Altera Quartus II Tutorial

Part II

(For ECE 465 Students at UIC)
Sajjad Rahaman
TA for ECE 465, Spring 2009
Department of Electrical and Computer Engineering
University of Illinois at Chicago
mrahaman@ece.uic.edu

The first part of Quartus® II tutorial illustrates schematic diagram based entry for the desired circuit. It becomes very difficult to use this method for a large design with hundreds of primitive gates. Hardware description languages (HDLs) provides standard text based expressions of the structure and behavior of digital circuits. The second part of Quartus® II tutorial is aimed at introducing HDL based design entry method. In this case, VHDL, Verilog or other HDL design files are used to synthesize and simulate the desired design. This tutorial will also introduce two types of simulation, namely, functional simulation and timing simulations, to assess the behavior and performance of the desired design.

Please note that this tutorial is based on Altera Quartus® II 8.1 web edition version.

Content
1. **VHDL design Entry**
2. **Functional Simulation**
3. **Timing Simulation**

1. Creating HDL Design Projects with Quartus II

In this section, a new HDL project containing an 2-to-4 decoder will be designed and compiled with Quartus® II. Design file will be written in VHDL. Please note that VHDL syntax and semantics are beyond the scope of this tutorial. The circuit configuration and VHDL code for an 2-to-4 decoder are shown below
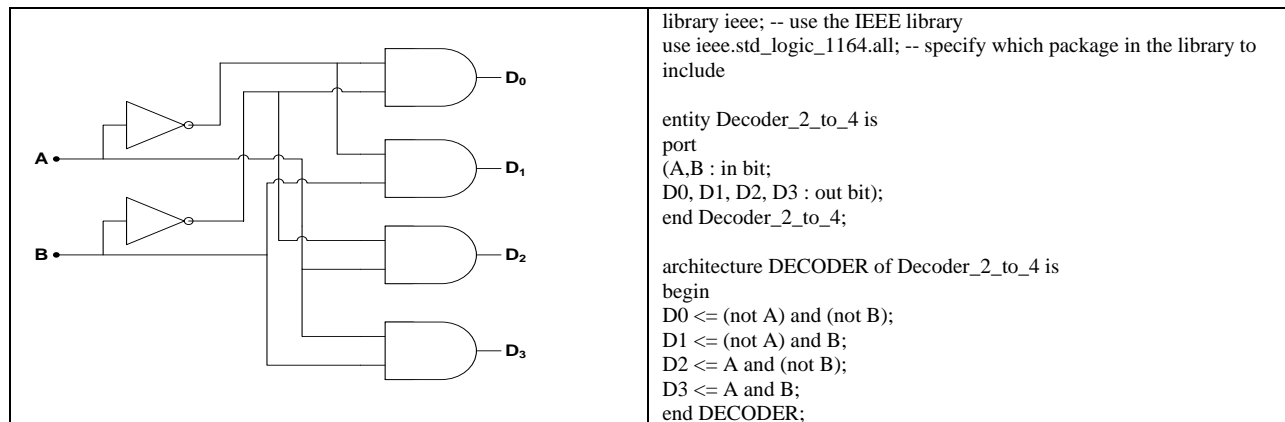


```
library ieee; -- use the IEEE library
use ieee.std_logic_1164.all; -- specify which package in the library to
include

entity Decoder_2_to_4 is
port
(A,B : in bit;
D0, D1, D2, D3 : out bit);
end Decoder_2_to_4;

architecture DECODER of Decoder_2_to_4 is
begin
D0 <= (not A) and (not B);
D1 <= (not A) and B;
D2 <= A and (not B);
D3 <= A and B;
end DECODER;
```
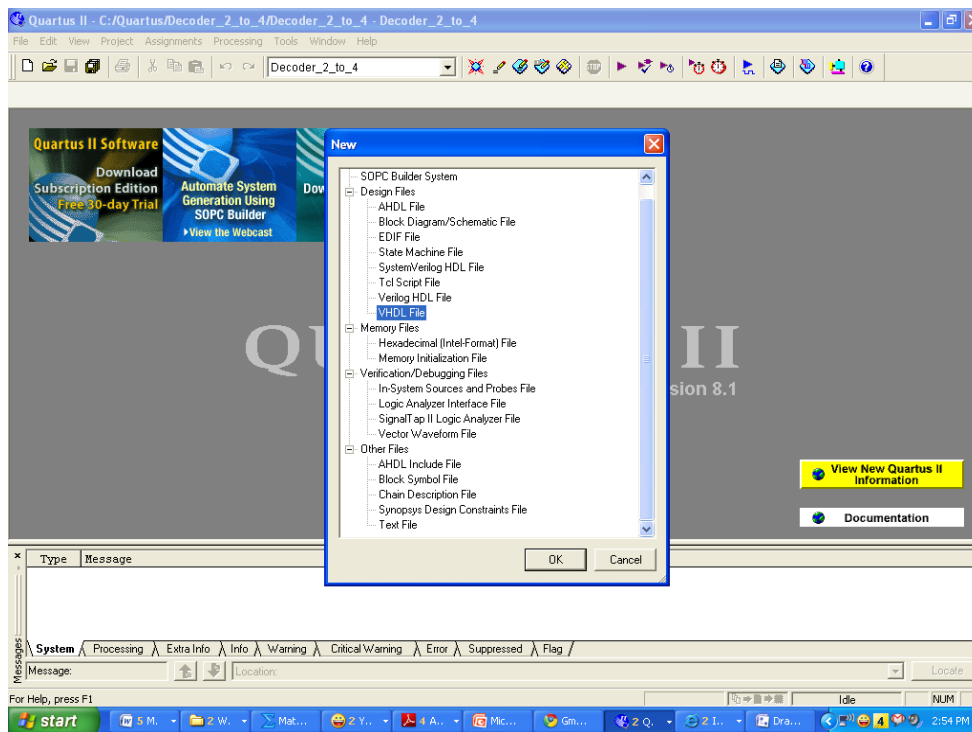
Figure 1. circuit configuration and VHDL code for an 2-to-4 decoder

HDL based design entry in Quartus II follows most of the steps mentioned in first part for schematic based design. After opening a new project using new project wizard we will choose VHDL design file by clicking

File> New>VHDL file

(a)

(b)

Figure 2. Creating a new VHDL based design file

Once we choose VHDL file, Quartus II will open a text editor file **vhdl1.vhd** to put our design file. We will write the VHDL code for 2-to-4 decoder in the window. Please note that VHDL file name has to be the same as the entity name. As shown in the following figure VHDL file has been saved as **Decoder_2_to_4.vhd**.
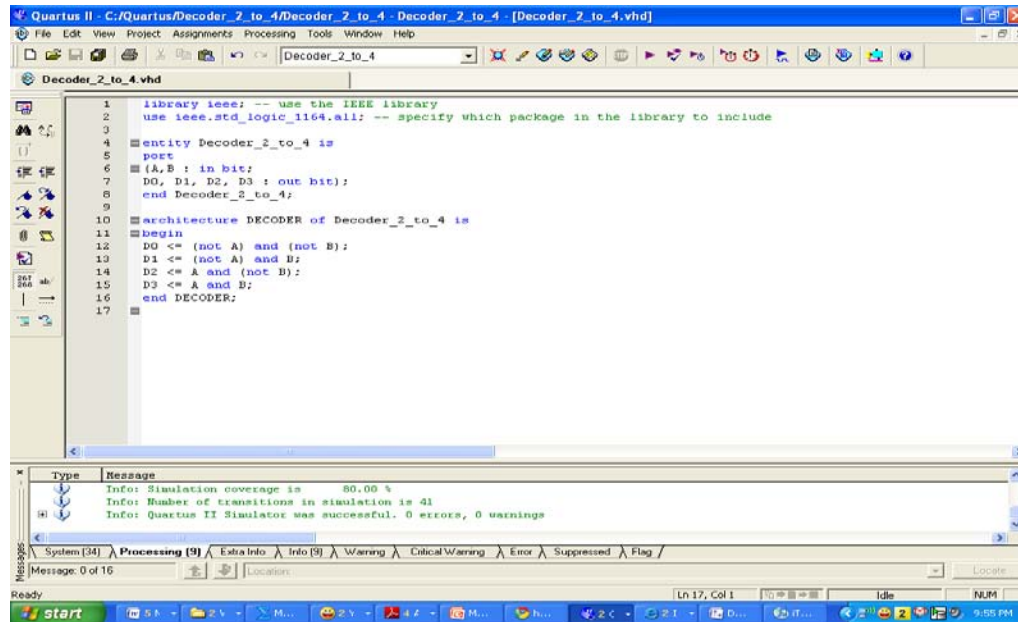
Figure 3. Saving VHDL based design file

Having saved VHDL file, we need to compile the design file for simulation. Compiler will process **Decoder_2_to_4.vhd** file. It is possible to run the full compilation or run individual module (Analysis & Synthesis, the Fitter, and Timing Analyzer). We could run partial compilation by selecting **Start Analysis and Elaboration** command to check **Decoder_2_to_4.vhd** file for syntax and semantics error. The following figures illustrate the command and also the out of the partial compilation.
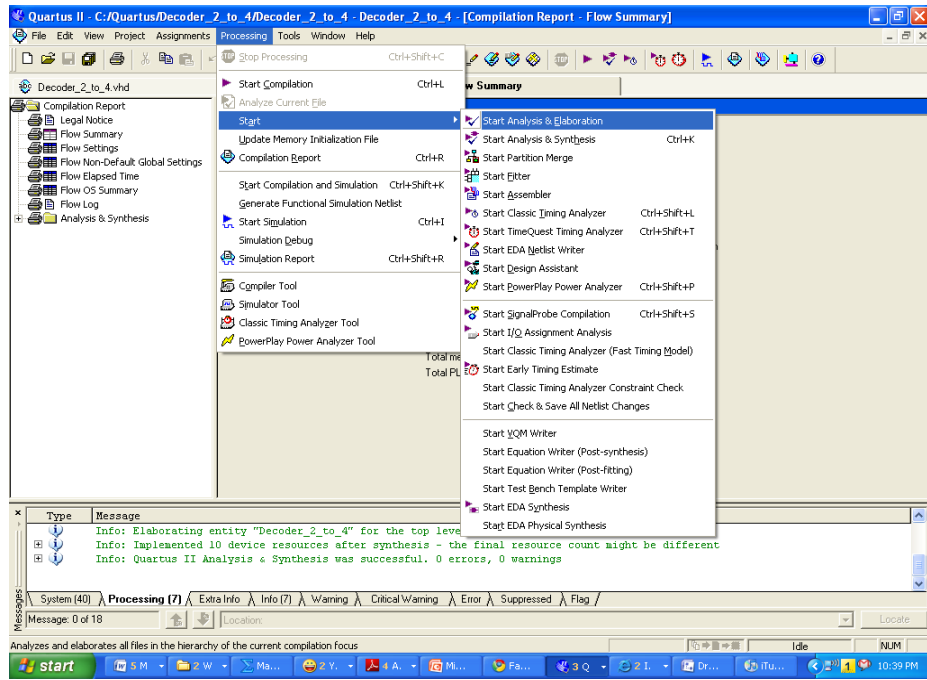
(a)
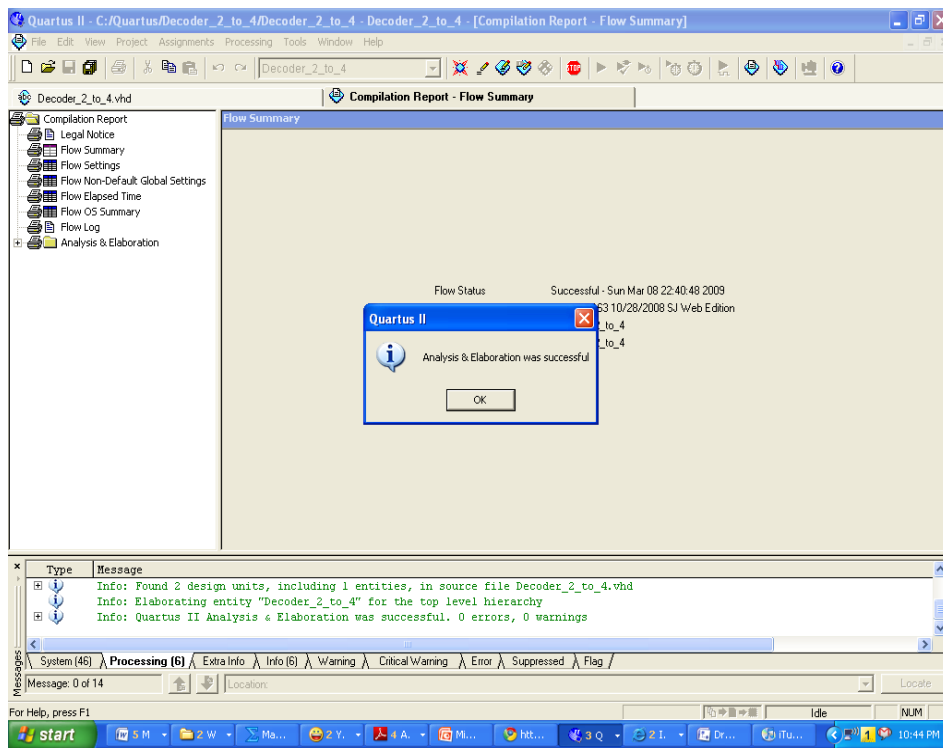
(b)

Figure 4. Start Analysis and Elaboration command

Having successfully compiled our design file, we need a vector waveform file to simulate out design file. Please follow the steps shown in part I of this tutorial.
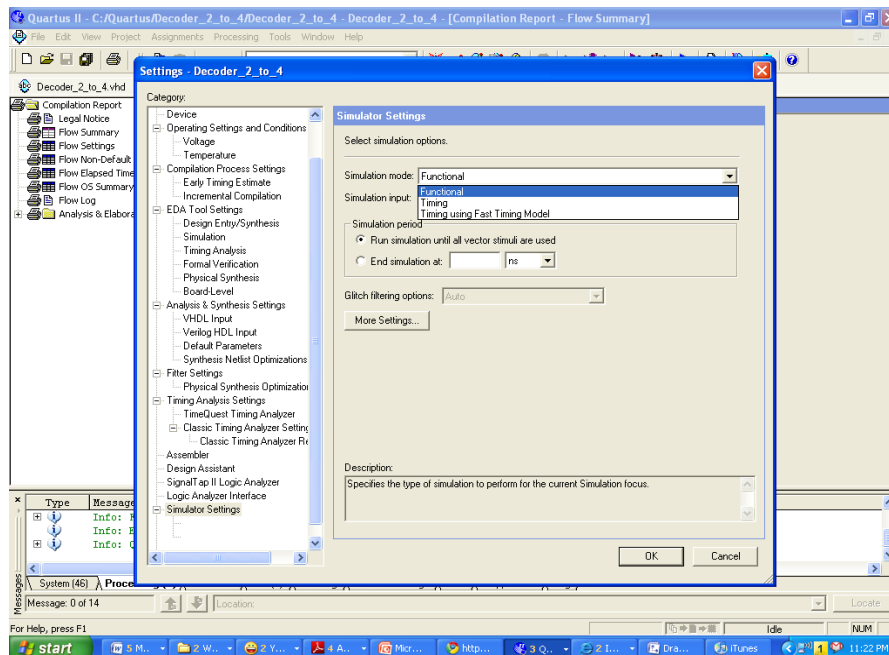
2.   Functional Simulation

Functional simulation verifies the correctness of logic operation of the synthesized circuits. It does not take timing issue into consideration.

Quartus II carries out timing simulation by default. So, the setting needs to be changed to run functional simulation. This is done by selecting **Assignment>Setting** and then selecting functional simulation mode. Figure 5 illustrates the changes necessary in default setting to run function simulation.
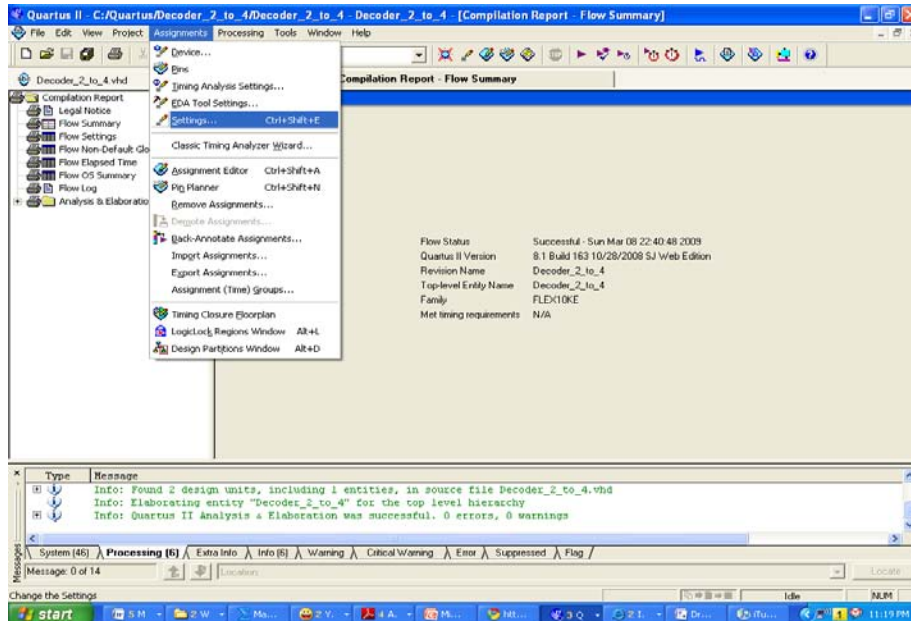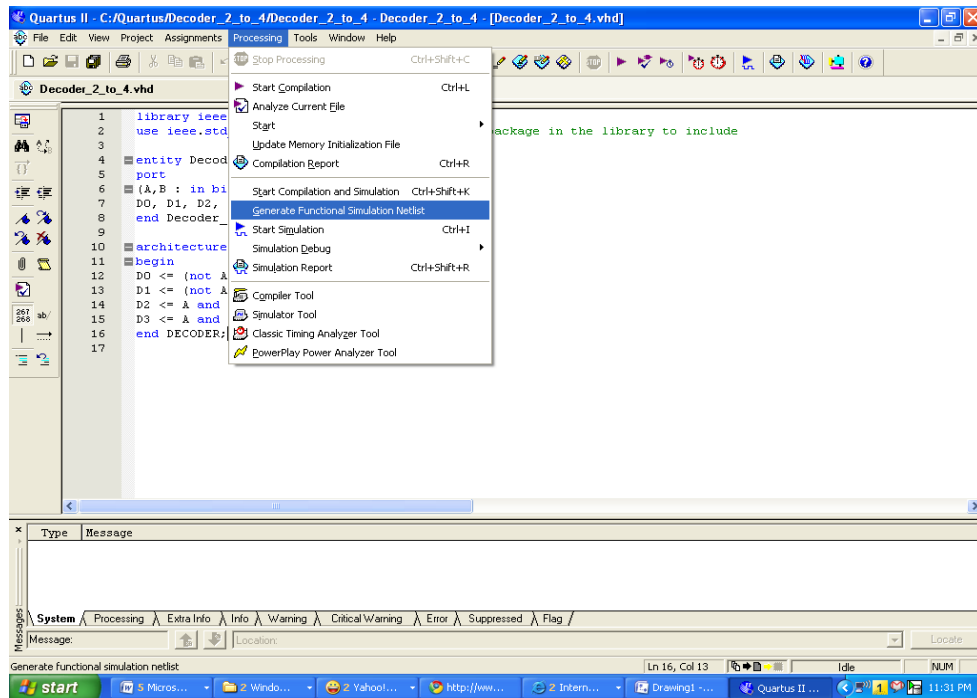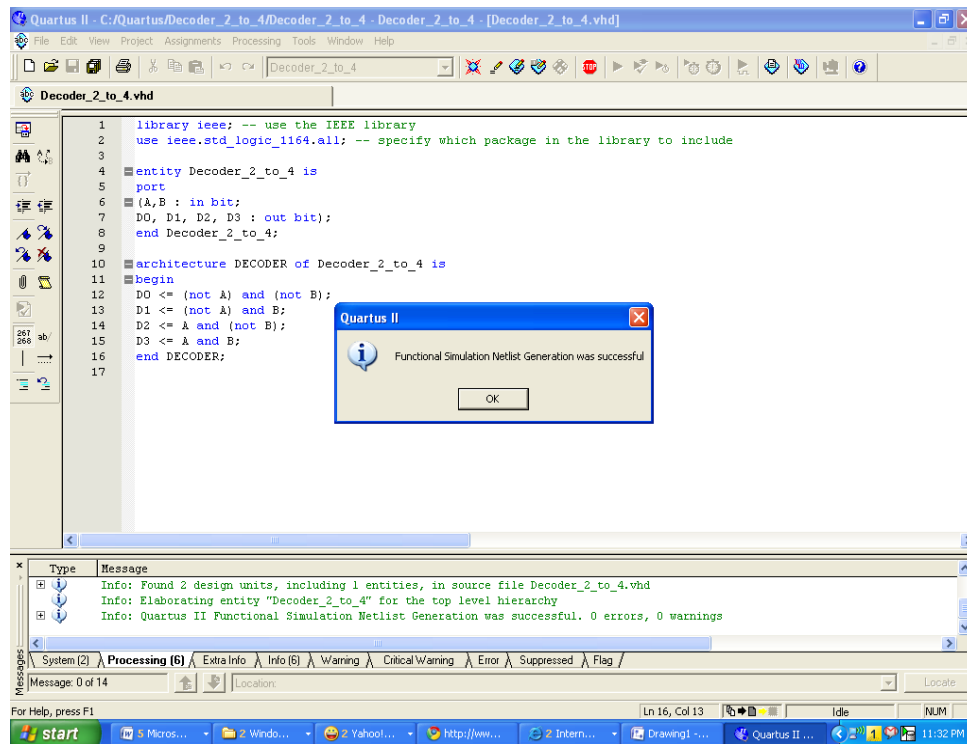

(a)

(b)

Figure 5. Setting for functional simulation

Before running functional simulation, a functional simulation netlist for the synthesized circuit needs to be generated. This is done by selecting **Processing > Generate functional Simulation Netlist**. Figure 6 shows the screen capture of the command and output.
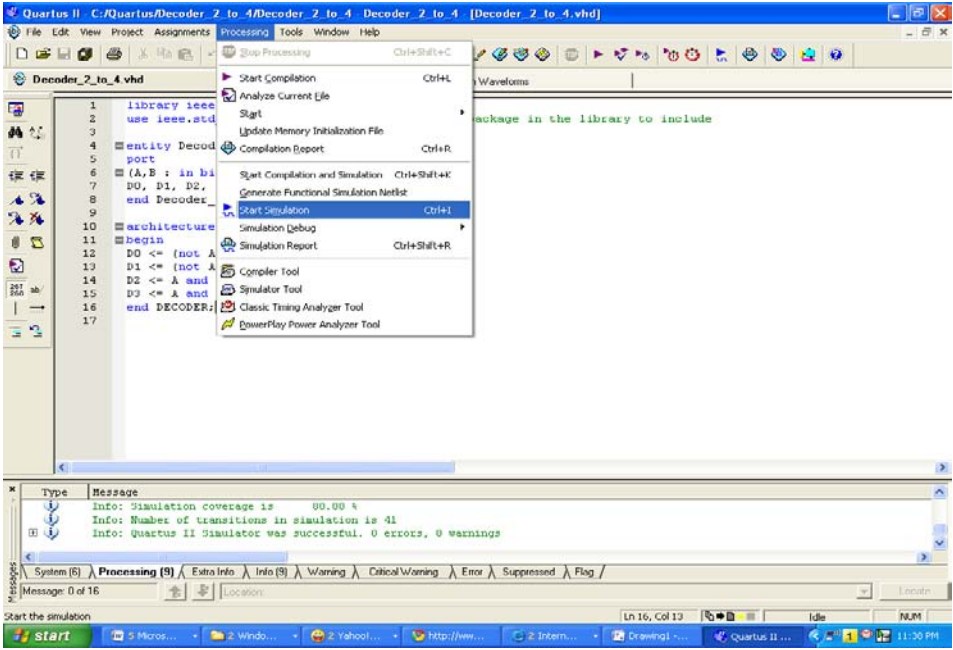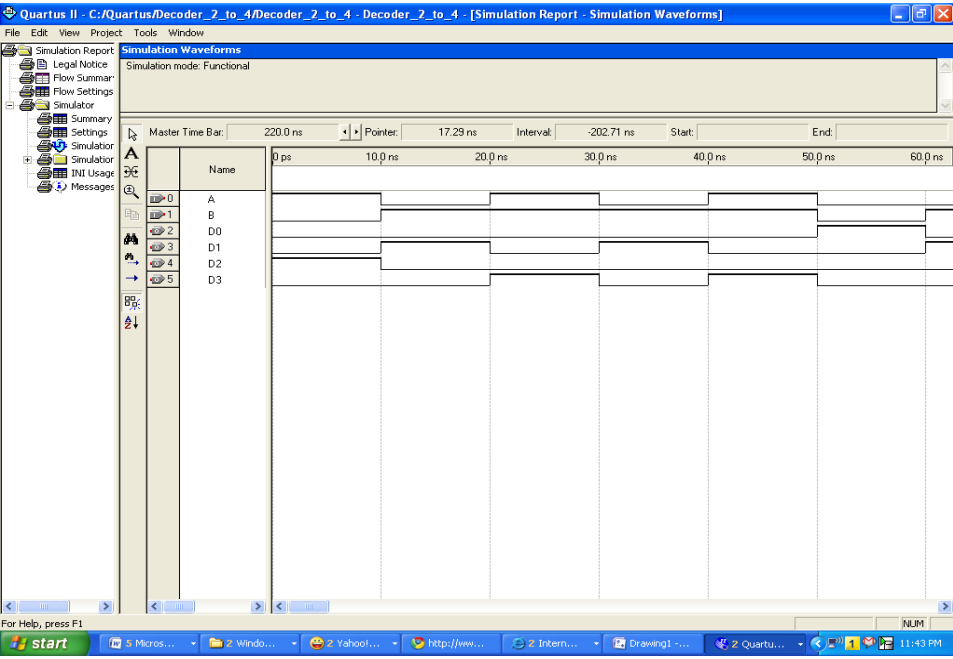

(a)

(b)

Figure 6. Functional simulation netlist generation

At this point, we are ready to run functional simulation. Simulation will use a vector waveform file created earlier. We will select **Process > Start Simulation** to begin the simulation. Successful simulation will generate simulation waveform (Figure 7)

(a)

(b)

Figure 7. Simulation output

As shown in the above figure, our design file correctly generates the output variables. It is also seen that there is no time delay between input and output switching.

3.   Timing Simulation

Having verified the logical correctness of the design file, it is necessary to determine whether the synthesized circuit meets the delay constraint. Timing simulation verifies both the logical correctness and timing. We will be able to find the propagation delay along various paths in the synthesized circuits.

We have changed the default setting to run the functional simulation. So, we have to change it back to Timing simulation mode to run timing simulation. Timing simulation is carried out by selecting **Processing > Start Compilation and Simulation**. After successful compilation and simulation, simulation waveform will be diplayed. Figure 8 shows the simulation waveform from timing analysis.
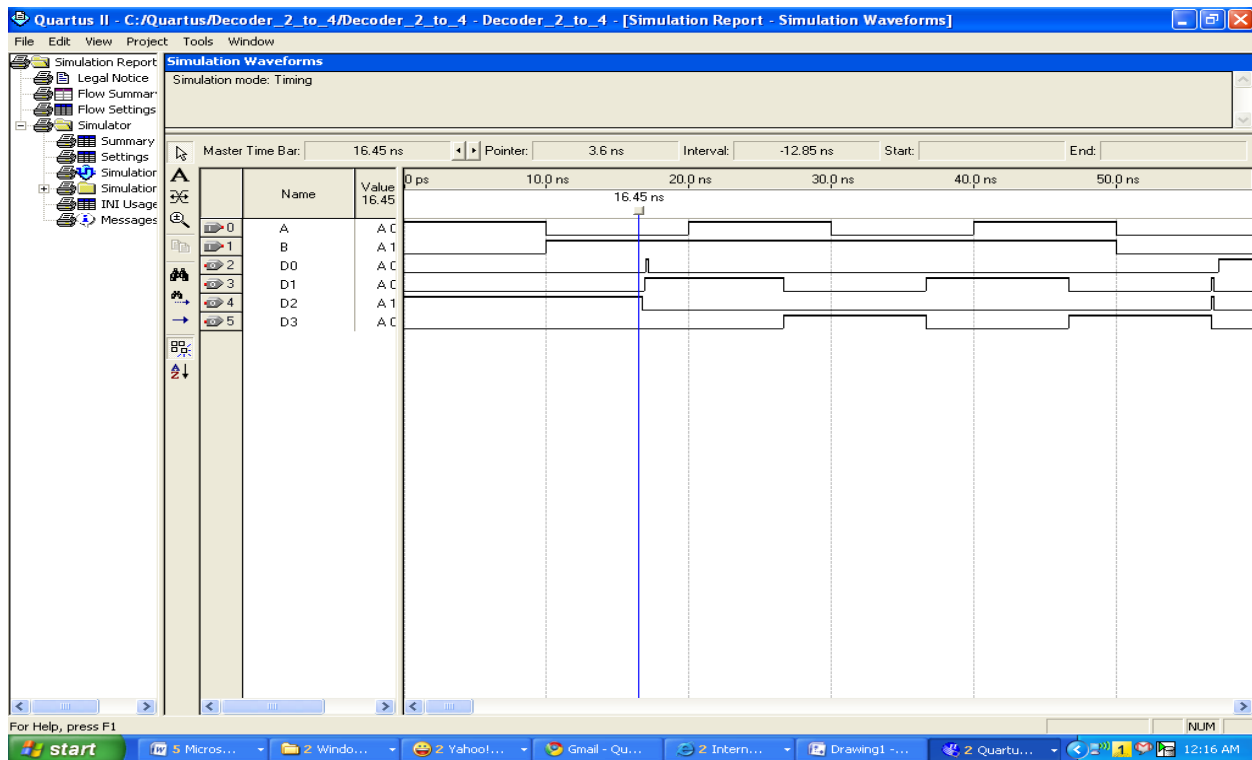

Figure 8. Timing simulation output

It is seen in Figure 8 that there are propagation delays between inputs and outputs. Timing Analyzer tool will provide propagation delays along all the paths and the worst case propagation delay. Figure 9 illustrates the timing parameter of our design file. Please note that this delay depends on the device chosen for the simulation. This tutorial uses FLEX10KE for simulation purpose.
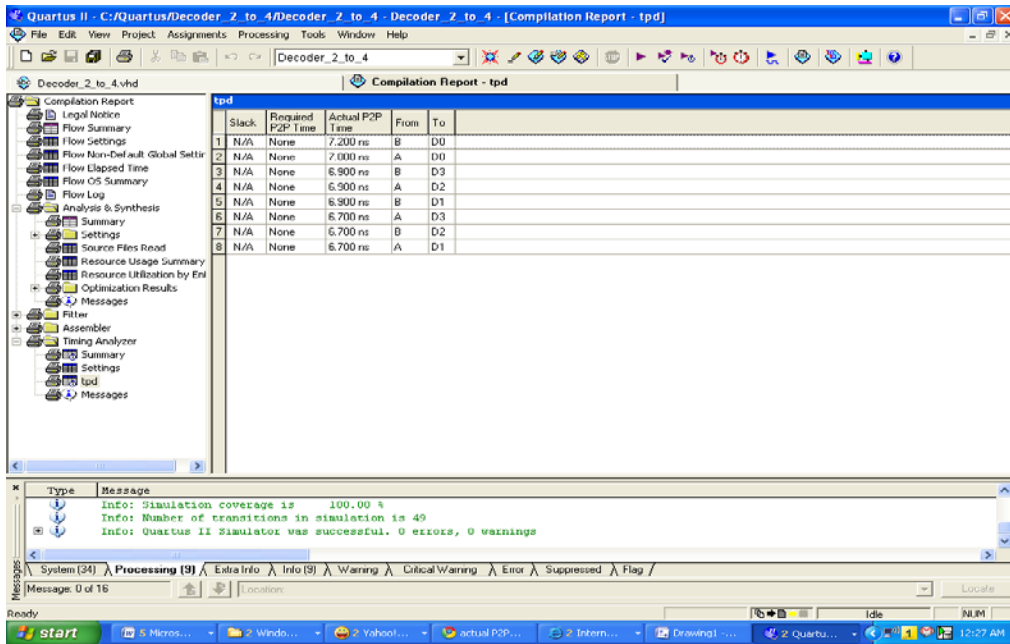
(a)
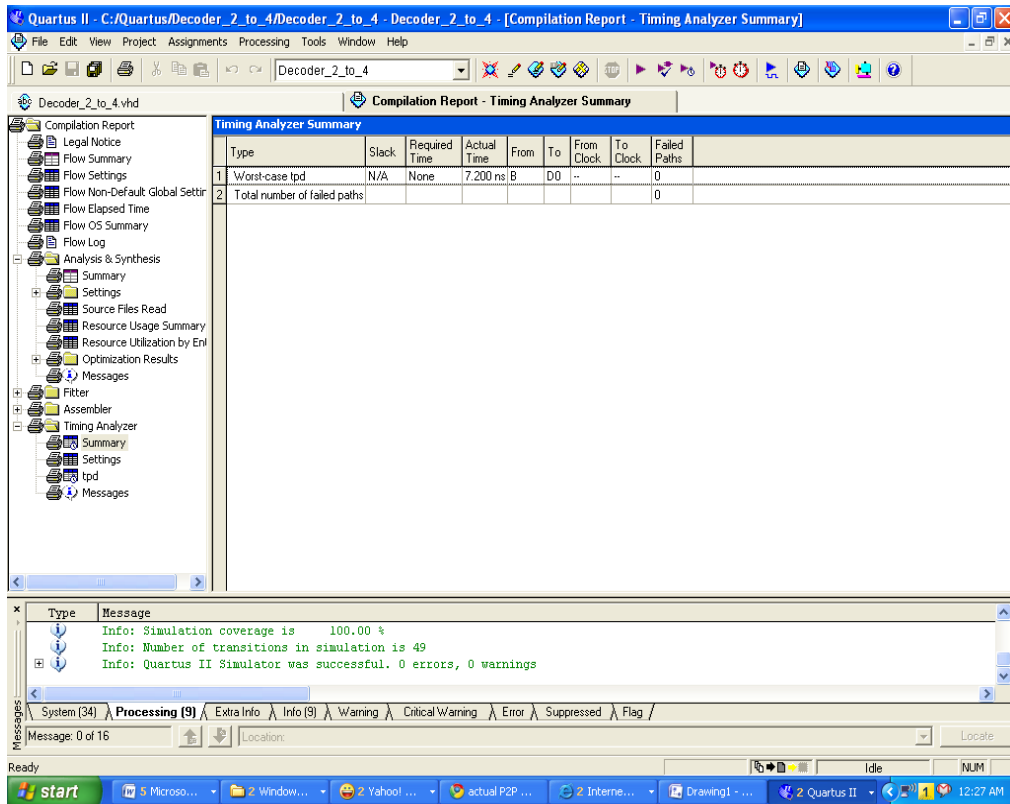
(b)

Figure 9. Timing analyzer output