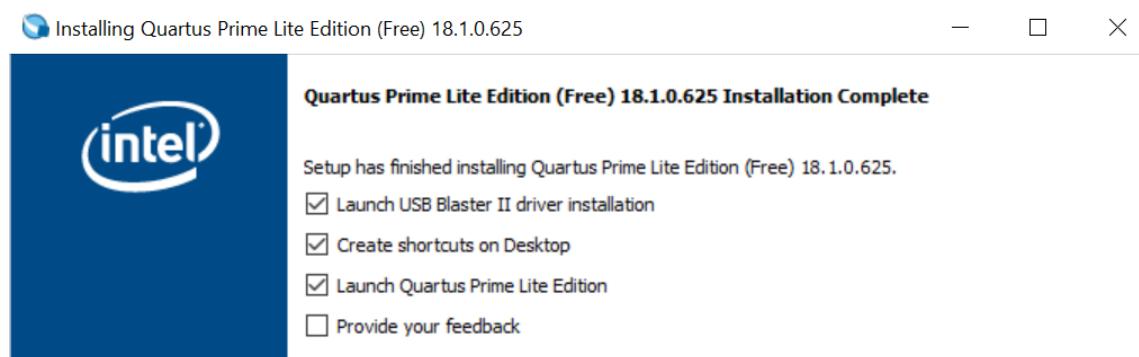
- Select the QuartusLite setup file and Run as administrator for windows installation. For Linux, modify the *.run file with executable permissions(chmod +x *.run) and execute the *.run on terminal as super user(sudo \.*.run).



- Press Yes and click Next in the QuartusLite setup installation prompt. Accept the agreement and click Next again. Select the appropriate installation folder or retain the default directory. Make sure the following boxes are marked(modelsim and MAX 10 device support) for installation. And Press Next and let the installation complete



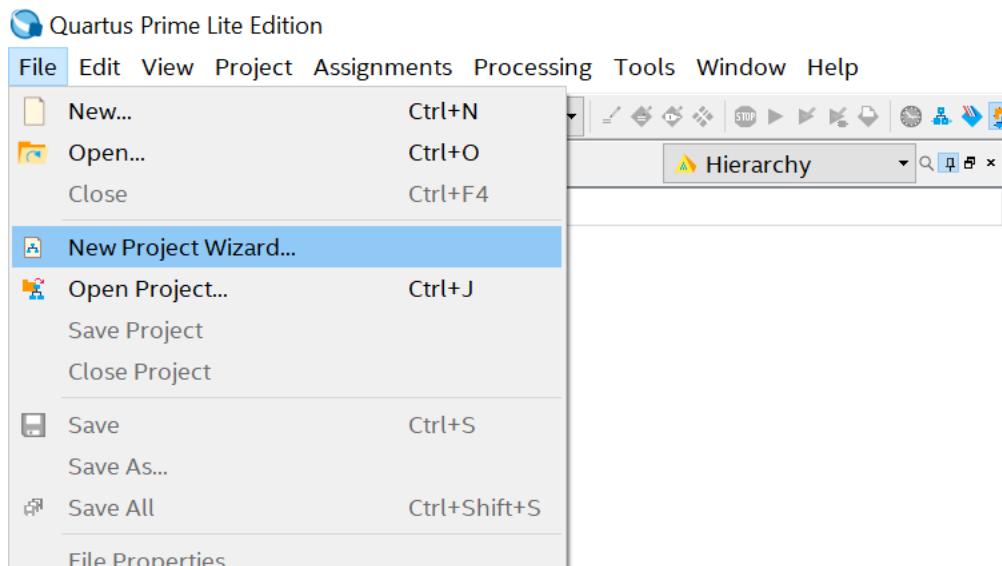
- The setup should also install ModelSim FPGA edition along with QuartusLite Software(If not, you can install it separately). After the installation, the setup will prompt for installation of USB Blaster 2 device driver which is required for FPGA programming. Click Next and finish the installation.



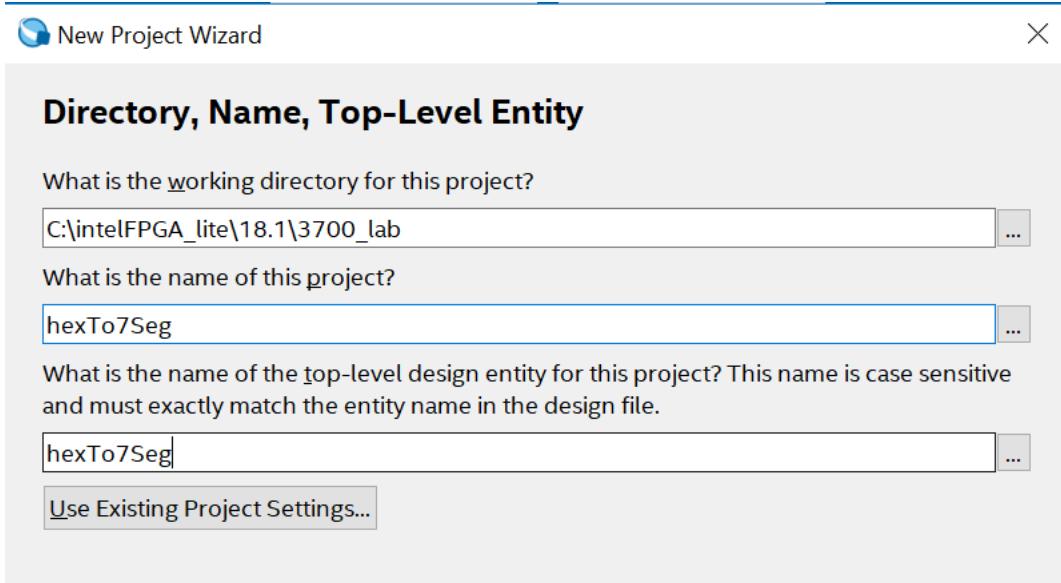
Section 2- Setup a new project in Quartus Lite Software

You begin this section by creating a new Quartus project. A project is a set of files that maintain information about your FPGA design. The Quartus Settings File (.qsf) and Quartus Project File (.qpf) files are the primary files in a Quartus project. To compile a design or make pin assignments, you must first create a project.

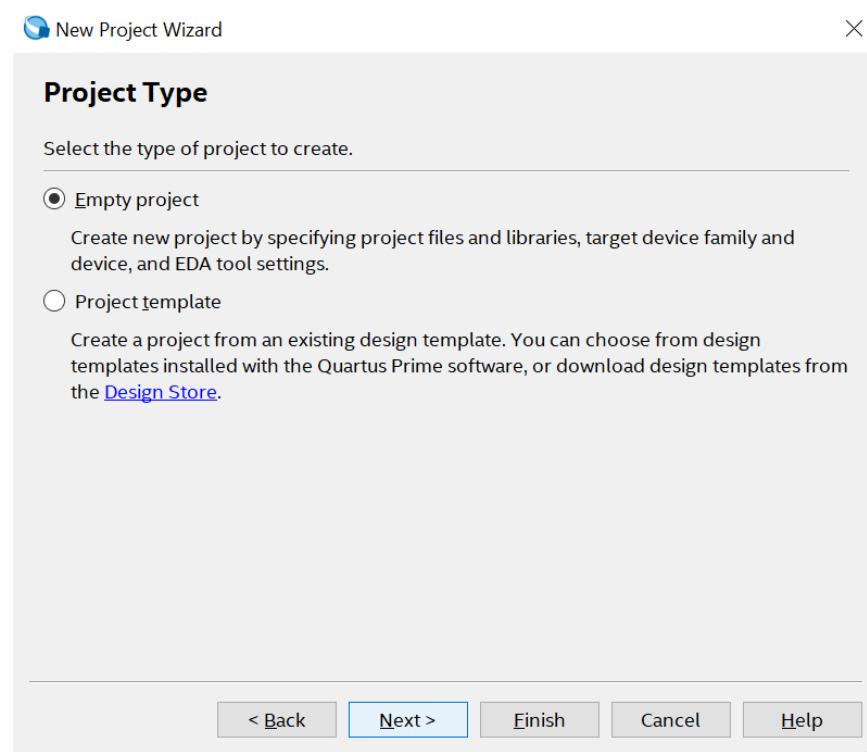
1. Launch the Quartus software, select File > New Project Wizard. The Introduction page opens



2. Enter the following information about your project as shown in the next snapshot:
 - a. What is the working directory for this project? Enter a directory in which you will store your Quartus project files for this design.
b. For example, C:\intelFPGA_lite\18.1\quartus\3700_lab\
c. File names, project names, and directories in the Quartus software cannot contain spaces.
d. What is the name of this project? Type hexTo7Seg.
e. What is the name of the top-level design entity for this project? Type hexTo7Seg.
f. Create a directory if it isn't present.



3. Select the Empty project template and click Next:



- Click Next twice and navigate to the Family, Device and Board Settings. Select the specifics on the respective device target as mentioned below (Device – **10M50DAF484C7G**). You can add the required Verilog files later or edit the files in the Quartus editor once the project has been setup.

The screenshot shows the Quartus Prime software interface for selecting a target device. The 'Device' tab is active. In the 'Device family' section, 'MAX 10 (DA/DF/DC/SA/SC)' is selected. Under 'Target device', the radio button for 'Specific device selected in 'Available devices' list' is checked. The 'Available devices' table lists four MAX 10 devices, with the 10M50DAF484C7G model highlighted in blue. The table columns include Name, Core Voltage, LEs, Total I/Os, GPIOs, Memory Bits, Embedded multiplier 9-bit elements, PLLs, and a header row with icons.

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit elements	PLLs																			
10M50DAF484C6GES	1.2V	49760	360	360	1677312	288	4	2																		
10M50DAF484C7G	1.2V	49760	360	360	1677312	288	4	2	10M50DAF484C8G	1.2V	49760	360	360	1677312	288	4	2	10M50DAF484C8GES	1.2V	49760	360	360	1677312	288	4	2
10M50DAF484C7G	1.2V	49760	360	360	1677312	288	4	2																		
10M50DAF484C8G	1.2V	49760	360	360	1677312	288	4	2																		
10M50DAF484C8GES	1.2V	49760	360	360	1677312	288	4	2																		

- Select the simulation Tool as ModelSim-Altera(note – this is not just ModelSim but the one with Altera) and format as Verilog HDL. Retain the other Tool selections to default.

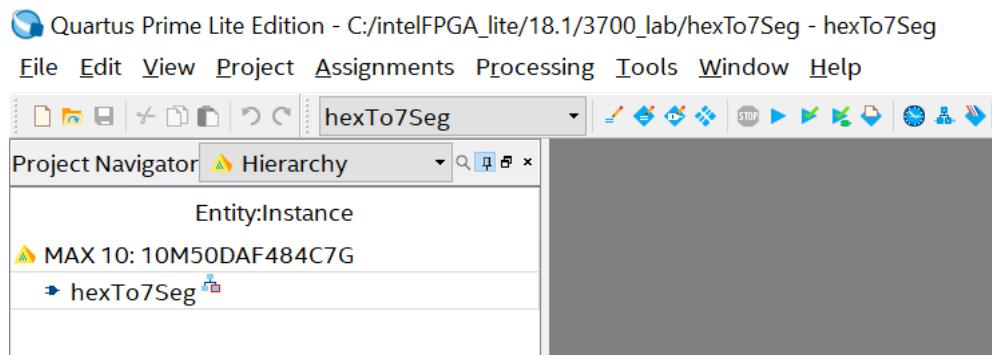
The screenshot shows the 'EDA Tool Settings' window within the 'New Project Wizard'. It lists various EDA tools and their configurations. The 'Tool Type' column includes 'Design Entry/Synthesis', 'Simulation', 'Board-Level', 'Timing', 'Symbol', 'Signal Integrity', and 'Boundary Scan'. The 'Tool Name' column shows 'ModelSim-Altera' for simulation, while others are set to '<None>'. The 'Format(s)' column shows 'Verilog HDL' for simulation and '<None>' for others. The 'Run Tool Automatically' section contains two checkboxes: 'Run this tool automatically to synthesize the current design' (unchecked) and 'Run gate-level simulation automatically after compilation' (unchecked).

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synthesis	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	Verilog HDL	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

6. Review the new project summary page and make sure all the specifics that you entered are reflected.

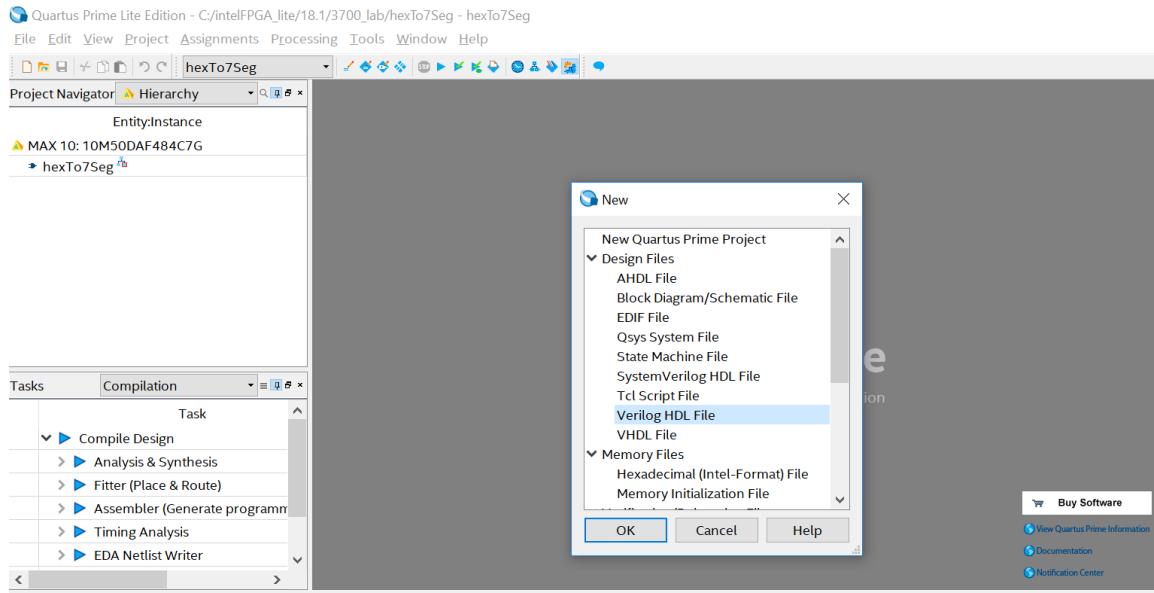
Summary	
When you click Finish, the project will be created with the following settings:	
Project directory:	C:\intelFPGA_lite\18.1\3700_lab
Project name:	hexTo7Seg
Top-level design entity:	hexTo7Seg
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	MAX 10 (DA/DF/DC/SA/SC)
Device:	10M50DAF484C7G
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	ModelSim-Altera (Verilog HDL)
Timing analysis:	()
Operating conditions:	
Core voltage:	1.2V
Junction temperature range:	0-85 °C

7. Quartus Lite Project navigator will now reflect the instance as your top level design entity under the device name.

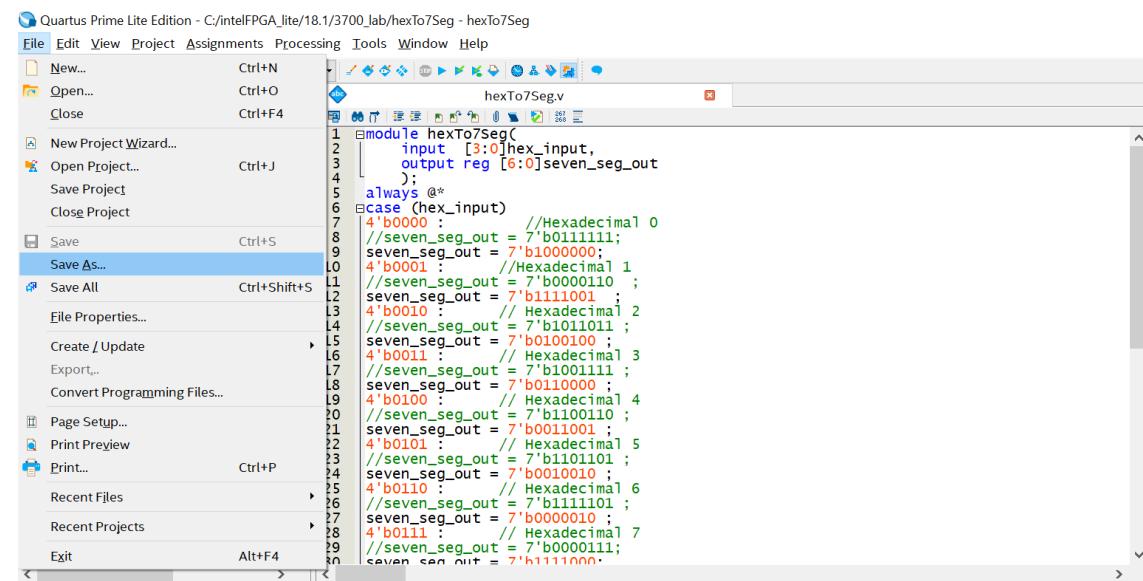


Section 3 – Implement a function using Verilog HDL

- Once you have created and setup your project, the next step is to describe your design using a hardware description language.
Go to File-->New→Select Verilog HDL File.

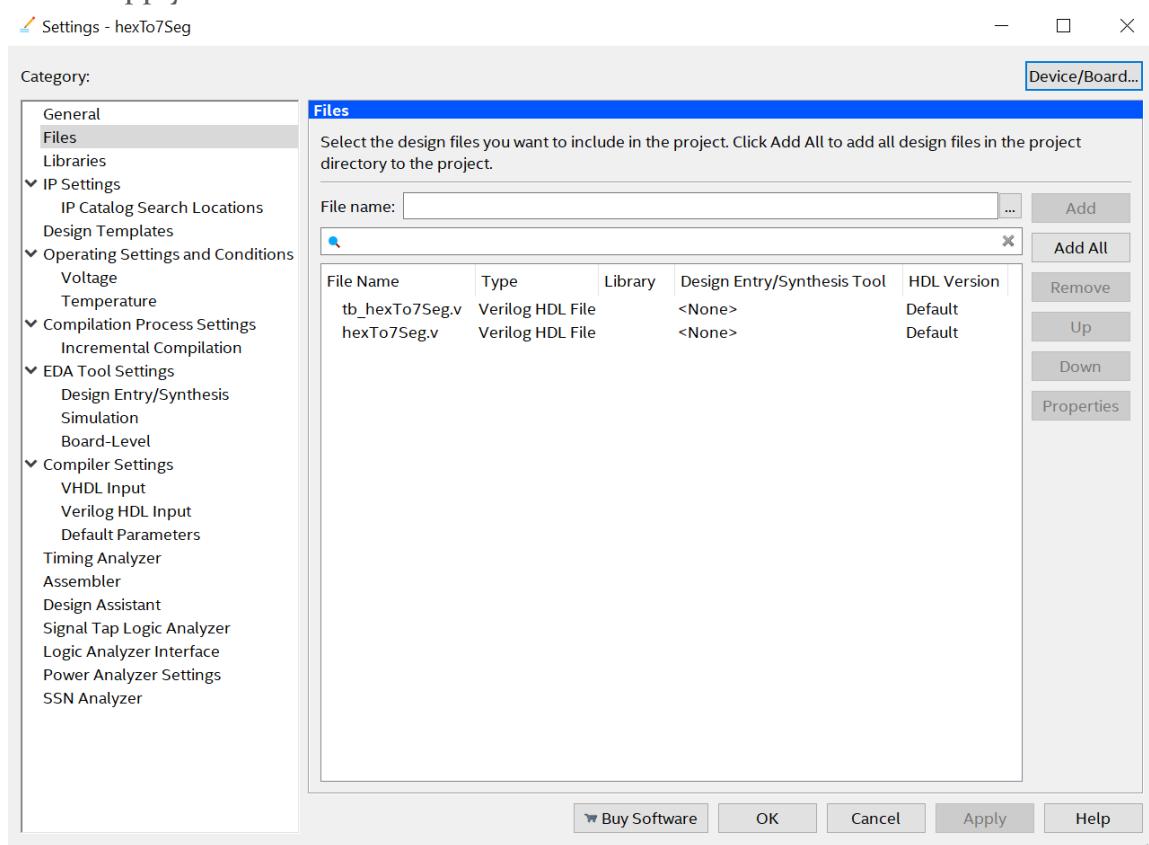


- Once a new file is created, you will be able to enter your code into it. When you save it, you have to have the module name and file name match up. Remember that one of the file names must match the name of the project. Double check and make sure that the Save as type is Verilog HDL Files.



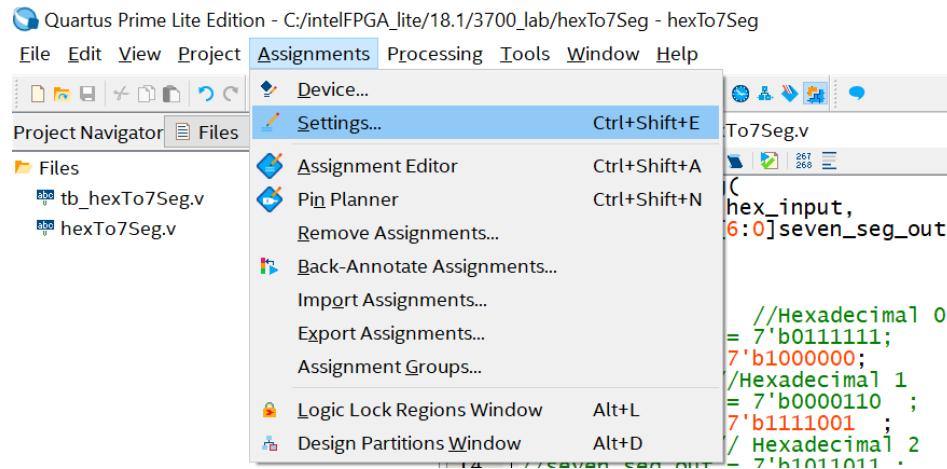
3. Every project in hardware needs a testbench to generate all necessary inputs and read outputs to ensure they are correct. This is very similar to writing test cases in software programming. The standard practice of naming a testbench is to add a "tb_" in front of the name of the module you are testing. In this case, we are testing hex to seven segment display, so the name of the new Verilog HDL file is saved as "tb_hexTo7Seg". A testbench is just another standard Verilog HDL File.

4. If you already have a file present in your directory, you can add/remove files to the project by navigating to “project->add/remove files in project”. Select the files -> apply -> ok.

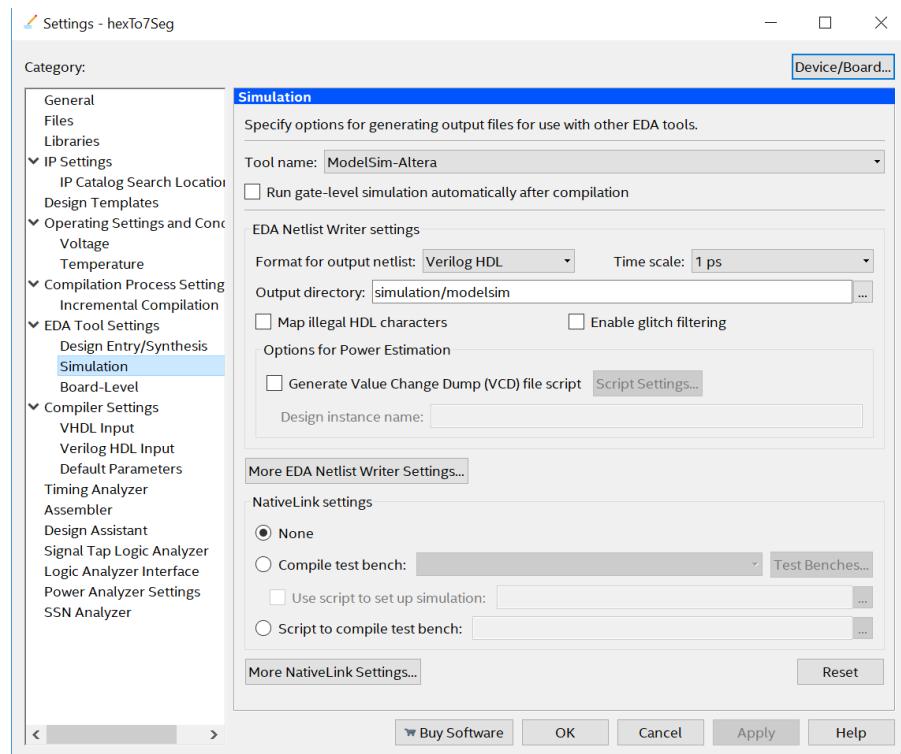


Section 4 - Simulate the Verilog circuit using ModelSim + Verilog test bench

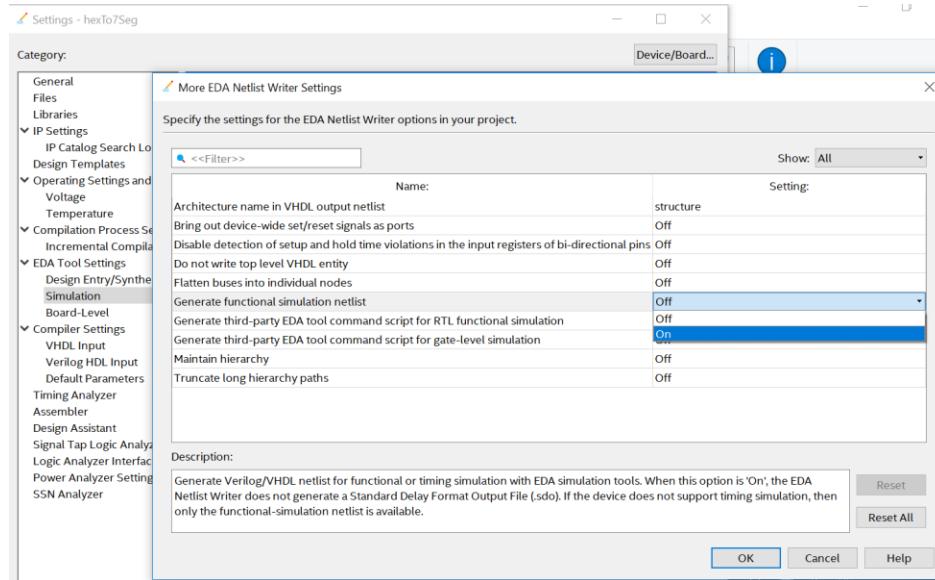
1. To setup the simulation Tool path for ModelSim-Altera, navigate to Assignments → Settings and modify the simulation parameters as shown below.



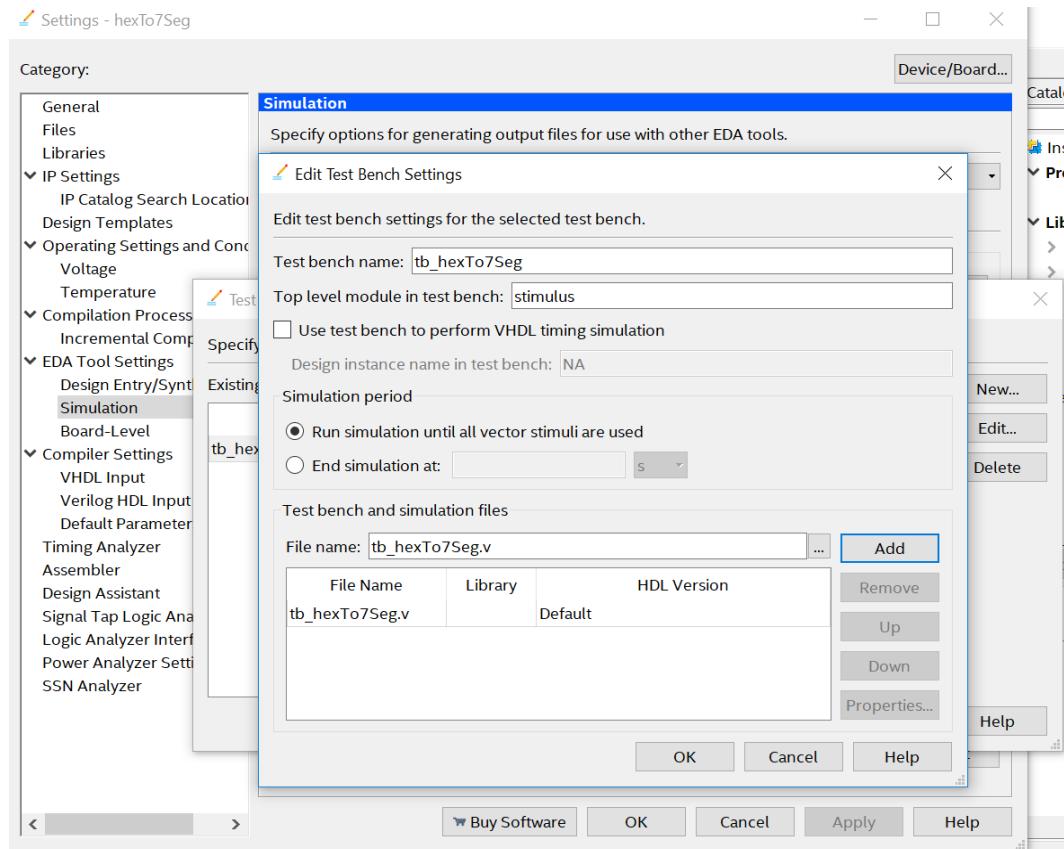
2. Under EDA Tool settings, check on Simulation option and it should show the Tool name as ModelSim-Altera.



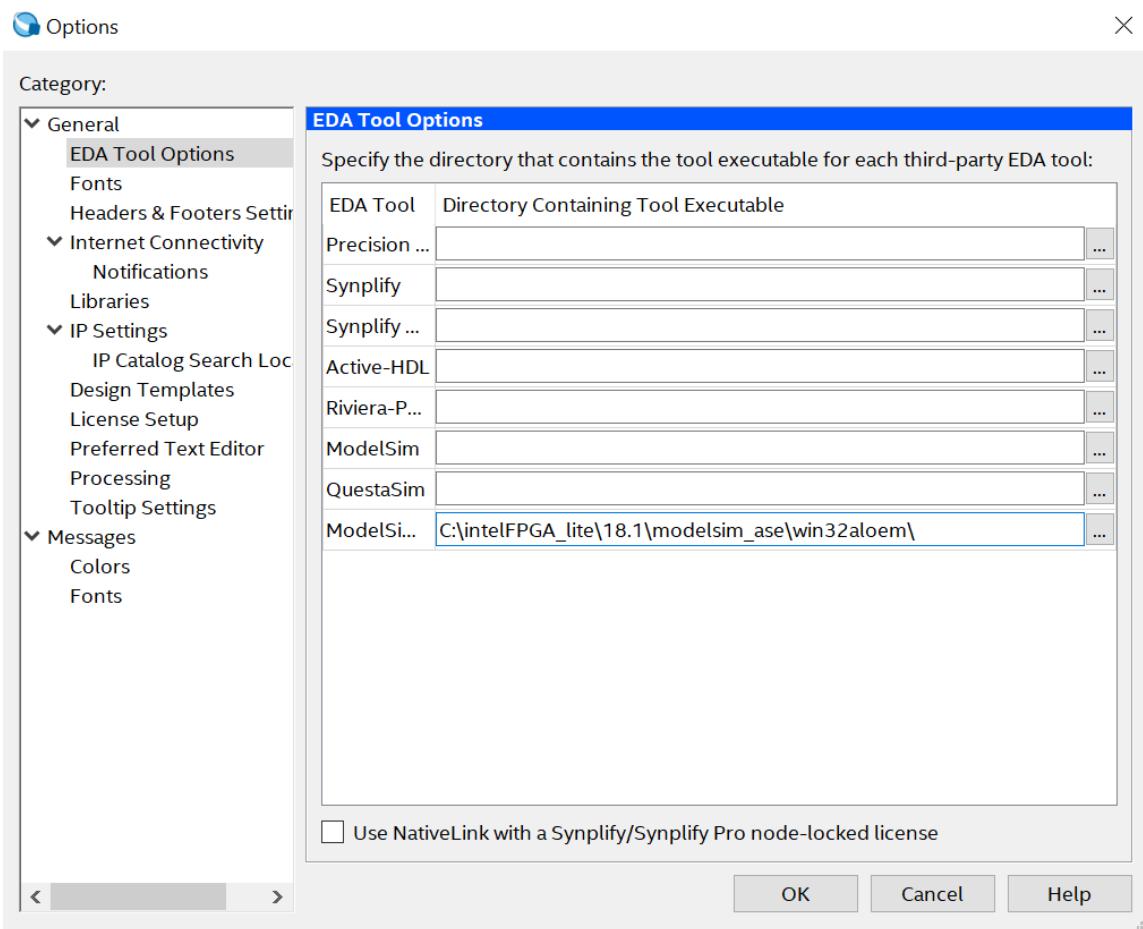
3. **Output directory** can be e.g. /simulation/modelsim. Then these new directories are created under the working directory, and the different Modelsim output files (.vo, .sdo) are placed here.
4. Navigate to More EDA Netlist Writer settings and set the generate functional simulation netlist option to ON: click OK and Apply.



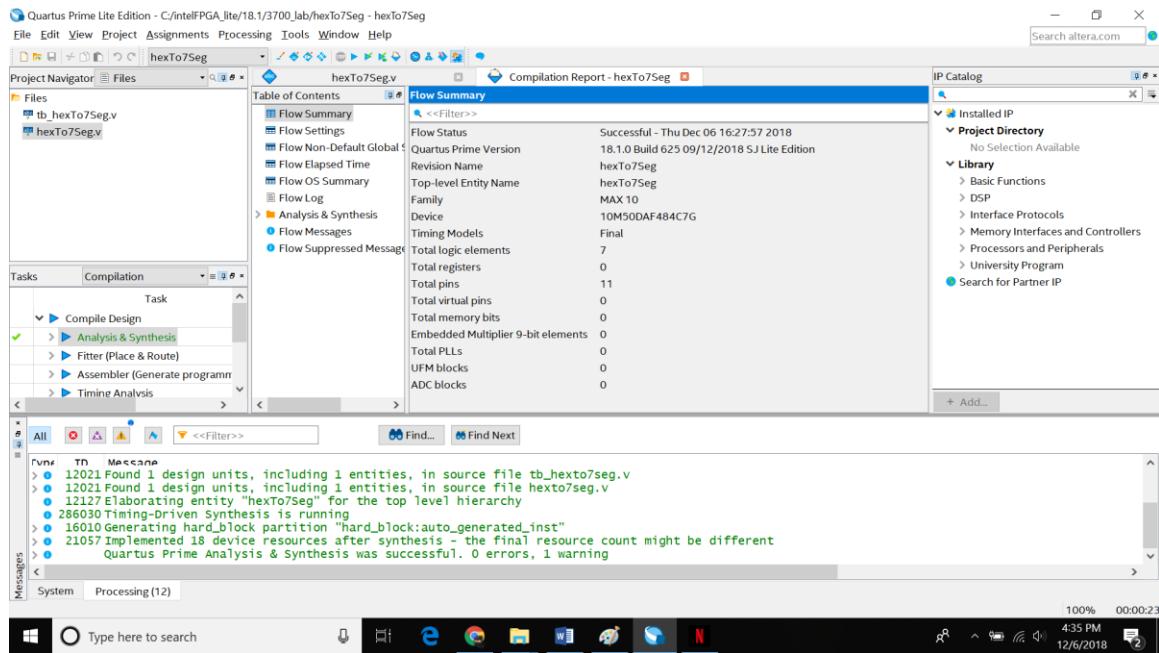
5. You can automate the modelsim execution (compile/simulate) by setting up the testbench parameters in quartus software as follows. Enter the information about the testbench file under NativeLink settings.
 - a. Select Compile test bench
 - b. Click Test Benches. The Test Benches dialog box appears.
 - c. Click New. The New Test Bench Settings dialog box appears.
 - d. Enter the test bench details from your design –
 - i. The Test bench name can be any suitable name by your choice. This name will later appear in the Compile test bench list.
 - ii. The Top level module in test bench must be the name of the entity of your testbench file.
 - e. You can also limit the simulation end time by entering a time limit in terms (us/ps/ms) in the “end simulation at” dialog box.



6. To setup the simulation executable path, click on Tools> Options> EDA Tool Options and set the Modelsim-Altera path to your installation directory.

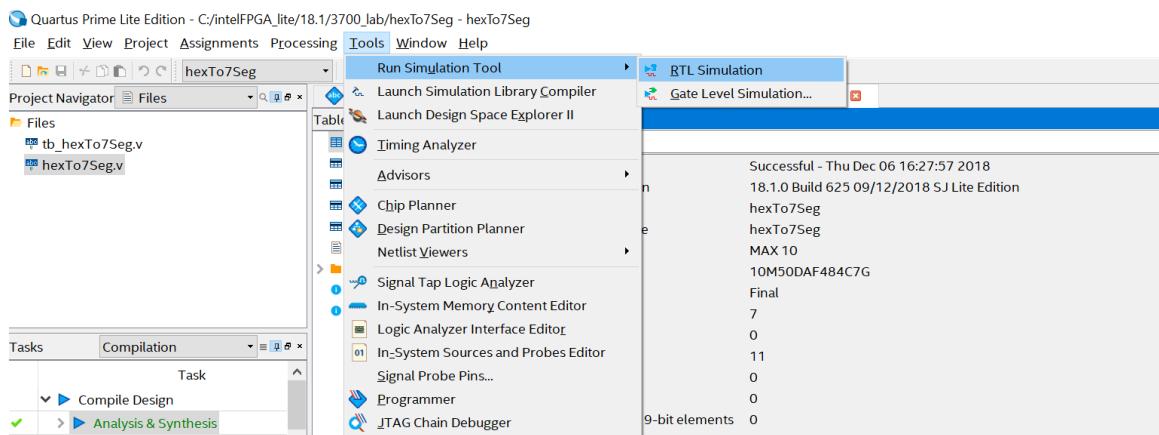


7. After you have finished coding up your modules, double click on Analysis & Synthesis under the Tasks pane. If you do not see Analysis & Synthesis, double check that Flow is set to Compilation. This will compile and synthesize your program(s). If there are no errors, you will see a pop-up saying the Analysis & Compilation was successful. If not, it will tell you your errors in the messages pane at the bottom of the screen. It is good habit to just review your warnings (if any) to ensure you have no latches or other design hazards.

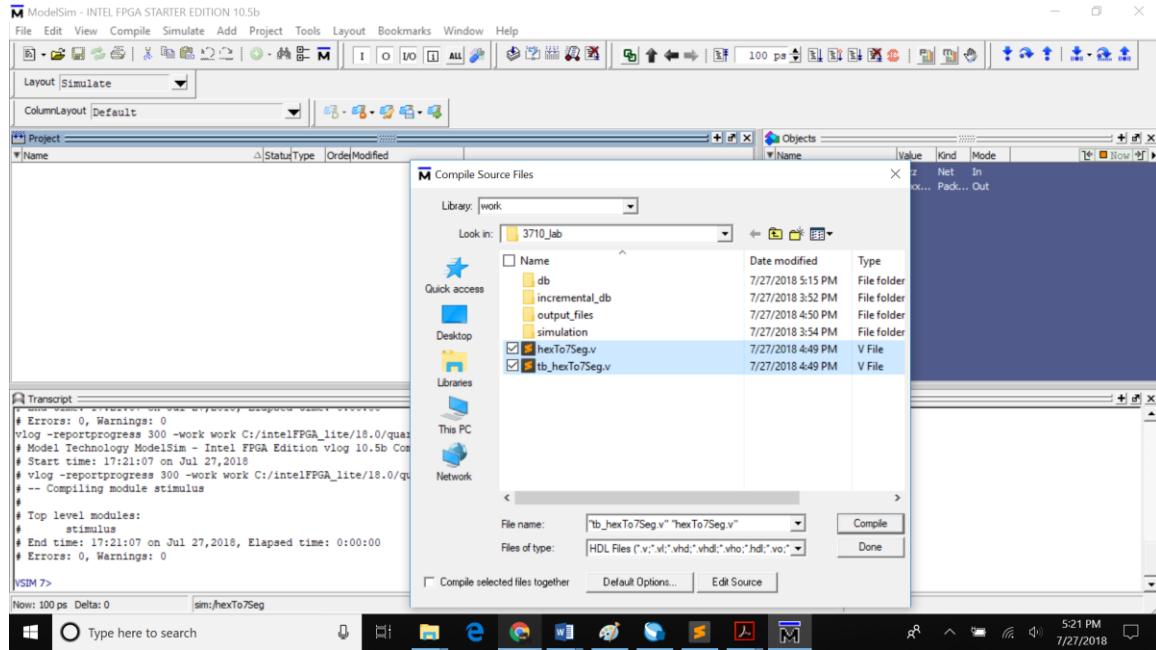


8. Once the Analysis & Synthesis is successful, you can do a RTL Simulation. Go to Tools-->Run Simulation Tool-->RTL Simulation.

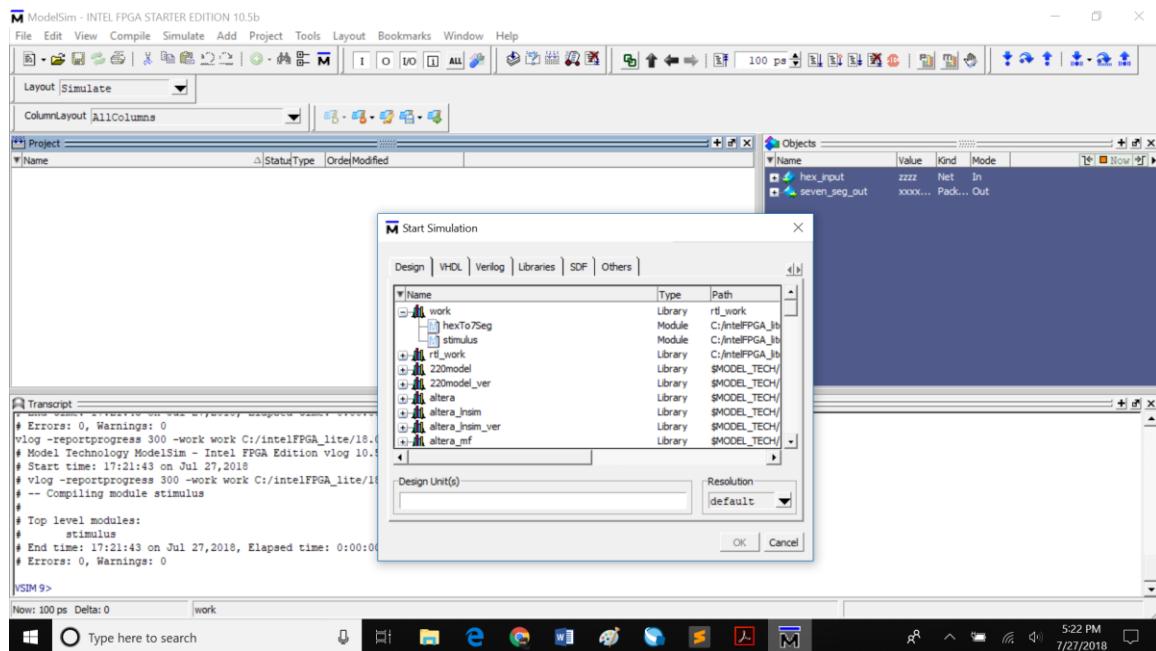
- Now, if you have setup the testbench setup in nativelink settings as described before, the modelsim window will automatically compile/simulate the testbench mentioned there and will dump the port level signals from the top level module onto waveform.
- If you have not mentioned any testbench settings, you can follow the steps from next bullet to compile/simulate the design in Modelsim.



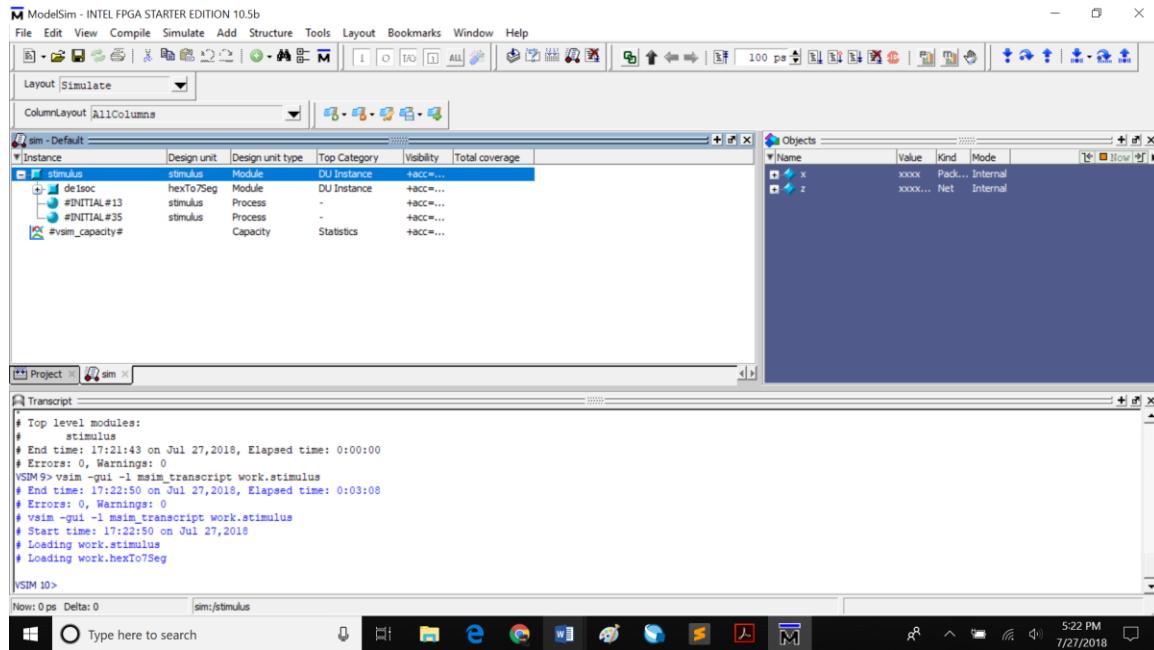
9. Go to Compile-->Compile. Ensure that the Library is work and navigate to your project directory. Select all Verilog HDL Files that pertain to your project. This includes the testbench. Hit Compile and then Done.



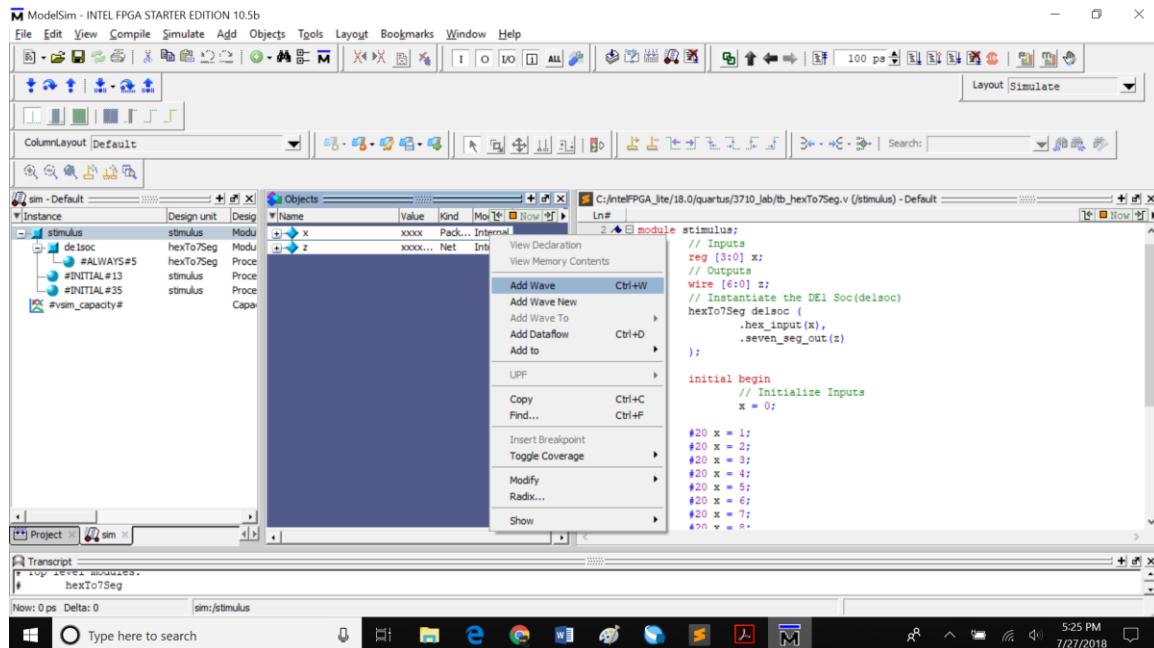
10. If you look at the bottom in the Transcript pane, you will see that it did compile and there were no errors. The work library should have all the files you just compiled. If not, repeat the previous step. Since the testbench does the signal generations and testing, double click on your testbench file.



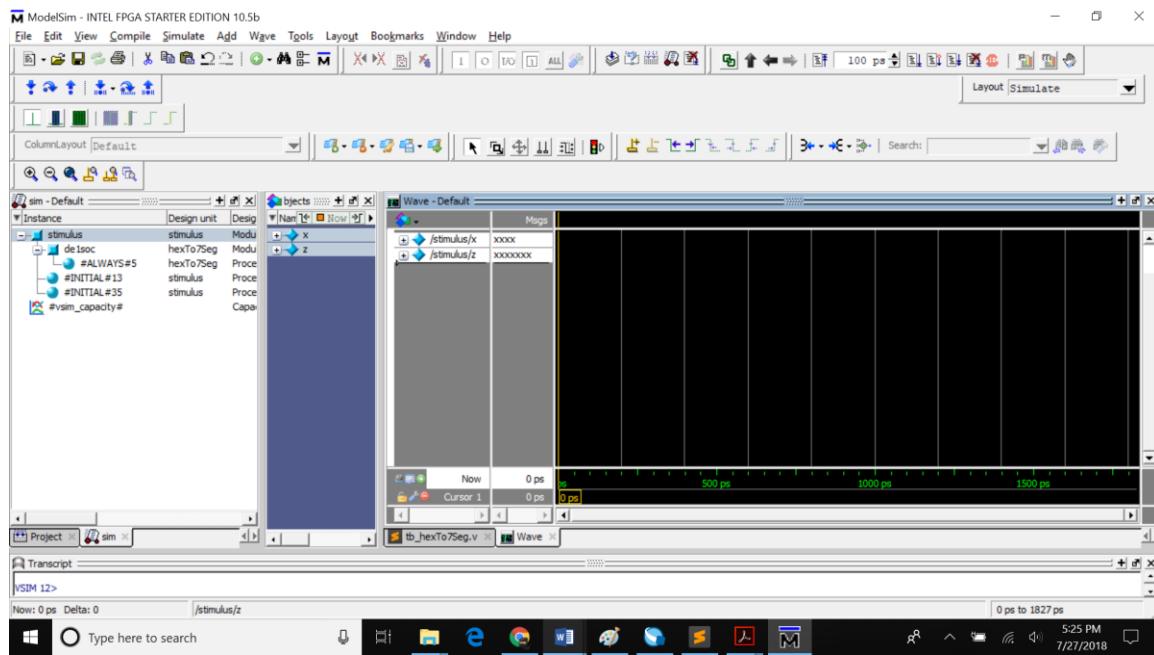
11. Click Simulate> Start simulation . Under work select the module to be simulated and click ok.



12. The next step is to add signals to the wave and show the wave if it is not already present. On the left, in the sim pane, right click on the testbench file which should be the top most file. Go to Add->To Wave->All signals in region.

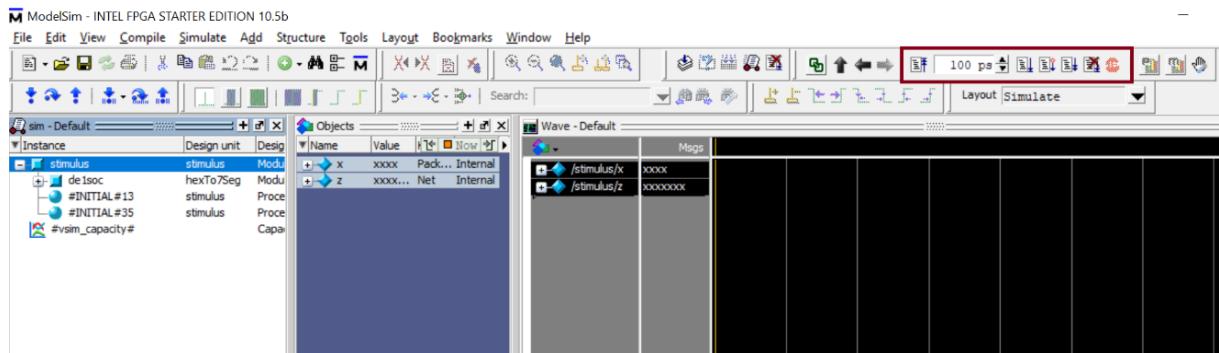


13. In the Wave pane, you will see all the signals declared and used in the testbench.

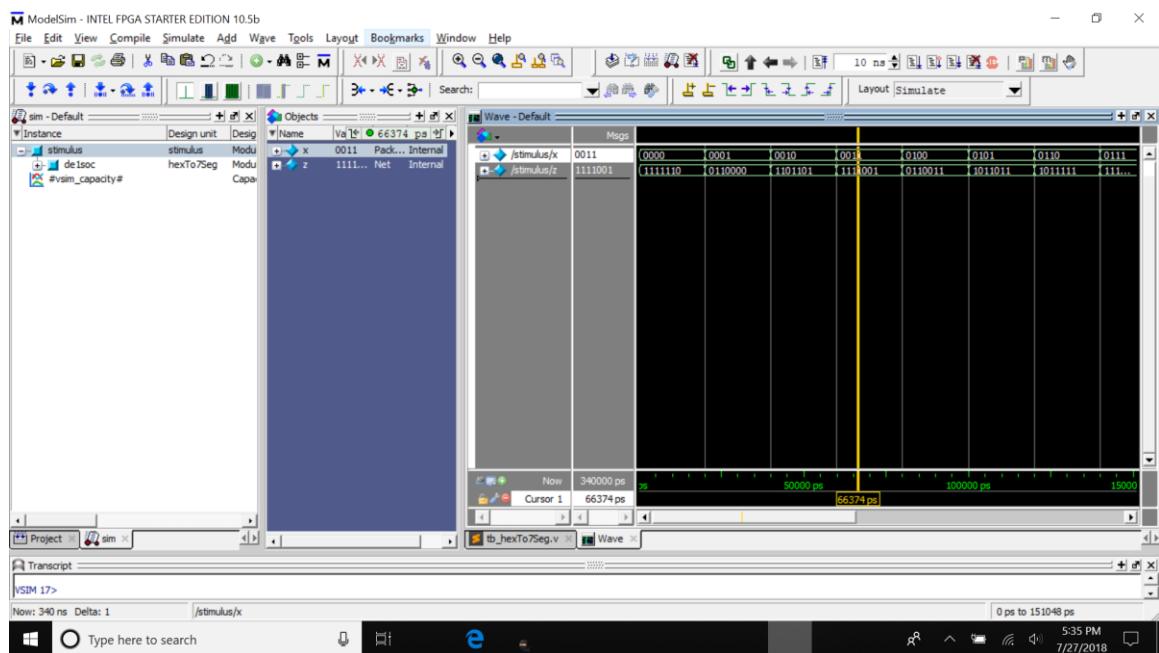


Note - When dealing with signals that are many bits, it is easier to see its value as an unsigned integer/Hex rather than binary. To make this conversion, right click on the signal you want, go to Radix and choose the format you want.

14. You are now ready to simulate your program. The icons boxed in the below screenshot are used to run the testbench. The first icon is Restart which will reset the simulation as if you never ran it. This is helpful to rerun the simulation without recompiling everything. The Run Length allows you to enter a specific amount of time you want the program to run for. It defaults to pico-seconds, but nano-seconds is the best time to use. The icon Run right after the Run Length is to run your program for the amount of time specified in the Run Length. If you set Run Length to be 10 ns, each time you press Run, the program will continue for 10 ns. Continue Run will run the program until it terminates. The same is true for Run -All. All the programs in this class will terminate in less than one second. If you find yourself waiting for longer than a few seconds until the program terminates, hit the Stop button and recheck your logic. You will see the following screen once your program terminates. It shows you where the program terminated. To go back to the Wave, click on the Wave tab.



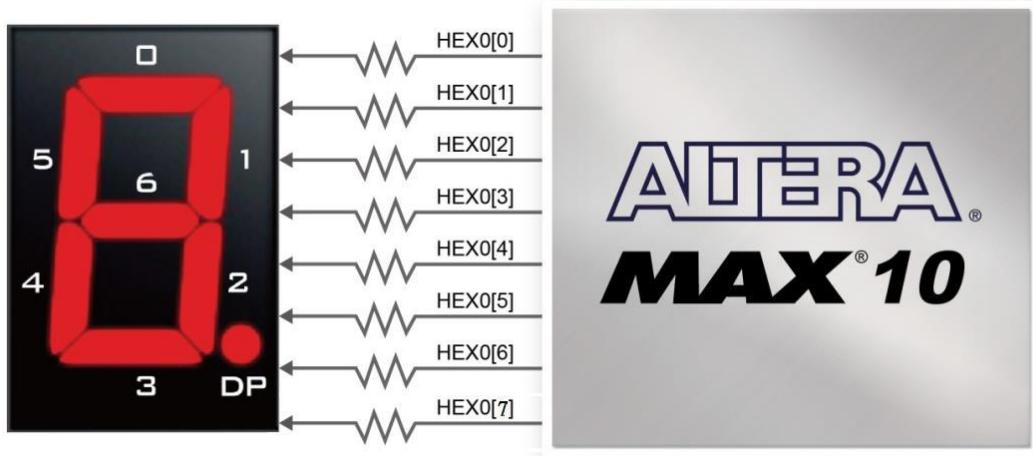
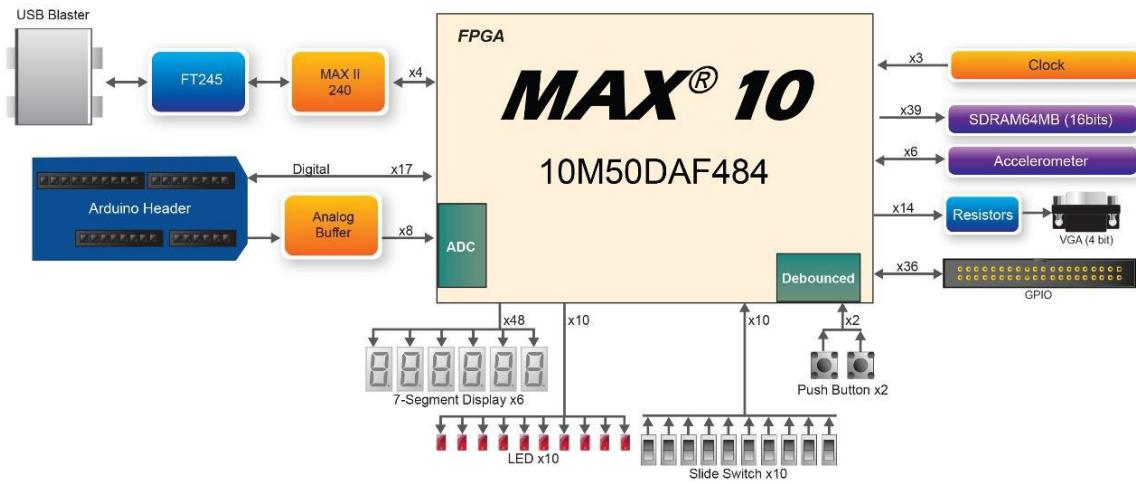
15. Once you click Run, you should see something like this on your Wave with values propagated and reflected on all the intended signals in the wave.



16. If you expand or scroll through the Transcript pane, you will see the output of any \$display statements you have in your code.

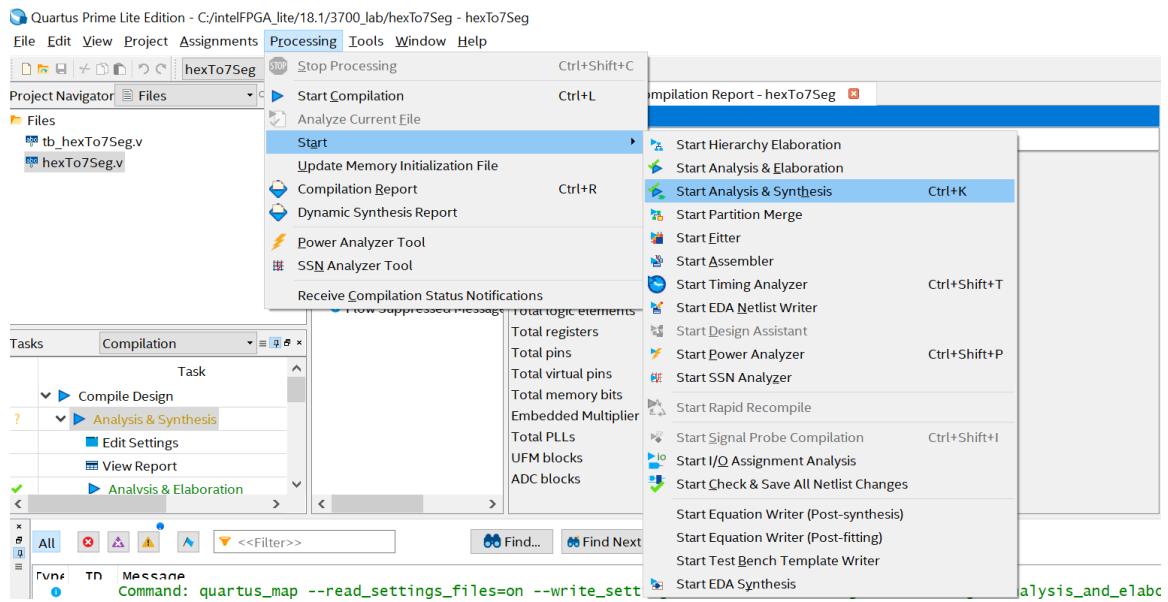
```
VSIM 13> run -continue
VSIM 14> run -all
# x=1,z=0110000
# x=2,z=1110101
# x=3,z=1111001
# x=4,z=0110011
# x=5,z=1011011
# x=6,z=1011111
# x=7,z=1111000
# x=8,z=1111111
# x=9,z=1111011
# x=a,z=1110111
# x=b,z=0011111
# x=c,z=10011110
# x=d,z=0111101
# x=e,z=10011111
# x=f,z=1000111
```

Section 5 – Pin assignment and Constraint generation

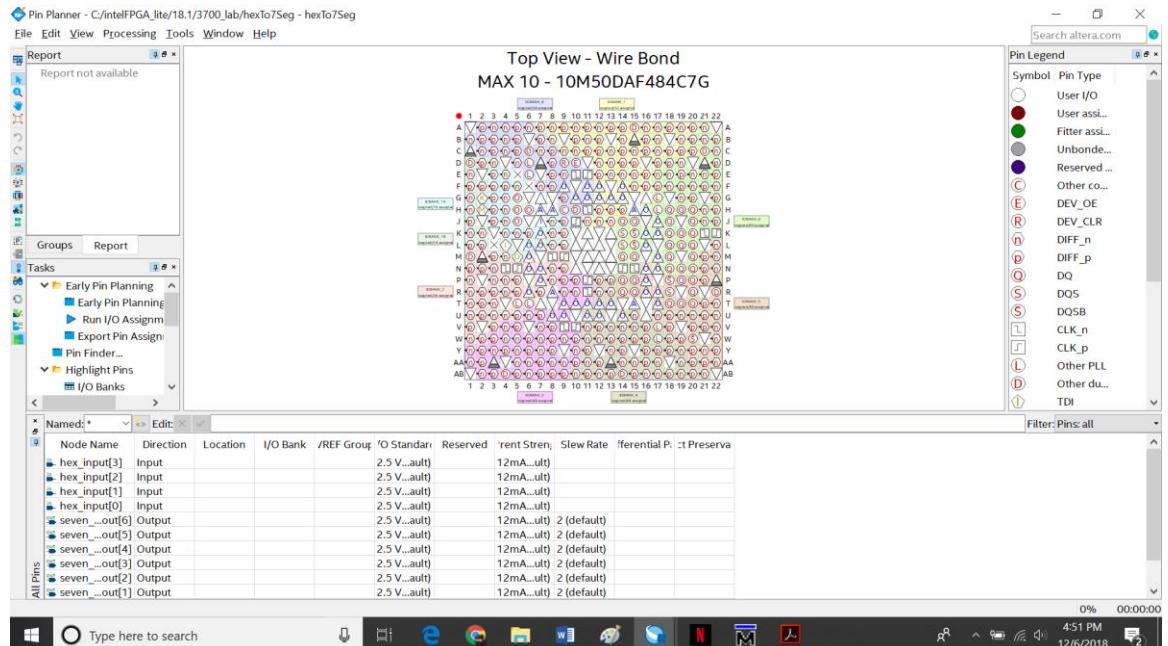


In this section, you will make pin assignments for all the signals from the design which interact with outer world. Before making pin assignments, perform the following steps:

1. Choose Processing > Start > Start Analysis & Synthesis in preparation for assigning pin locations. Click OK in the message window that appears after analysis and elaboration completes.



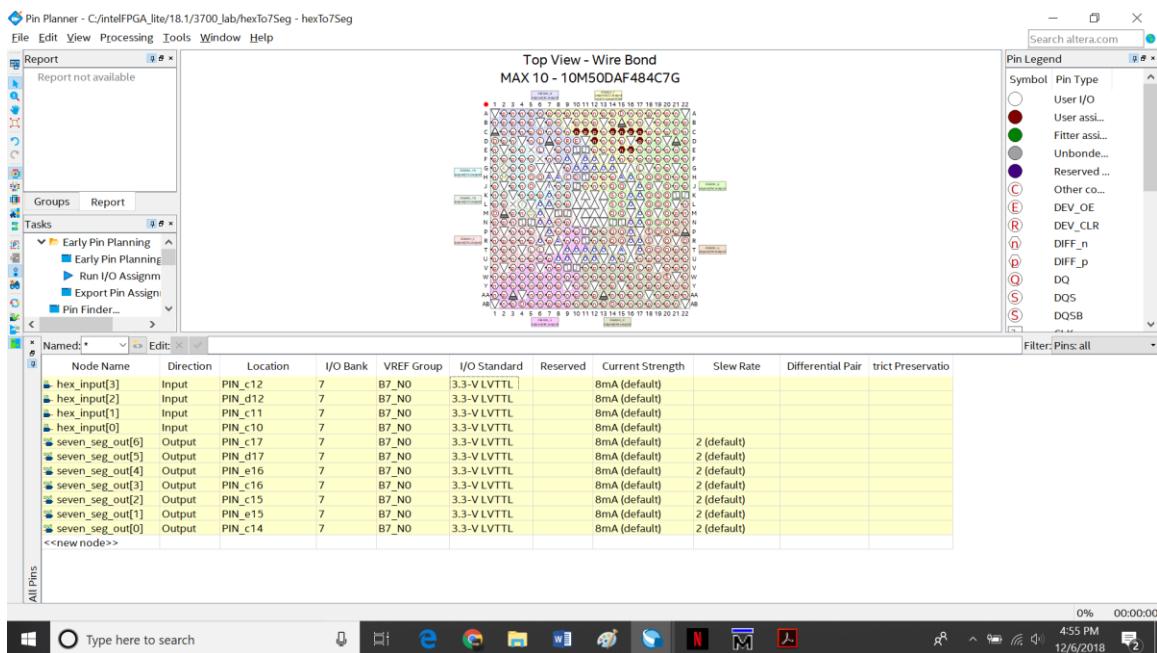
- To make pin assignments that correlate to the hex_input[3:0] input pins and seven_seg_out[6:0] output pins, perform the following steps: Choose Assignments > Pin Planner, which opens the Pin Planner, a spreadsheet-like table of specific pin assignments. The Pin Planner shows the designs pins.



- In the Location column next to each of the node names, add the coordinates(pin numbers) as shown from the tables below for the actual values to use with your DE1o-Lite board(check manual for any particular

pins you are looking for based on your design usage). Double-click in the Location column for any of the pins to open a drop-down list and type the pin number shown in the table. Alternatively, you can select the pin from a drop-down list. For example, if you type C12 and press the Enter key, the Quartus software fills in the full PIN_C12 location name for you. The software also keeps track of corresponding FPGA data such as the I/O bank and VREF Group. Each bank has a distinct color, which corresponds to the top-view wire bond drawing in the upper right window.

- After filling pin numbers for all ports, close the pin planner window.



Signal Name	FPGA Pin No.	Description	I/O Standard
HEX00	PIN_C14	Seven Segment Digit 0[0]	3.3-V LVTTL
HEX01	PIN_E15	Seven Segment Digit 0[1]	3.3-V LVTTL
HEX02	PIN_C15	Seven Segment Digit 0[2]	3.3-V LVTTL
HEX03	PIN_C16	Seven Segment Digit 0[3]	3.3-V LVTTL
HEX04	PIN_E16	Seven Segment Digit 0[4]	3.3-V LVTTL
HEX05	PIN_D17	Seven Segment Digit 0[5]	3.3-V LVTTL
HEX06	PIN_C17	Seven Segment Digit 0[6]	3.3-V LVTTL
HEX07	PIN_D15	Seven Segment Digit 0[7], DP	3.3-V LVTTL
HEX10	PIN_C18	Seven Segment Digit 1[0]	3.3-V LVTTL
HEX11	PIN_D18	Seven Segment Digit 1[1]	3.3-V LVTTL
HEX12	PIN_E18	Seven Segment Digit 1[2]	3.3-V LVTTL
HEX13	PIN_B16	Seven Segment Digit 1[3]	3.3-V LVTTL

HEX14	PIN_A17	Seven Segment Digit 1[4]	3.3-V LVTTL
HEX15	PIN_A18	Seven Segment Digit 1[5]	3.3-V LVTTL
HEX16	PIN_B17	Seven Segment Digit 1[6]	3.3-V LVTTL
HEX17	PIN_A16	Seven Segment Digit 1[7] , DP	3.3-V LVTTL
HEX20	PIN_B20	Seven Segment Digit 2[0]	3.3-V LVTTL
HEX21	PIN_A20	Seven Segment Digit 2[1]	3.3-V LVTTL
HEX22	PIN_B19	Seven Segment Digit 2[2]	3.3-V LVTTL
HEX23	PIN_A21	Seven Segment Digit 2[3]	3.3-V LVTTL
HEX24	PIN_B21	Seven Segment Digit 2[4]	3.3-V LVTTL
HEX25	PIN_C22	Seven Segment Digit 2[5]	3.3-V LVTTL
HEX26	PIN_B22	Seven Segment Digit 2[6]	3.3-V LVTTL
HEX27	PIN_A19	Seven Segment Digit 2[7] , DP	3.3-V LVTTL
HEX30	PIN_F21	Seven Segment Digit 3[0]	3.3-V LVTTL
HEX31	PIN_E22	Seven Segment Digit 3[1]	3.3-V LVTTL
HEX32	PIN_E21	Seven Segment Digit 3[2]	3.3-V LVTTL
HEX33	PIN_C19	Seven Segment Digit 3[3]	3.3-V LVTTL
HEX34	PIN_C20	Seven Segment Digit 3[4]	3.3-V LVTTL
HEX35	PIN_D19	Seven Segment Digit 3[5]	3.3-V LVTTL
HEX36	PIN_E17	Seven Segment Digit 3[6]	3.3-V LVTTL
HEX37	PIN_D22	Seven Segment Digit 3[7] , DP	3.3-V LVTTL
HEX40	PIN_F18	Seven Segment Digit 4[0]	3.3-V LVTTL
HEX41	PIN_E20	Seven Segment Digit 4[1]	3.3-V LVTTL
HEX42	PIN_E19	Seven Segment Digit 4[2]	3.3-V LVTTL
HEX43	PIN_J18	Seven Segment Digit 4[3]	3.3-V LVTTL
HEX44	PIN_H19	Seven Segment Digit 4[4]	3.3-V LVTTL
HEX45	PIN_F19	Seven Segment Digit 4[5]	3.3-V LVTTL
HEX46	PIN_F20	Seven Segment Digit 4[6]	3.3-V LVTTL
HEX47	PIN_F17	Seven Segment Digit 4[7] , DP	3.3-V LVTTL
HEX50	PIN_J20	Seven Segment Digit 5[0]	3.3-V LVTTL
HEX51	PIN_K20	Seven Segment Digit 5[1]	3.3-V LVTTL
HEX52	PIN_L18	Seven Segment Digit 5[2]	3.3-V LVTTL
HEX53	PIN_N18	Seven Segment Digit 5[3]	3.3-V LVTTL
HEX54	PIN_M20	Seven Segment Digit 5[4]	3.3-V LVTTL
HEX55	PIN_N19	Seven Segment Digit 5[5]	3.3-V LVTTL
HEX56	PIN_N20	Seven Segment Digit 5[6]	3.3-V LVTTL
HEX57	PIN_L19	Seven Segment Digit 5[7] , DP	3.3-V LVTTL

Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTL

5. The <design>.qsf file should reflect these pin assignments once you have set them on pin planner.

```

set_location_assignment PIN_C14 -to seven_seg_out[0]
set_location_assignment PIN_E15 -to seven_seg_out[1]
set_location_assignment PIN_C15 -to seven_seg_out[2]
set_location_assignment PIN_C16 -to seven_seg_out[3]
set_location_assignment PIN_E16 -to seven_seg_out[4]
set_location_assignment PIN_D17 -to seven_seg_out[5]
set_location_assignment PIN_C17 -to seven_seg_out[6]
set_location_assignment PIN_C10 -to hex_input[0]
set_location_assignment PIN_C11 -to hex_input[1]
set_location_assignment PIN_D12 -to hex_input[2]
set_location_assignment PIN_C12 -to hex_input[3]

```

6. Timing settings are critically important for a successful design. For this tutorial you will create a basic Synopsys Design Constraints File (.sdc) that the Quartus TimeQuest Timing Analyzer uses during design compilation. For more complex designs and requirements, you will need to create your .sdc files by considering the timing requirements more carefully.

To create an SDC, perform the following steps:

- Open the TimeQuest Timing Analyzer by choosing Tools > TimeQuest Timing Analyzer.
- Choose File > New SDC file. The SDC editor opens in the quartus software with a file extension as .sdc.
- Type in your timing constraints(clock/pll) into the file and save it as top-level file(hexTo7Seg.sdc in this case). Since the current design doesn't have a clock, we will ignore the constraints. Refer the DE10-Lite manual for creating a PLL/clock as it is explained in detail.
- Naming the SDC with the same name as the top-level file except for the .sdc extension causes the Quartus software to using this timing analysis file automatically by default. If you used another name, you would need to add the SDC to the assignments file list.

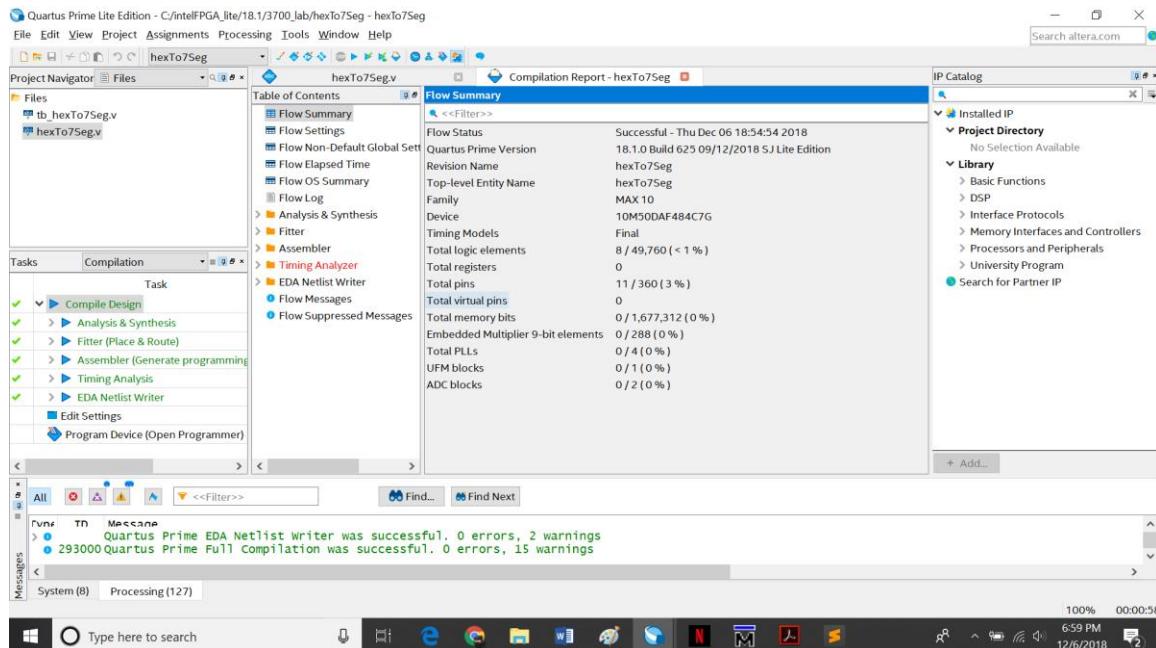
Section 6 – Synthesize, Implement, Generate, and Program DE1o-Lite board

After creating your design you must compile it to convert the design into a bitstream that can be downloaded into the FPGA. The most important output of compilation is an SRAM Object File (.sof), which you use to program the device. The software also generates other report files that provide information about your code as it compiles. If you want to store “*.SOF” in memory device (such as flash or EEPROMs), you must first convert the SOF to a file type specifically for the targeted device.

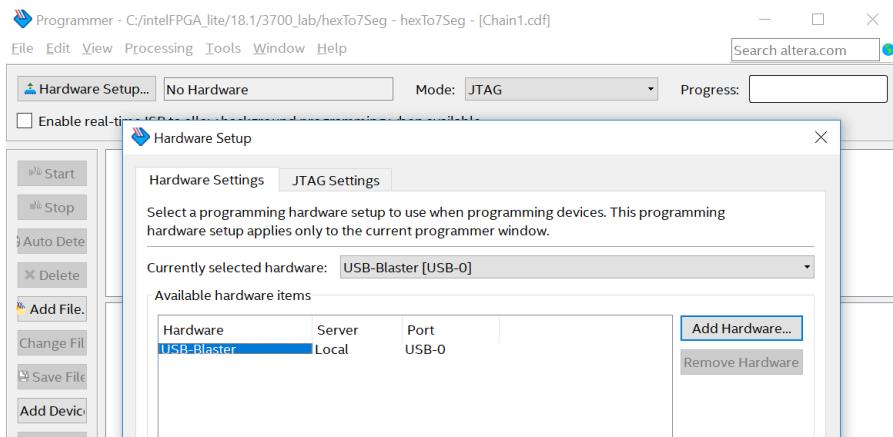
Now that you have created a complete Quartus project and entered all assignments, you can compile the design.

1. In the task window – double click compile design or
2. In the Processing menu, choose Start Compilation or
3. click the Play button on the toolbar (cntrl+L).

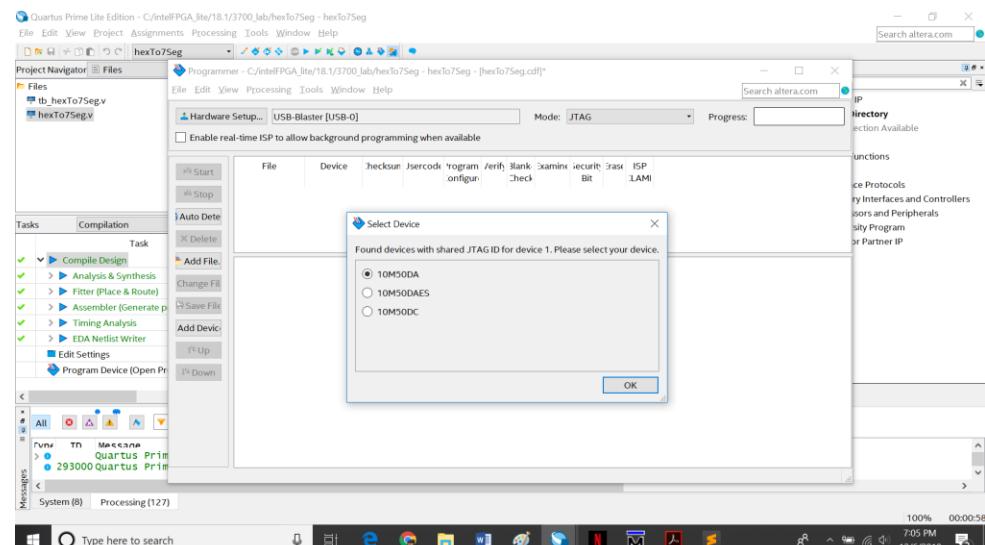
The Quartus Messages window displays many messages during compilation. It should not display any critical warnings; it may display a few warnings that indicate that the device timing information is preliminary or that some parameters on the I/O pins used for the LEDs were not set. The software provides the compilation results in the Compilation Report tab as shown.



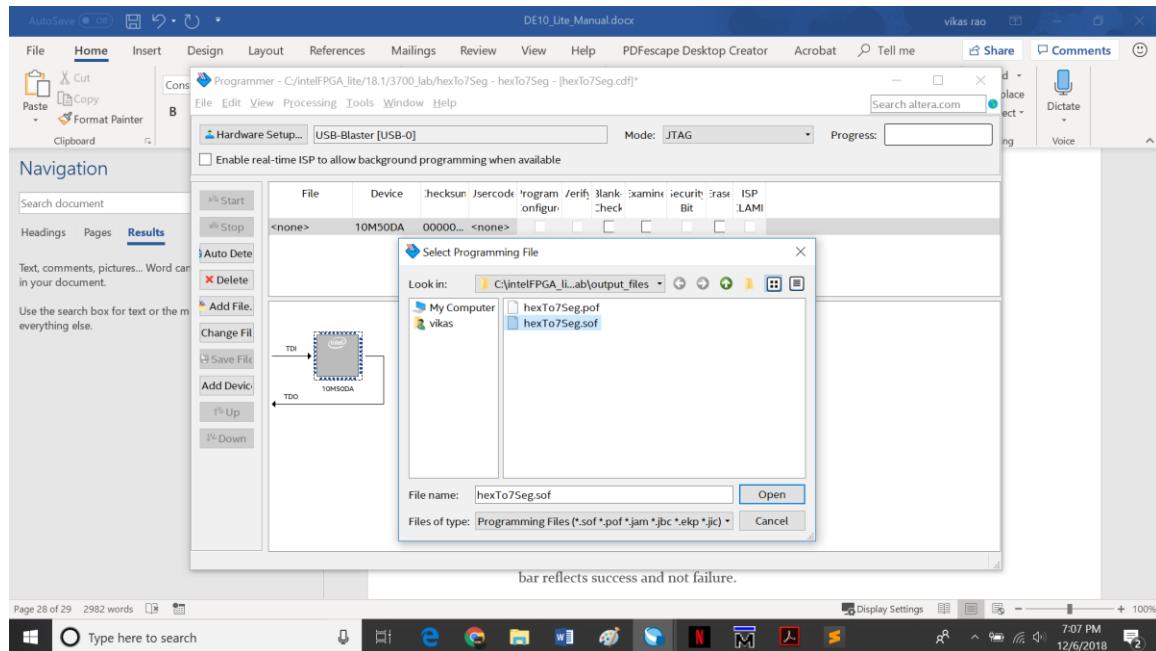
4. After compiling and verifying your design you are ready to program the FPGA on the development board. You download the SOF you just created into the FPGA using the USB-Blaster circuitry on the board. Set up your hardware for programming using the following steps:
 - a. For the DE1o-Lite board, connect the USB-Blaster (included in your development kit) to pin J₃ and the other end of the USB cable to the host computer.
5. Program the FPGA using the following steps.
 - a. Choose Tools > Programmer. Once the Programmer window opens, click Hardware Setup. If it is not already turned on, turn on the DE1o-Lite [USB-0] option under currently selected hardware.



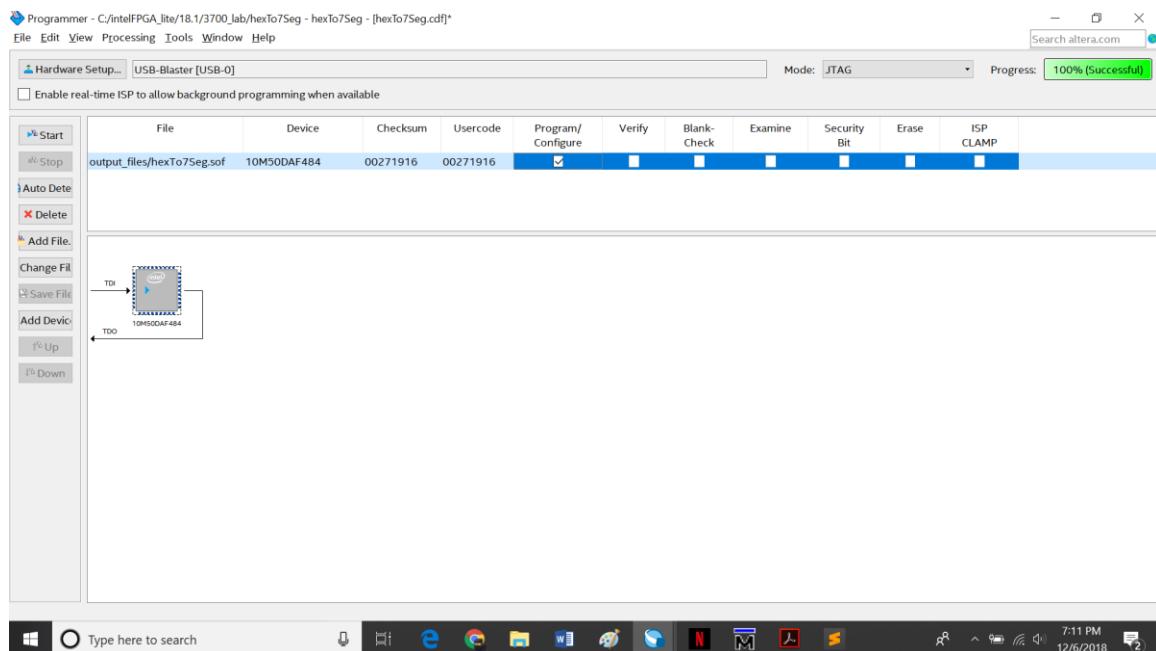
- b. Make sure the Mode is set to JTAG.
- c. Click Auto Detect to detect all the devices on the JTAG chain. Select the device as 10M50DA which reflects the device ID for DE1o-Lite MAX-10 board. Click OK after the selection.



- Select FPGA device and click Add File to program the FPGA with the relevant .sof file(hexTo7Seg.sof in this case).



- click Start to program the .sof file into FPGA and make sure the progress bar reflects success and not failure.



8. When you verify the design in hardware, observe the runtime behavior of the FPGA hardware and ensure that it is functioning appropriately.
 - a. Play around with the switches (SW0-SW3) and see the 7-segment display varying .

“Congratulations, you have created, compiled, and programmed your first FPGA design! The compiled SRAM Object File (.sof) is loaded onto the FPGA on the development board and the design should be running.”

References and miscellaneous links –

- Altera Quartus Lite software: http://fpgasoftware.intel.com/i8.1/quartus_lite
- User manual/datasheets/demonstrations/control-panel/sample codes:
 - Click the link below (<http://DE10-Lite.terasic.com/cd>) and download the latest system build([DE10-Lite v.2.0.3 SystemCD.zip](#)) with Quartus 16.0(>) support.
- For enabling inbuilt Linux Subsystem in windows please follow the instructions mentioned in the article, it works like a charm:
<https://www.windowscentral.com/how-install-bash-shell-command-line-windows-10>