University of British Columbia
Electrical and Computer Engineering
Digital Design and Microcomputers CPEN312

L06: Introduction to Quartus Prime and VHDL.

Dr. Jesús Calviño-Fraga. P.Eng.
Department of Electrical and Computer Engineering, UBC
Office: KAIS 3024
E-mail: jesusc@ece.ubc.ca
Phone: (604)-827-5387

January 19, 2017

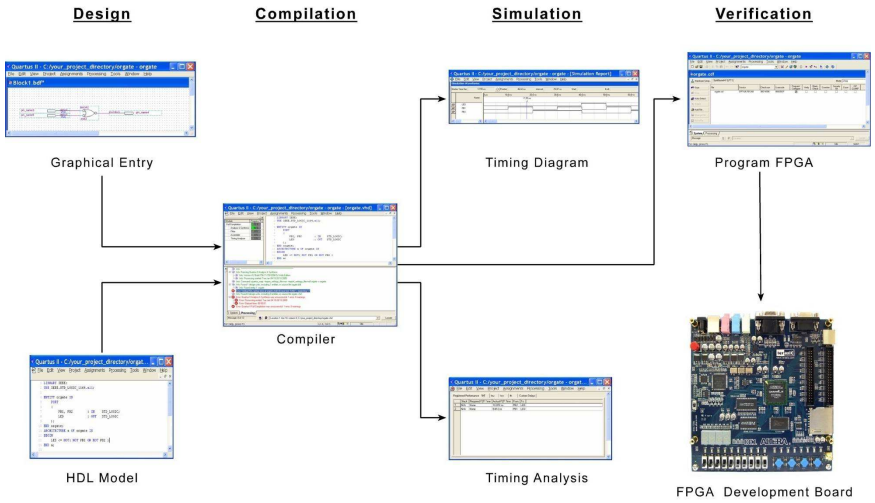
Objectives

- Create digital circuits with Quartus Prime in graphical form.
- Simulate digital circuits.
- Load and test digital circuits into the Altera DE0-CV board.
- Create digital circuits with Quartus Prime using VHDL

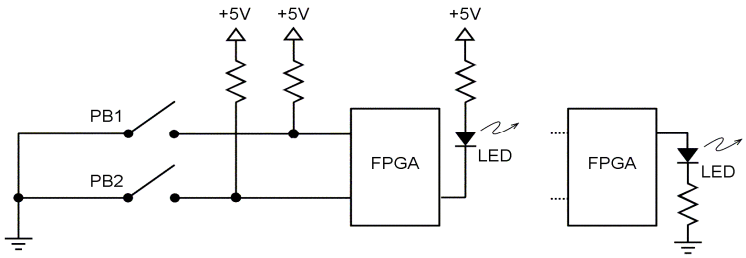
Create Digital Circuits Using the Graphic Editor in Quartus Prime

- We will be using Quartus Prime V16
- This updated procedure is based on chapter 1 of *Rapid Prototyping of Digital Systems* [SOPC edn] - J. Hamblen, et al., (Springer, 2008). Available in Connect. Unfortunately, this chapter was written for Quartus II V7. Many things have changed since that version!
- Some pop-up windows have been edited so they fit in the slides!

Design Process for Schematic or VHDL.

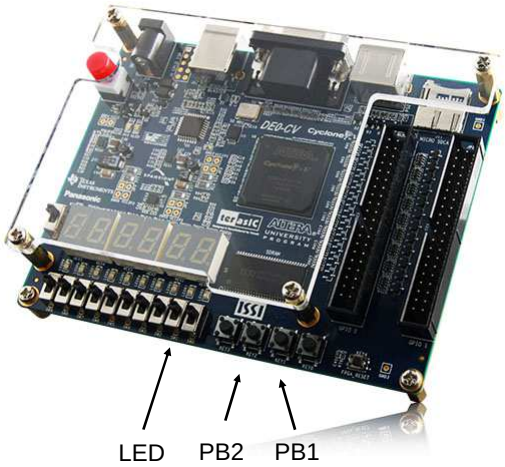


Simple Design

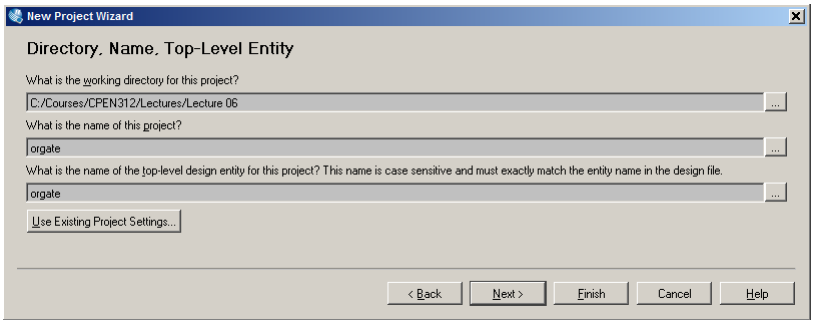


Whenever PB1 or PB2 are pressed the LED turns on

Altera DE0-CV Board

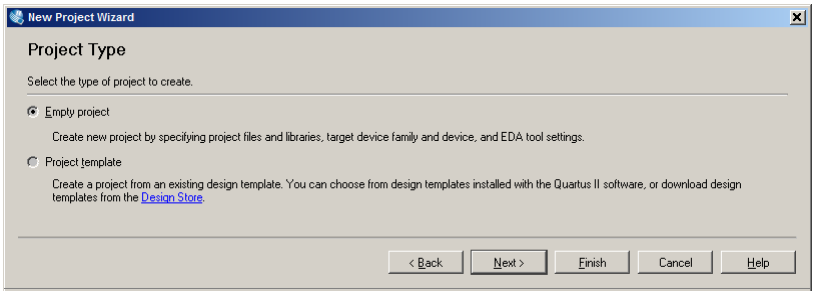


New Project Wizard



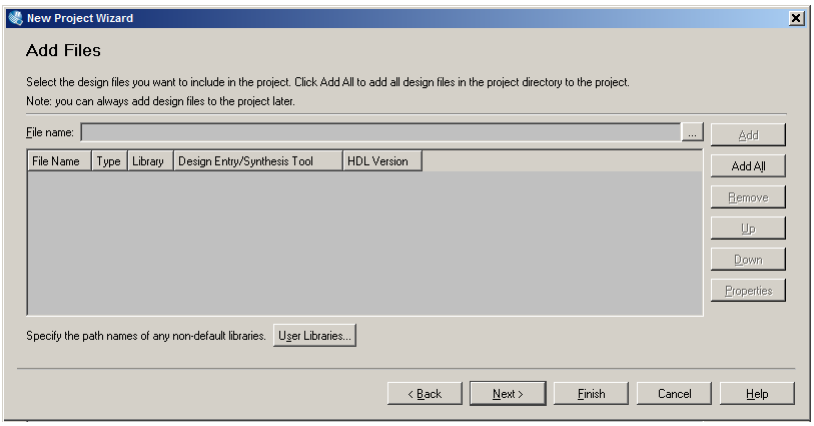
Pick a folder in your computer. For now, use the same name for the second and third field. In my case, I choose ‘orgate’

New Project Wizard



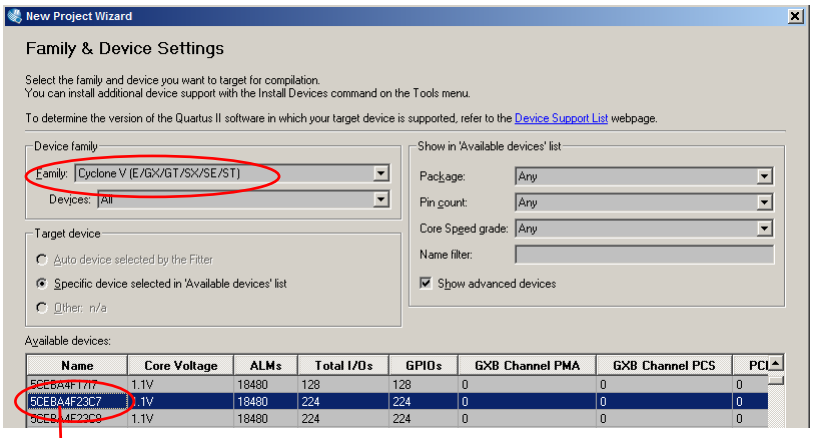
Empty Project…

New Project Wizard



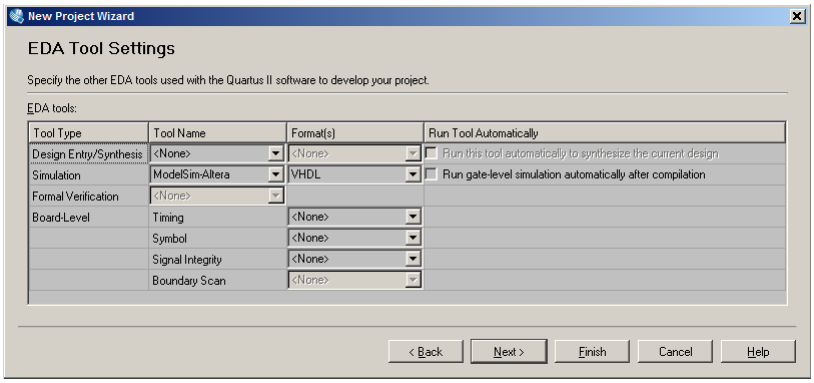
Nothing to change here. Click 'Next'.

New Project Wizard. Select the Device in the Altera DE0-CV…



5CEBA4F23C7

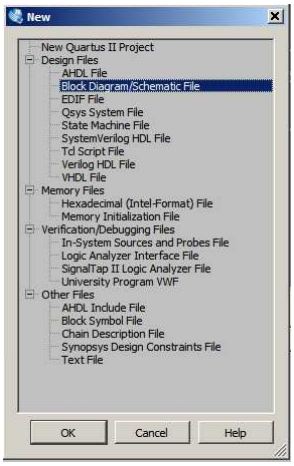
New Project Wizard



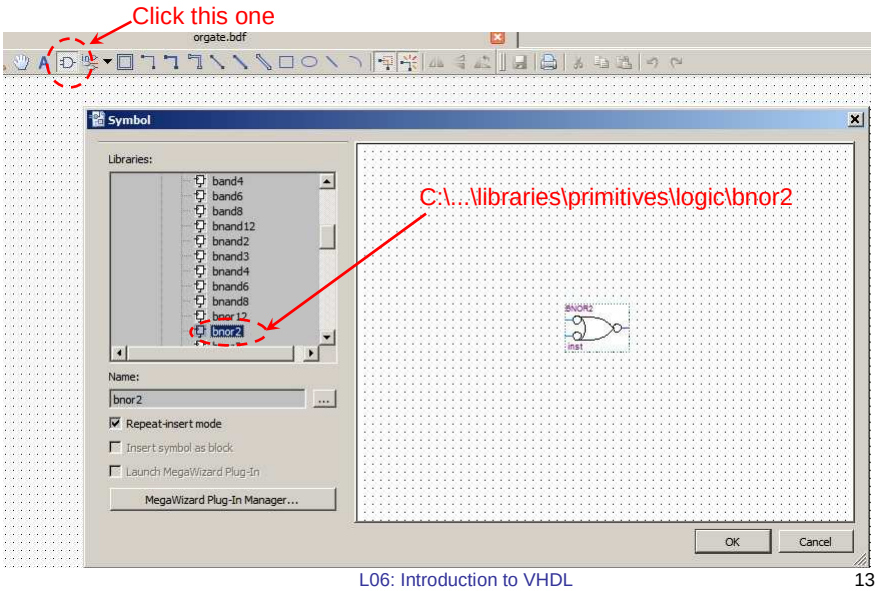
I had no idea what was going on here, so clicked 'Finish'.

Create the Schematic File.

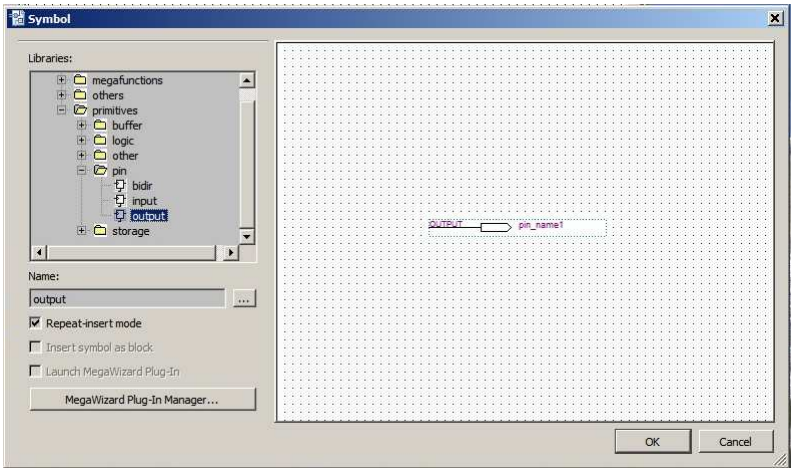
Click 'File'->'New'->'Block Diagram/Schematic File'



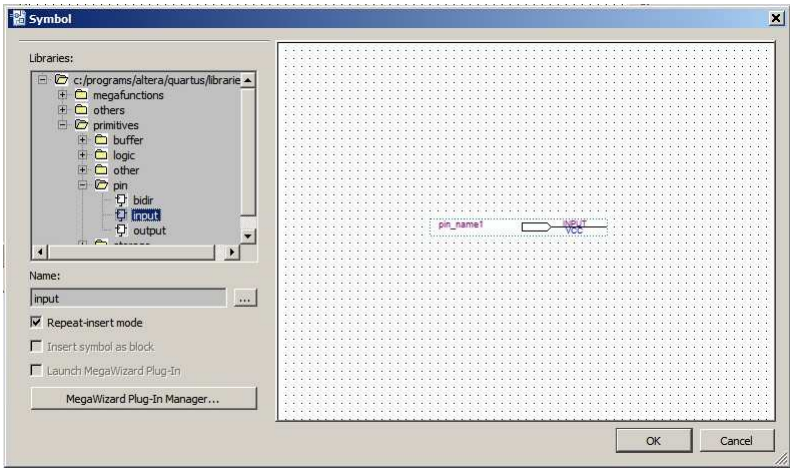
Add Gate to Schematic



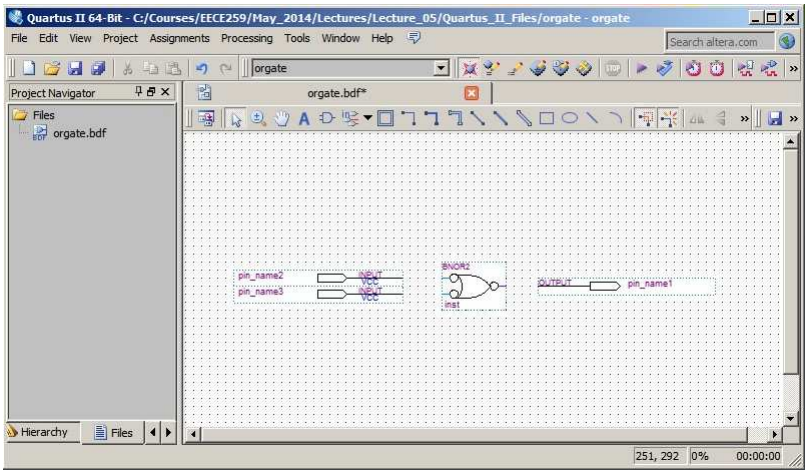
Add the Output Port



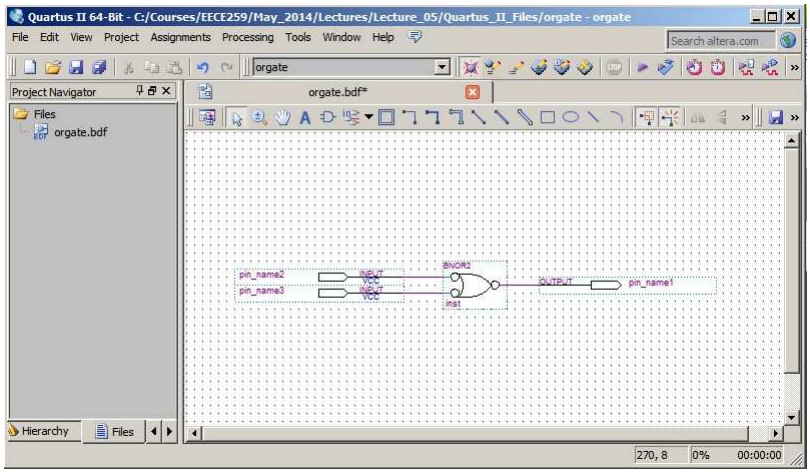
Add the Input Ports



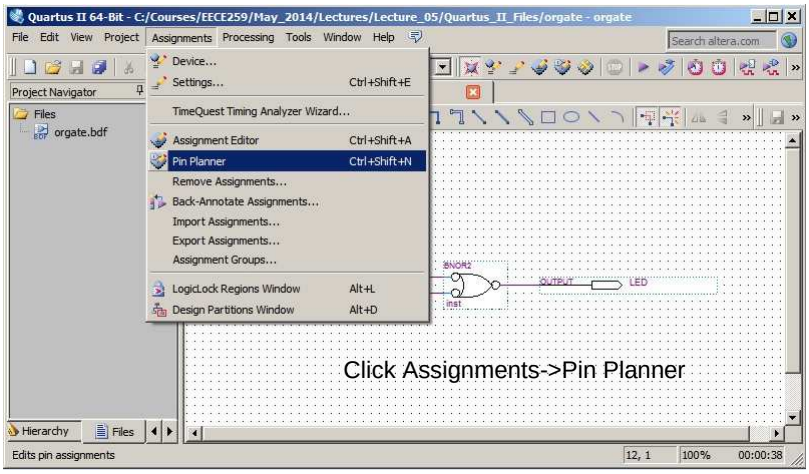
So Far What We Have…



Connect the Ports to the Gate



Rename the ports as PB1, PB2, and LED, then assign them…



Altera DE0-CV Pins

- The Altera DE0-CV has a bunch LEDs, push-buttons, switches, and many other devices available for you to use. They can be connected to your design using the pin assignment window.
- All the pins and their function described in the DE0-CV user manual available at:

http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=921

Altera DE0-CV Pins

Table 3-2 Pin Assignment of Push-buttons

Signal Name	*FPGA Pin No.*	*Description*
KEY0	PIN_U7	Push-button[0]
KEY1	PIN_W9	Push-button[1]
KEY2	PIN_M7	Push-button[2]
KEY3	PIN_M6	Push-button[3]

Table 3-4 Pin Assignment of LEDs

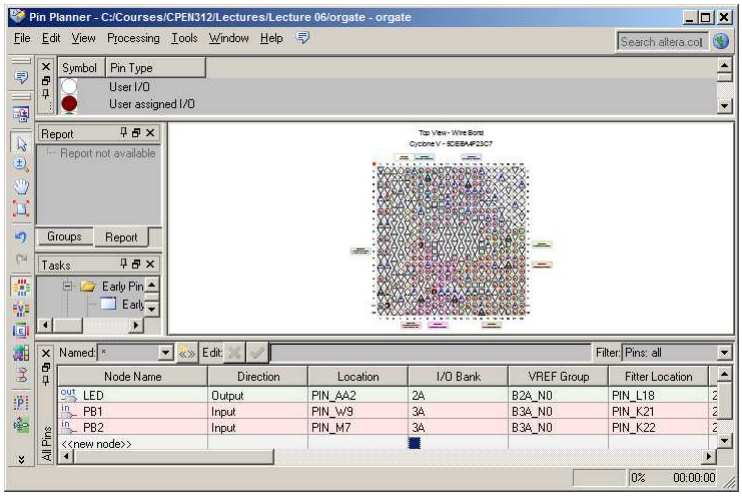
Signal Name	*FPGA Pin No.*	*Description*
LEDR0	PIN_AA2	LED [0]
LEDR1	PIN_AA1	LED [1]

Name	Pin
PB1	PIN_W9
PB2	PIN_M7
LED	PIN_AA2

Assign Pins

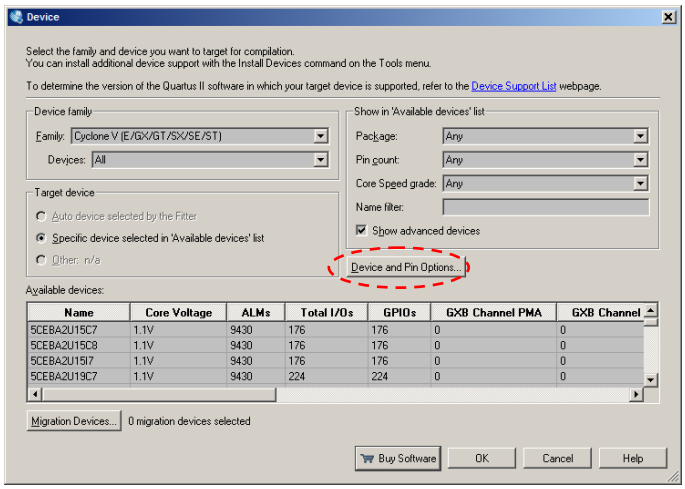
- Save the project.
- Compile the project. Processing->Start Compilation. It takes several minutes. In my old Windows 7 laptop it took 2:27 minutes.
- Click Assignments->Pin Planner

Assign Pins to the Signals



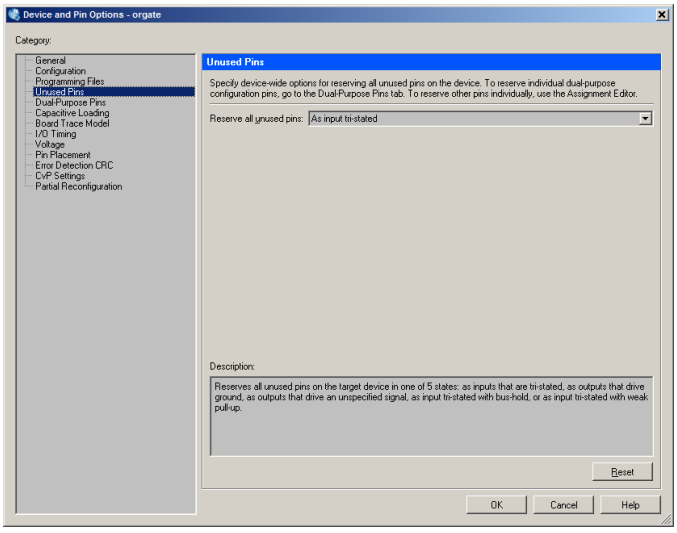
Make all Unused Pins Inputs: Click Assignments->Device…

IMPORTANT!



Make all Unused Pins Inputs

IMPORTANT!



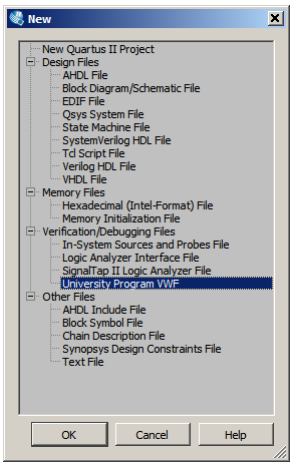
Save and Recompile

- Save.
- Compile the project. Processing->Start Compilation. It may take several minutes…

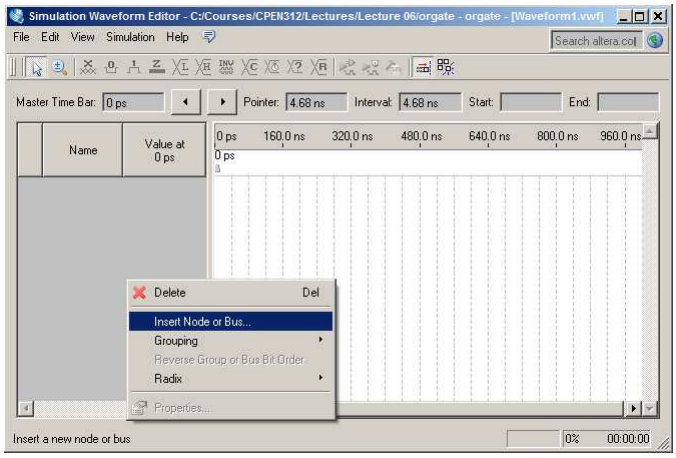
Simulate or Load/Run…

- At this point, we can do two things with the compiled project:
  - Simulate on Quartus Prime. Before we simulate we need an 'University Program Vector Wave File.'
  - Program into the Altera DE0-CV board.

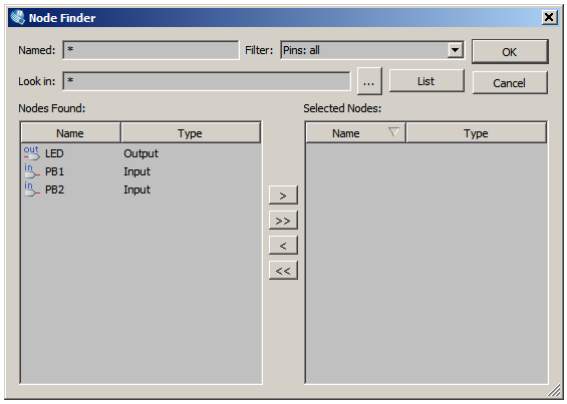
Create University Program VWF



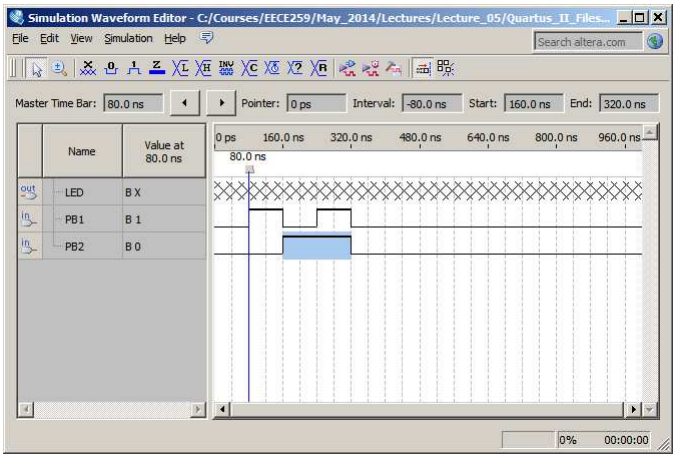
Insert the Signal in out Design



Use ‘Node Finder’ to Quickly Find the Signals.



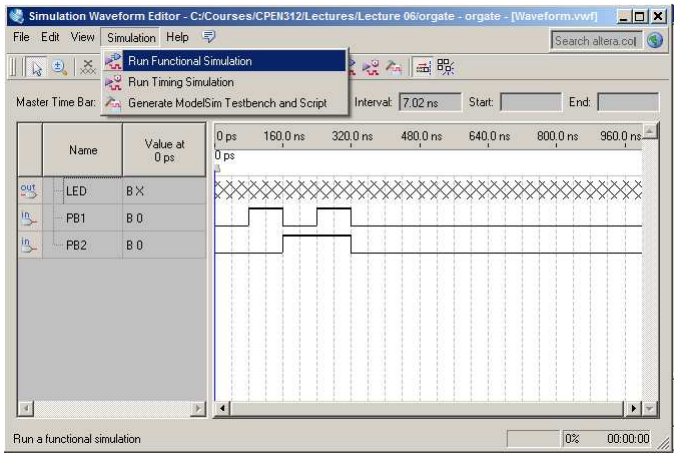
Modify the Input Signals at Your Convenience.



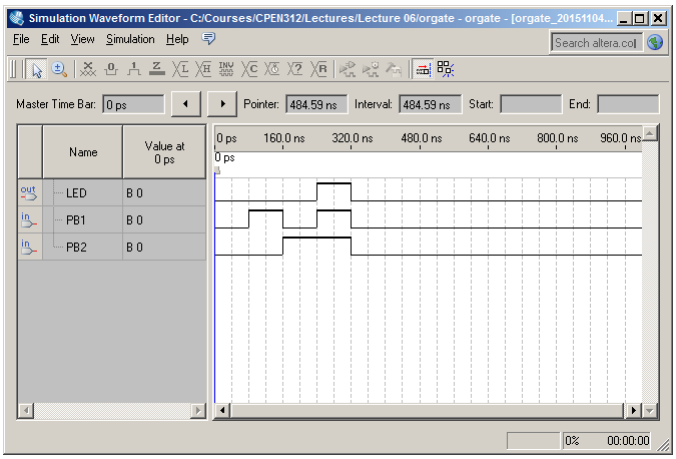
Simulation Options

- Quartus Prime offers us two simulation options:
  - Functional simulation. No gate delays are taken into account.
  - Timing simulation. Gate delays are taken into account.

Run Functional Simulation



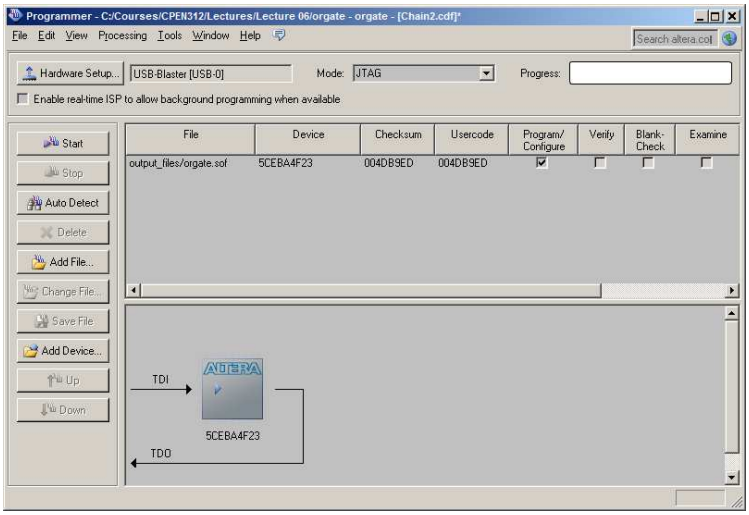
Functional Simulation Output



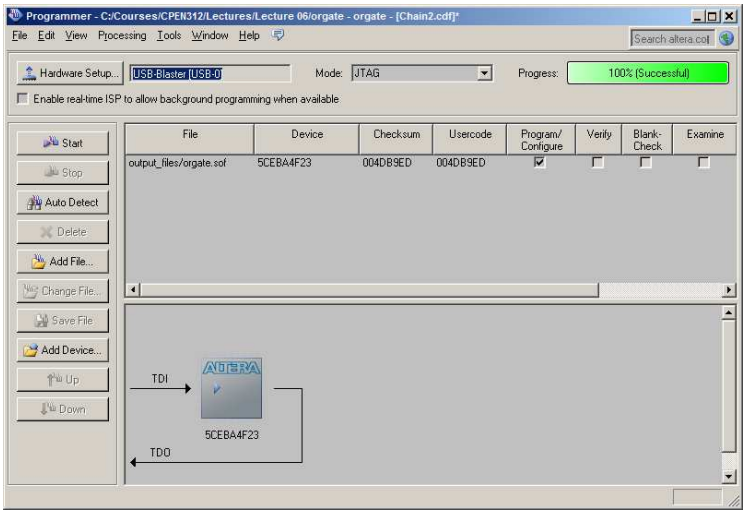
Load design into Altera DE0-CV Board

- Connect the USB cable to the 'USB BLASTER' connector in the DE0-CV board. Connect the other end to your computer.
- If this is the first time you connect the board to your computer, the USB Blaster driver will be installed. If it is not found automatically, it is available with your Quartus Prime installation. In my computer: C:\Altera\16.0\quartus\drivers
- In Quartus Prime click Tools->Programmer. Click "Add File" and look for a file with 'sof' extension. In my computer it was located in the 'output_files' folder.

Program into Altera DE0-CV Board

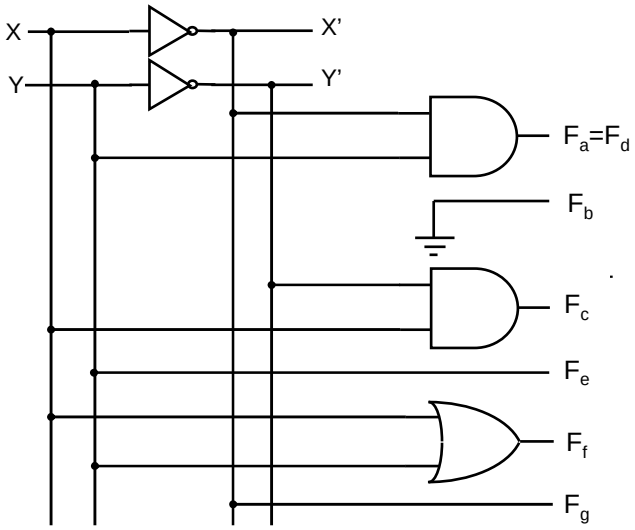


Program into Altera DE0-CV Board



Quartus project available in course's web page as Orgate.zip

Example 1: 2-bit to 7-seg



Example 1: 2-bit to 7-seg

- Connect X and Y to SW0 and SW1. Use HEX0 to display the result.
- The pin assignments for the DE0-CV:

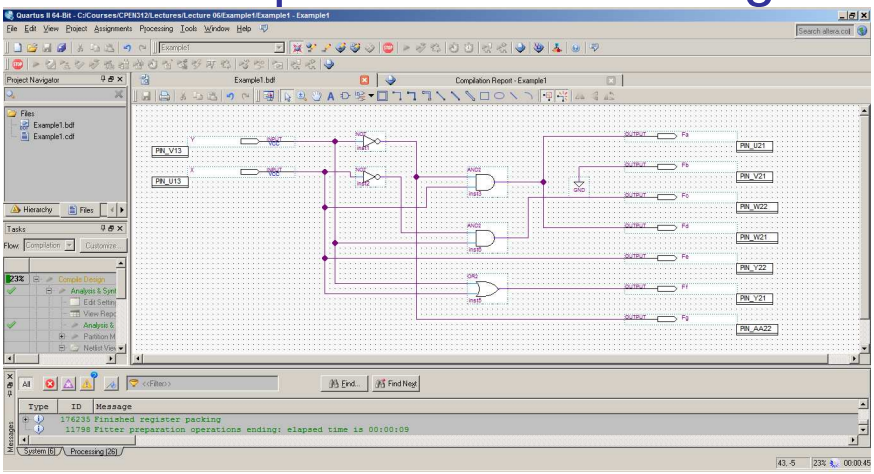
Table 3-3 Pin Assignment of Slide Switches

Signal Name	*FPGA Pin No.*	*Description*
SW0	PIN_U13	Slide Switch[0]
SW1	PIN_V13	Slide Switch[1]
SW2	PIN_T13	Slide Switch[2]

Table 3-5 Pin Assignment of 7-segment Displays

Signal Name	*FPGA Pin No.*	*Description*
HEX00	PIN_U21	Seven Segment Digit 0[0]
HEX01	PIN_V21	Seven Segment Digit 0[1]
HEX02	PIN_W22	Seven Segment Digit 0[2]
HEX03	PIN_W21	Seven Segment Digit 0[3]
HEX04	PIN_Y22	Seven Segment Digit 0[4]
HEX05	PIN_Y21	Seven Segment Digit 0[5]
HEX06	PIN_AA22	Seven Segment Digit 0[6]

Example 1: 2-bit to 7-seg

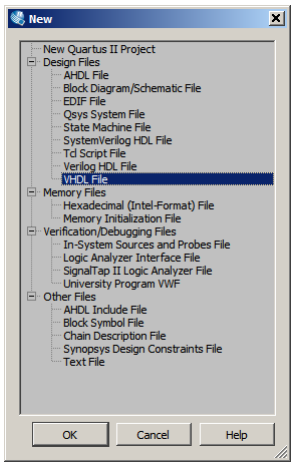


Quartus project available in course's web page as Example1.zip

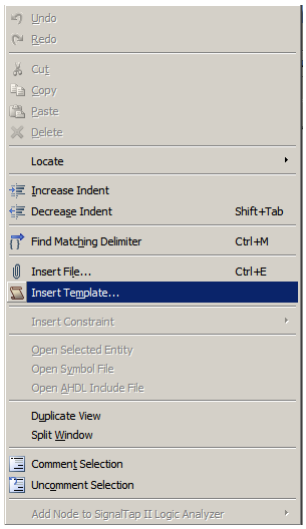
Basic VHDL Circuits

- VHDL is a HARDWARE description language. It is NOT a programming language.
- In VHDL, we create and connect components using statements. Optionally we can create components that can be used in the graphic editor.
- Initially VHDL was created to simulate digital circuits. Latter on it was used to synthesize digital circuits.
- The emphasis of this course is NOT VHDL. Digital logic is!
- As an example, let us create a VHDL circuit that implements the LED=(PB1'+PB2')', the same circuit as from the schematic entry tutorial above:

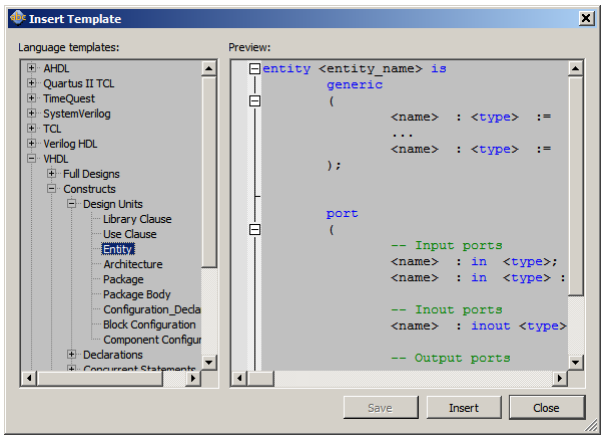
VHDL Entry Tutorial



Insert Template



Entity Declaration



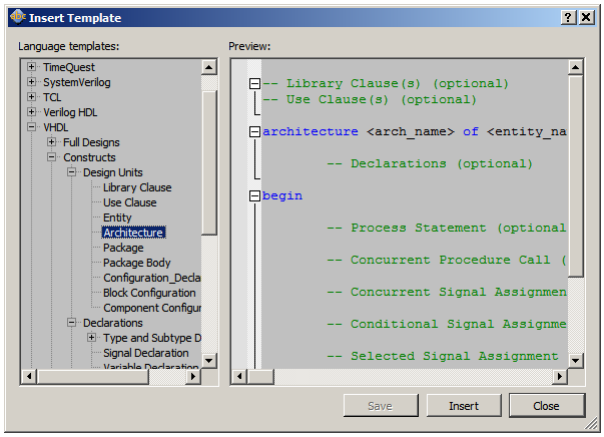
Change Entity Declaration to:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity orvhdl is
    port
    (
        -- Input ports
        PB1, PB2    : in  STD_LOGIC;
        -- Output ports
        LED         : out STD_LOGIC
    );
end orvhdl;
```

The simplest entity declaration just has a name and the name of the inputs and outputs of our circuit. Modify the template entity inserted by Quartus so it looks like the one above.

Architecture Body



Change Architecture to:

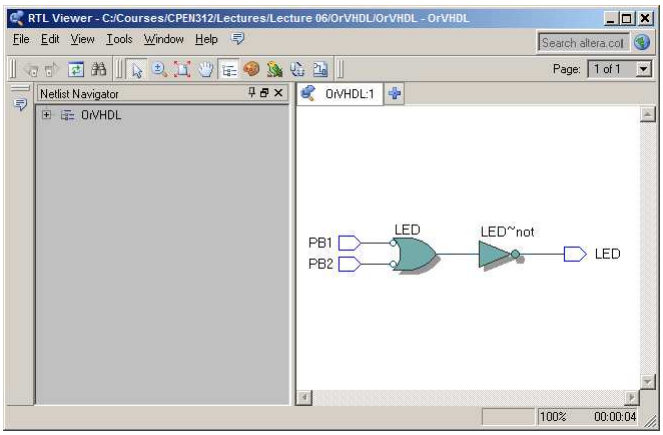
```
architecture mytest of orvhdl is
    -- Declarations (optional)
begin
    LED <= NOT( (NOT PB1) OR (NOT PB2) );
end mytest;
```

Save…

Assign Pins, Compile, Simulate, Load to DE0-CV board…

- Exactly the same we did for the schematic example!
- So, which one is simpler?
  - For smaller designs schematics are ok.
  - For larger designs (hundreds to thousands of gates) schematics are not ok!
- Curious to see how Quartus interpreted our VHDL circuit? Try Tools->Netlist Viewers->RTL Viewer:

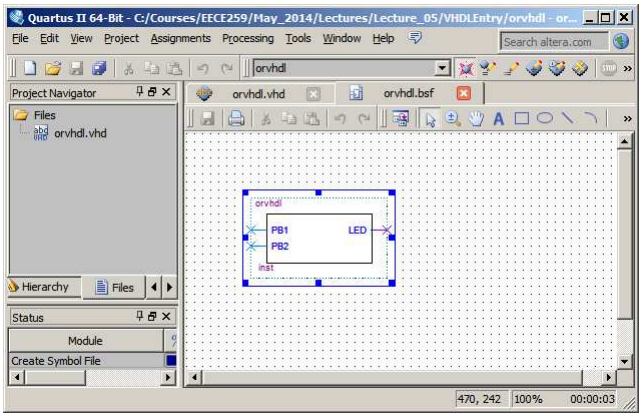
RTL Viewer



Create Schematic Symbol for VHDL Component

- Many complex VHDL projects use only VHDL circuits. No schematics at all.
- Other complex digital circuits combine both VHDL, Verilog, and Schematics.
- It is possible to create a symbol for the VHDL file we just made, and use it in the Schematic Editor!
- File->Create/Update->Create Symbol Files for Current File:

Symbol for VHDL file

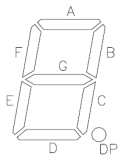


Example 2: VHDL BCD to 7-Segment Decoder

- VHDL has features that allow for easy description of hardware.
- Let us check some of this features by making a BCD to 7-Segment decoder.

Example 2: VHDL BCD to 7-Segment Decoder

Inputs				Outputs						
D	C	B	A	F_a	F_b	F_c	F_d	F_e	F_f	F_g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0



Remember how we did this with Boolean equations last time?
It is way easier in VHDL:

Example 2: VHDL BCD to 7-Segment Decoder

```
library ieee;
use ieee.std_logic_1164.all;

entity BCDto7Seg is
    port
    (
        -- Input ports
        BCD     : in  STD_LOGIC_VECTOR (3 downto 0);
        -- Output ports
        DISPLAY : out STD_LOGIC_VECTOR (0 to 6)
    );
end BCDto7Seg;

architecture a of BCDto7Seg is
    -- Declarations (optional)
    signal bcd_in      : STD_LOGIC_VECTOR (3 downto 0);
    signal display_out : STD_LOGIC_VECTOR (0 to 6);
```

Example 2: VHDL BCD to 7-Segment Decoder

```
begin
        bcd_in  <= BCD;
        DISPLAY <= display_out;

    process( bcd_in) is
    -- Declaration(s)
    begin
    -- Sequential Statement(s)
    case bcd_in is
        when "0000" =>
            display_out<="0000001";
        when "0001" =>
            display_out<="1001111";
        when "0010" =>
            display_out<="0010010";
        when "0011" =>
            display_out<="0000110";
        when "0100" =>
            display_out<="1001100";
```

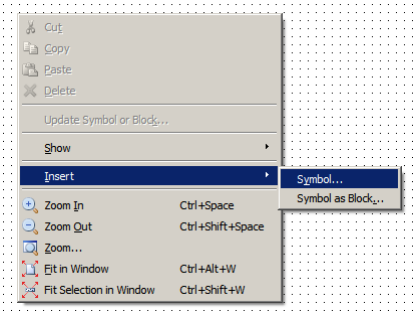

Example 2: VHDL BCD to 7-Segment Decoder

```
            when "0101" =>
                display_out<="0100100";
            when "0110" =>
                display_out<="0100000";
            when "0111" =>
                display_out<="0001111";
            when "1000" =>
                display_out<="0000000";
            when "1001" =>
                display_out<="0000100";
            when others =>
                display_out<="1111111";
        end case;
    end process;
end a;
```

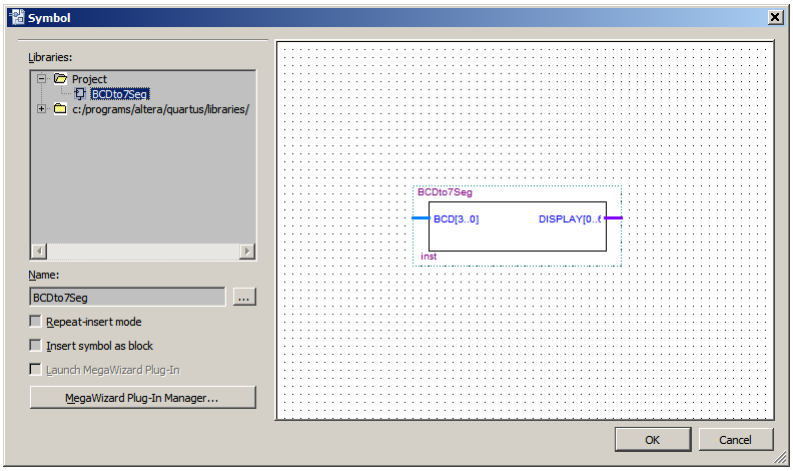
Example 3:

- Create a graphic editor component from example 2, and use it to drive both HEX0 and HEX1:
  - From the previous example File->Create/Update->Create Symbol Files for Current File.
  - Create a new project and copy both BCDto7seg.vhd and BCDto7Seg.bsf from the last example to the new folder. Create a new schematic and right click, then insert symbol:

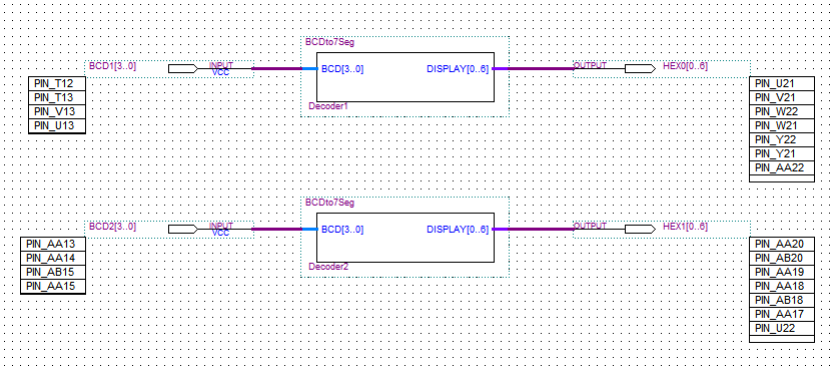
Insert Created Symbol in Quartus Schematic.



Insert Created Symbol in Quartus Schematic.



The New Schematic With the two Decoders.



Here is the complete schematic file with the two BCD to 7Segment decoders (derived from BCDto7Seg.vhd and its symbol).

Exercises

- Modify BCDto7Seg so it converts a Hexadecimal number (0 to F) to its 7-segment representation. Call it HEXto7Seg.vhd. Generate also its symbol file.
- Create a Quartus schematic diagram to read the 10-bit binary number from switches SW0 to SW9 and displays it in hexadecimal using the 7-segment displays HEX0, HEX1, and HEX2. Use the VHDL & symbol files from the previous exercise.