Experiment 28 - Digital Electronics with Altera Supporting Material

Department of Electrical Engineering & Electronics

April 2022, Ver. 8.1

1. Introduction

This material will take you through programming an Altera® DE1 Development and Education board (Figure 1). Featuring an Altera Cyclone® II 2C20 FPGA, the DE1 board is designed for university and college laboratory use. It is suitable for a wide range of exercises in courses on digital logic and computer organization, from simple tasks that illustrate fundamental concepts to advanced designs.

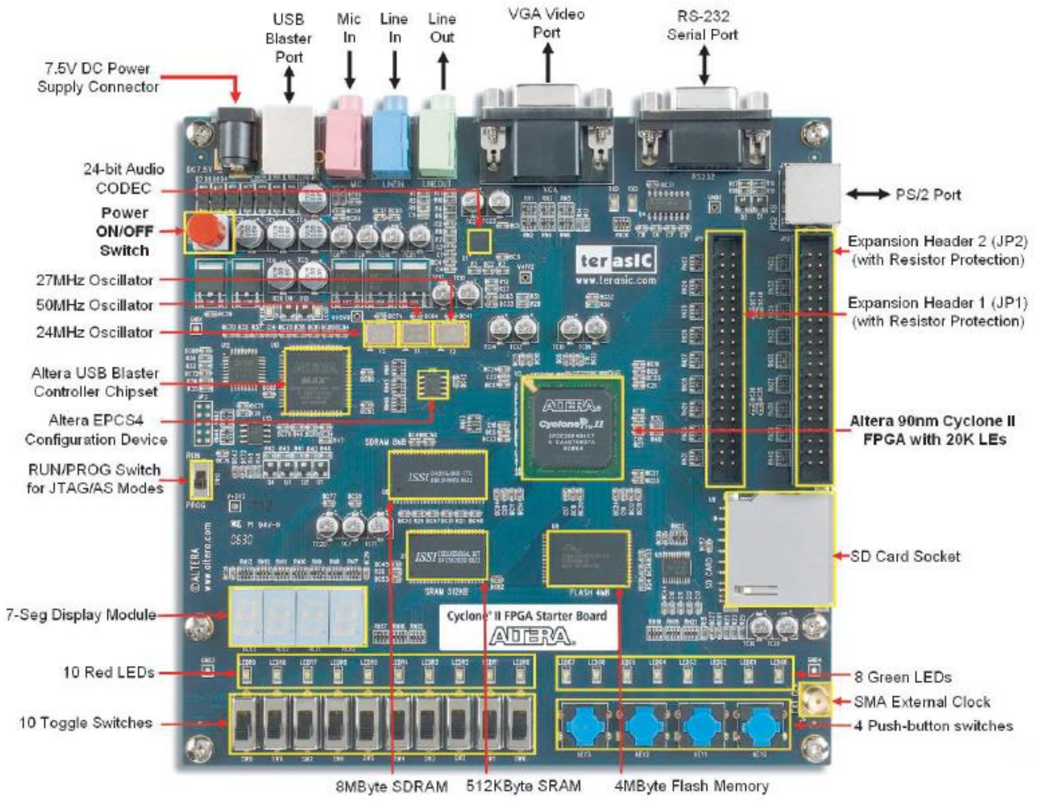


Figure 1 Altera DE1 Board

The following hardware is provided on a DE1 board:

- Altera Cyclone® II 2C20 FPGA device
- Altera Serial Configuration device – EPCS4
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 512-Kbyte SRAM
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory

- SD Card socket
- 4 pushbutton switches
- 10 toggle switches
- 10 red user LEDs
- 8 green user LEDs
- 50-MHz oscillator, 27-MHz oscillator and 24-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (4-bit resistor network) with VGA-out connector
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- Two 40-pin Expansion Headers with resistor protection
- Powered by either a 7.5V DC adapter or a USB cable

In addition to these hardware features, the DE1 board has software support for standard I/O interfaces and a control panel facility for accessing various components. Also, software is provided for a number of demonstrations that illustrate the advanced capabilities of the DE1 board.

In order to use the DE1 board, the user has to be familiar with the Quartus II software, which this assignment teaches you to use. Useful knowledge on both can also be acquired by reading the following Altera tutorials located on Canvas: *Getting Started with Altera's DE1 Board*, *Quartus II Introduction Using Schematic Design* and *Quartus II Introduction Using Verilog Design*. So if you are interested, or you are finding any guidance in this document and the PreLab / main lab script confusing and just want another perspective, why not take a look?

2. DE1 Board Components (Hardware)

For the DE1 board which this design assignment is aimed at, the outputs to the four seven- segment displays and a number of LEDs and the inputs from the push buttons and toggle switches are hard-wired to specific pins of the FPGA.

2.1. Seven Segment Displays

The development board provides four adjacent 7-segment displays, **HEX0–HEX3** (Figure 2**)**, for displaying numerical values from the FPGA. Each segment connects to an FPGA general-purpose I/O pin. A LOW logic level applied at the pin lights up the segment; a HIGH logic level turns the segment off.

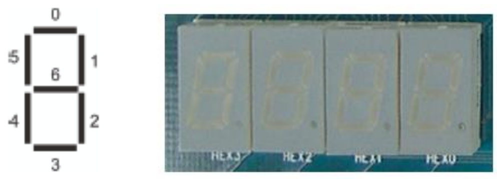


Figure 2. 7-Segment Displays on the DE1 board

An index from 0 to 6 identifies each segment and its position. The DE1 board does not connect or use the dot in the display. Table 1 shows the FPGA pin allocations for the 7 segment displays.

Table 17 Segment pin allocations

Signal Name	FPGA Pin	Description
HEX0[0]	PIN_J2	Seven-Segment segment 0[0]
HEX0[1]	PIN_J1	Seven-Segment segment 0[1]
HEX0[2]	PIN_H2	Seven-Segment segment 0[2]
HEX0[3]	PIN_H1	Seven-Segment segment 0[3]
HEX0[4]	PIN_F2	Seven-Segment segment 0[4]
HEX0[5]	PIN_F1	Seven-Segment segment 0[5]
HEX0[6]	PIN_E2	Seven-Segment segment 0[6]
HEX1[0]	PIN_E1	Seven-Segment segment 1[0]
HEX1[1]	PIN_H6	Seven-Segment segment 1[1]
HEX1[2]	PIN_H5	Seven-Segment segment 1[2]
HEX1[3]	PIN_H4	Seven-Segment segment 1[3]
HEX1[4]	PIN_G3	Seven-Segment segment 1[4]
HEX1[5]	PIN_D2	Seven-Segment segment 1[5]
HEX1[6]	PIN_D1	Seven-Segment segment 1[6]
HEX2[0]	PIN_G5	Seven-Segment segment 2[0]
HEX2[1]	PIN_G6	Seven-Segment segment 2[1]
HEX2[2]	PIN_C2	Seven-Segment segment 2[2]
HEX2[3]	PIN_C1	Seven-Segment segment 2[3]
HEX2[4]	PIN_E3	Seven-Segment segment 2[4]
HEX2[5]	PIN_E4	Seven-Segment segment 2[5]
HEX2[6]	PIN_D3	Seven-Segment segment 2[6]
HEX3[0]	PIN_F4	Seven-Segment segment 3[0]
HEX3[1]	PIN_D5	Seven-Segment segment 3[1]
HEX3[2]	PIN_D6	Seven-Segment segment 3[2]
HEX3[3]	PIN_J4	Seven-Segment segment 3[3]
HEX3[4]	PIN_L8	Seven-Segment segment 3[4]
HEX3[5]	PIN_F3	Seven-Segment segment 3[5]
HEX3[6]	PIN_D4	Seven-Segment segment 3[6]

2.2. Toggle Switches

The board has 10 toggle switches, SW0 to SW9. In the DOWN or OFF position (closest to the edge of the board), a switch provides a LOW logic level (0 volts) to the FPGA. In the UP position a switch provides a HIGH logic level (3.3 volts). Figure 3 shows the toggle switches.

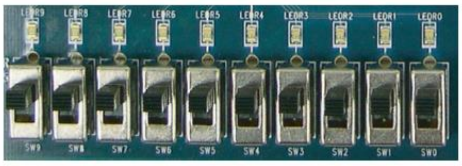


Figure 3 Toggle Switches

Table 2 shows the FPGA pin allocations for the toggle switches.

Table 2 FPGA pin allocations for the toggle switches

Switch	FPGA Pin	Description
SW[0]	PIN_L22	Toggle Switch[0]
SW[1]	PIN_L21	Toggle Switch[1]
SW[2]	PIN_M22	Toggle Switch[2]
SW[3]	PIN_V12	Toggle Switch[3]
SW[4]	PIN_W12	Toggle Switch[4]
SW[5]	PIN_U12	Toggle Switch[5]
SW[6]	PIN_U11	Toggle Switch[6]
SW[7]	PIN_M2	Toggle Switch[7]
SW[8]	PIN_M1	Toggle Switch[8]
SW[9]	PIN_L2	Toggle Switch[9]

2.3. Push button Switches

The DE1 board has four push button switches (Figure 4), **KEY0-KEY3**, located at the bottom right on the DE1 board below the green LEDs, **LEDG0-LEDG7.** The momentary-contact switches provide stimulus to designs in the FPGA.

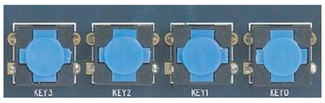


Figure 4 Push button switches

A switch generates an active-LOW pulse at 0 volts when pressed, returning to a HIGH logic level at 3.3 volts when released. A Schmitt Trigger circuit on each switch debounces the signal (Figure 5).

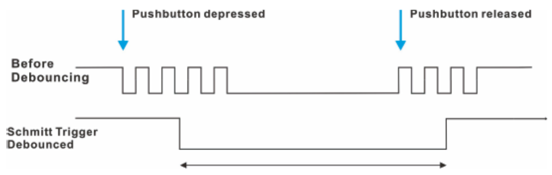


Figure 5 Switch signal debounce

Table 3 shows the FPGA pin allocations for the push buttons.

Table 3 Pushbutton pin allocations

Switch	FPGA Pin	Description
KEY[0]	PIN_R22	Pushbutton[0]
KEY[1]	PIN_R21	Pushbutton[1]
KEY[2]	PIN_T22	Pushbutton[2]
KEY[3]	PIN_T21	Pushbutton[3]

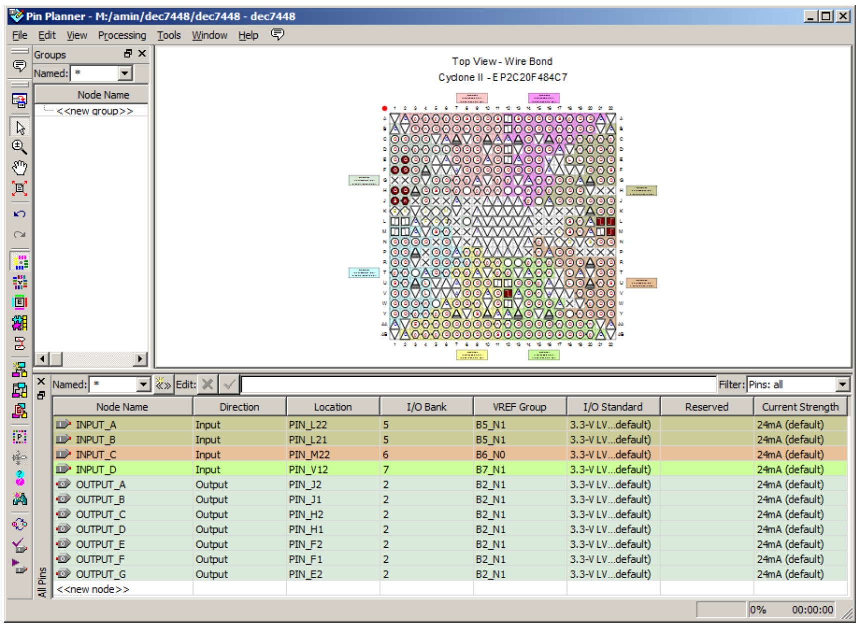
We therefore need to assign the output pins as indicated.

3. Assign pins, Compilation and programming the DE1 board

3.1. Assign Pins

To use the DE1 board components (hardware), it is important to assign the required pins. Imagine that in your design you have an input of on or off (1/0). This easily can be used with the pushbutton key on the DE1 board as your input key. In this case, from Table 3 you choose your key switch (e.g. KEY[1]) and its pin number (PIN_R22), to assign it on the DE1 board. This helps you to apply your designed project on the DE1 board and use it, based on the assignments in the **Pin planner**.

In Quartus II, you should always compile your design before trying to assign pins. Then, click on the **Assignments** menu and then click on the **Pin planner** to open the **Pin Planner** window (Figure 6). Assign pins to the inputs and outputs by double clicking on the location drop-down box and selecting the pins which are physically connected to the Seven Segment and the input Toggle switches of the board.


Pin Planner - M:/amin/dec7448/dec7448 - dec7448
File Edit View Processing Tools Window Help
Groups
Named: *
Node Name
<<new group>>
Top View - Wire Bond
Cyclone II - EP2C20F484C7
Named: *
Edit:
Filter: Pins: all
All Pins
0%
00:00:00

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength
INPUT_A	Input	PIN_L22	5	B5_N1	3.3-V LV...default)		24mA (default)
INPUT_B	Input	PIN_L21	5	B5_N1	3.3-V LV...default)		24mA (default)
INPUT_C	Input	PIN_M22	6	B6_N0	3.3-V LV...default)		24mA (default)
INPUT_D	Input	PIN_V12	7	B7_N1	3.3-V LV...default)		24mA (default)
OUTPUT_A	Output	PIN_J2	2	B2_N1	3.3-V LV...default)		24mA (default)
OUTPUT_B	Output	PIN_J1	2	B2_N1	3.3-V LV...default)		24mA (default)
OUTPUT_C	Output	PIN_H2	2	B2_N1	3.3-V LV...default)		24mA (default)
OUTPUT_D	Output	PIN_H1	2	B2_N1	3.3-V LV...default)		24mA (default)
OUTPUT_E	Output	PIN_F2	2	B2_N1	3.3-V LV...default)		24mA (default)
OUTPUT_F	Output	PIN_F1	2	B2_N1	3.3-V LV...default)		24mA (default)
OUTPUT_G	Output	PIN_E2	2	B2_N1	3.3-V LV...default)		24mA (default)
<<new node>>							

Figure 6 Pin Planner

Figure 7 shows an example of a schematic with assigned pins.

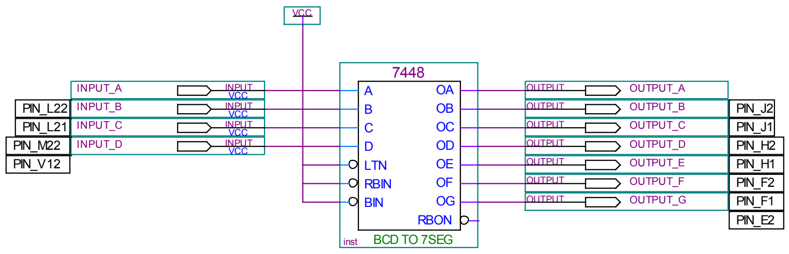


Figure 7 Schematic with pin allocations

We have made the pin assignments, but they won't take effect until we compile again.

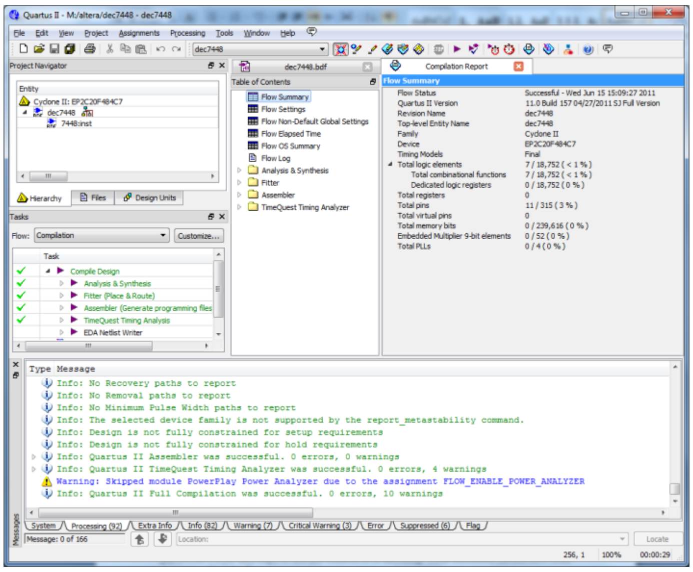
3.2. Compilation (again)

In Quartus II, click on the **Start Compilation** button to compile. We are now ready to program our PLD. As a general rule, you should re-simulate after changing pin assignments, but as pin assignments should only affect timing, and timing is not a concern for this project, we won't check the simulation again here. If we were concerned with the speed and timing of our design, and you almost always will be in practice, we would most certainly have to re-simulate at this point.

If you did everything correctly, you won't see any errors, but in developing actual projects, you will probably see some errors and lots of warnings. An error is a problem so serious that the compilation can't continue. You will have to fix the error before you can compile. A warning won't usually stop the compilation, but you need to look at the warning message to see if it something you need to fix before proceeding. Some warnings can be ignored, while others will need to be fixed. Look at the help messages and use your own judgment and experience to determine which warnings you need to heed.

The error messages are generally fairly detailed and clear and you can usually determine what they mean just by reading them. You can also use **Help → Message List** to get more information on the error.

The Compilation Report (Figure 8) shows how many pins your design used, and also how many Logic Cells (LCs) your design used.



Flow Summary

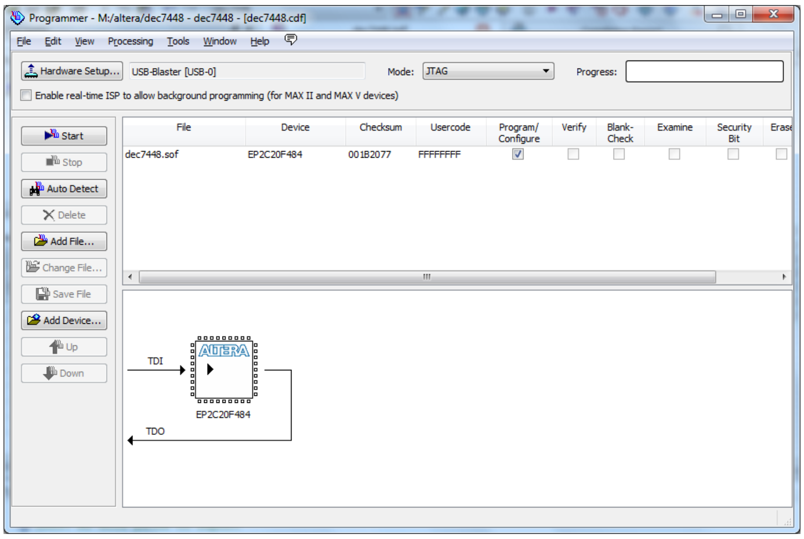
Flow Status	Successful - Wed Jun 15 15:09:27 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Full Version
Revision Name	dec7448
Top-level Entity Name	dec7448
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	7 / 18,752 (< 1 %)
Total combinational functions	7 / 18,752 (< 1 %)
Dedicated logic registers	0 / 18,752 (0 %)
Total registers	0
Total pins	11 / 315 (3 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

```
Type Message
    Info: No Recovery paths to report
    Info: No Removal paths to report
    Info: No Minimum Pulse Width paths to report
    Info: The selected device family is not supported by the report_metastability command.
    Info: Design is not fully constrained for setup requirements
    Info: Design is not fully constrained for hold requirements
    Info: Quartus II Assembler was successful. 0 errors, 0 warnings
    Info: Quartus II TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
    Warning: Skipped module PowerPlay Power Analyzer due to the assignment FLOW_ENABLE_POWER_ANALYZER
    Info: Quartus II Full Compilation was successful. 0 errors, 10 warnings
```

Figure 8 Compilation Report

3.3. Program the DE1 board

Connect the supplied USB lead from the PC to the **USB Blaster Port** of the DE1 Board (Figure 1) and power on the DE1 board by pressing the Red Power On/Off switch. Ensure that the **RUN/PROG** switch is set to **RUN**. Open the Programmer window by selecting **Tools > Programmer**. The programmer window should look like Figure 9. Note that the programming file for your project, *YI_dec7448.sof* will be selected by default. If USB-Blaster [USB-0] is not displayed next to **Hardware Setup** click on **Hardware Setup** and select the **USB-Blaster**. If no USB-Blaster is displayed call a Demonstrator.


Programmer - M:/altera/dec7448 - dec7448 - [dec7448.cdf]

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress:

☐ Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
dec7448.sof	EP2C20F484	001B2077	FFFFFFFF	✓	☐	☐	☐	☐	☐

Figure 9 The programmer window

Click on the **Start** button. Make sure that the Program/Configure check box is selected. The device will be examined, programmed and the programming will be verified.

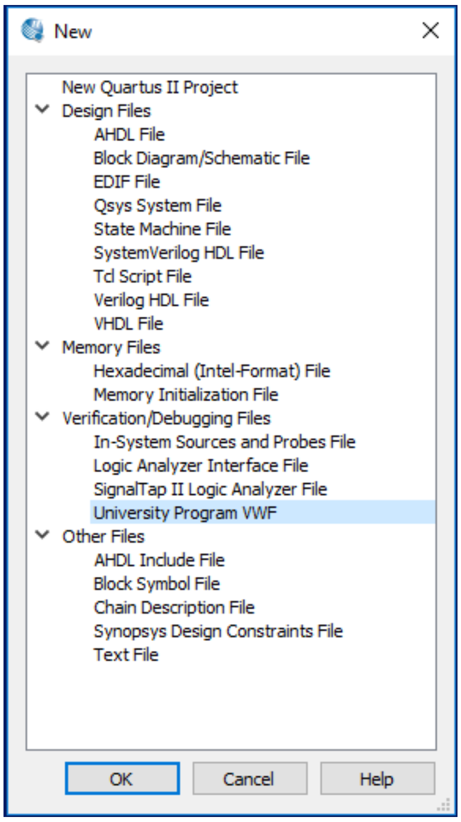
4. Simulation and Entry of the Input Waveform

We have entered our design and it has compiled without errors. If this were a C program, the next step would likely be to run the program and see if it worked as we expected. We could, at this point, program our chip and try it out and see if it works. However, it is better to be as certain as we can be that our design will work properly, before we program the FPGA. We will do this by using the Simulator.

Simply put, simulation means that you “pre-test” your designed project. This can allow you to predict what the project results should be for certain inputs. Imagine your design project is an adder of two inputs A and B, and it has an output Z. If your inputs are A = 1 and B = 4, you know that the output should be Z = 5. This can be tested by simulating the input signals, and viewing the outputs. After simulation you should see the output value Z = 5 to show, in principle, that the design project works correctly. You could also use several test inputs for simulation.

4.1. Waveform Entry for Simulation

In this experiment Altera’s simulator tool is used. From the File menu, select **New University Program VWF**, Figure 10.


New

- New Quartus II Project
- Design Files
  - AHDL File
  - Block Diagram/Schematic File
  - EDIF File
  - Qsys System File
  - State Machine File
  - SystemVerilog HDL File
  - Tcl Script File
  - Verilog HDL File
  - VHDL File
- Memory Files
  - Hexadecimal (Intel-Format) File
  - Memory Initialization File
- Verification/Debugging Files
  - In-System Sources and Probes File
  - Logic Analyzer Interface File
  - SignalTap II Logic Analyzer File
  - University Program VWF
- Other Files
  - AHDL Include File
  - Block Symbol File
  - Chain Description File
  - Synopsys Design Constraints File
  - Text File

OK | Cancel | Help

Figure 10 File menu, University program VWF

This command opens the Waveform Editor tool (Figure 11) which allows you to specify the desired input waveforms.

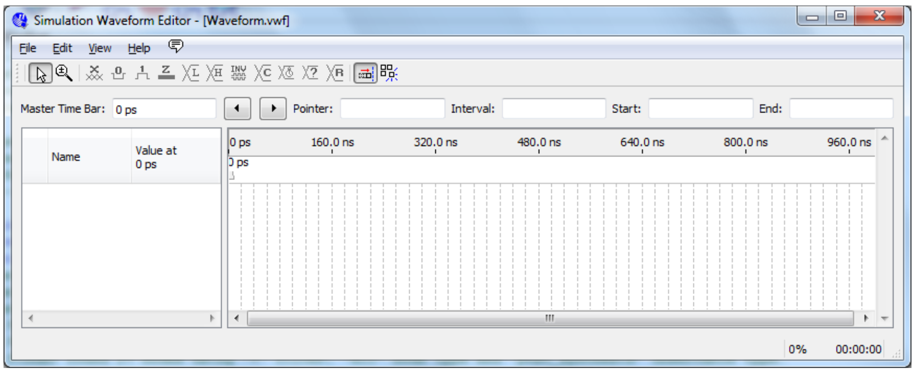


Figure 11 Waveform Editor tool

We will run the simulation for 1 μs; so, select **Edit > Set End Time** in the Waveform Editor and in the pop-up window that appears, specify the time of 1 us, and click OK.

Before drawing the input waveforms, it is necessary to locate the desired signals in the implemented circuit. In FPGA jargon, the term "node" is used to refer to a signal in a circuit. This could be an input signal (input node), output signal (output node), or an internal signal. We also need to find the input and output nodes. This is done by using a utility program called the Node Finder.

In the Waveform Editor window, select **Edit > Insert > Insert Node or Bus...** (Figure 12)

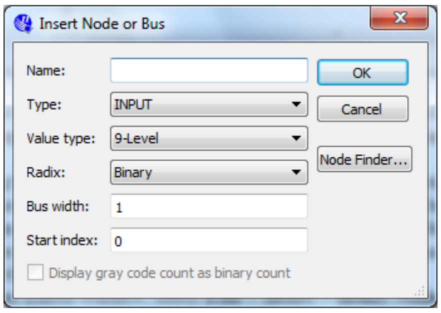
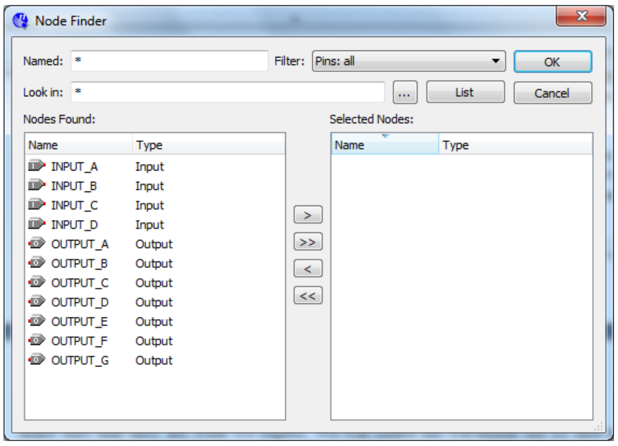


Figure 12 Insert Node or Bus Dialog

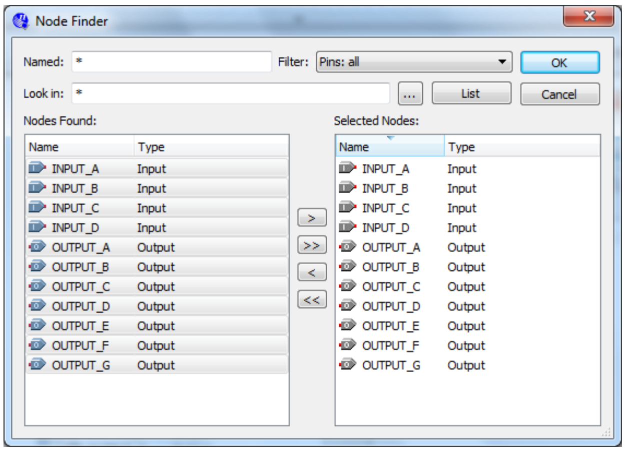
In the pop-up window that appears click on **Node Finder**. A filter is used to identify the nodes of interest. In our circuit, we are only interested in the nodes that appear on the pins (i.e. external connections) of the FPGA chip. Hence, the filter setting should be **Pins: all**. Click on **List**, which will display the nodes from your schematic as indicated in Figure 113.



Name	Type
INPUT_A	Input
INPUT_B	Input
INPUT_C	Input
INPUT_D	Input
OUTPUT_A	Output
OUTPUT_B	Output
OUTPUT_C	Output
OUTPUT_D	Output
OUTPUT_E	Output
OUTPUT_F	Output
OUTPUT_G	Output

Figure 13 Node Finder Dialog

In a large circuit there could be many nodes displayed. We need to select the nodes that we wish to observe in the simulation. This is done by highlighting the desired nodes and clicking on the **>** button. Figure 14 shows an example of all nodes selected.



Nodes Found:

Name	Type
INPUT_A	Input
INPUT_B	Input
INPUT_C	Input
INPUT_D	Input
OUTPUT_A	Output
OUTPUT_B	Output
OUTPUT_C	Output
OUTPUT_D	Output
OUTPUT_E	Output
OUTPUT_F	Output
OUTPUT_G	Output

Selected Nodes:

Name	Type
INPUT_A	Input
INPUT_B	Input
INPUT_C	Input
INPUT_D	Input
OUTPUT_A	Output
OUTPUT_B	Output
OUTPUT_C	Output
OUTPUT_D	Output
OUTPUT_E	Output
OUTPUT_F	Output
OUTPUT_G	Output

Figure 14 Node Finder Dialog with all nodes selected

Click OK in this window and OK in the **Insert Node or Bus** window. This returns to the Waveform Editor window, with the selected signals included (Figure 15).

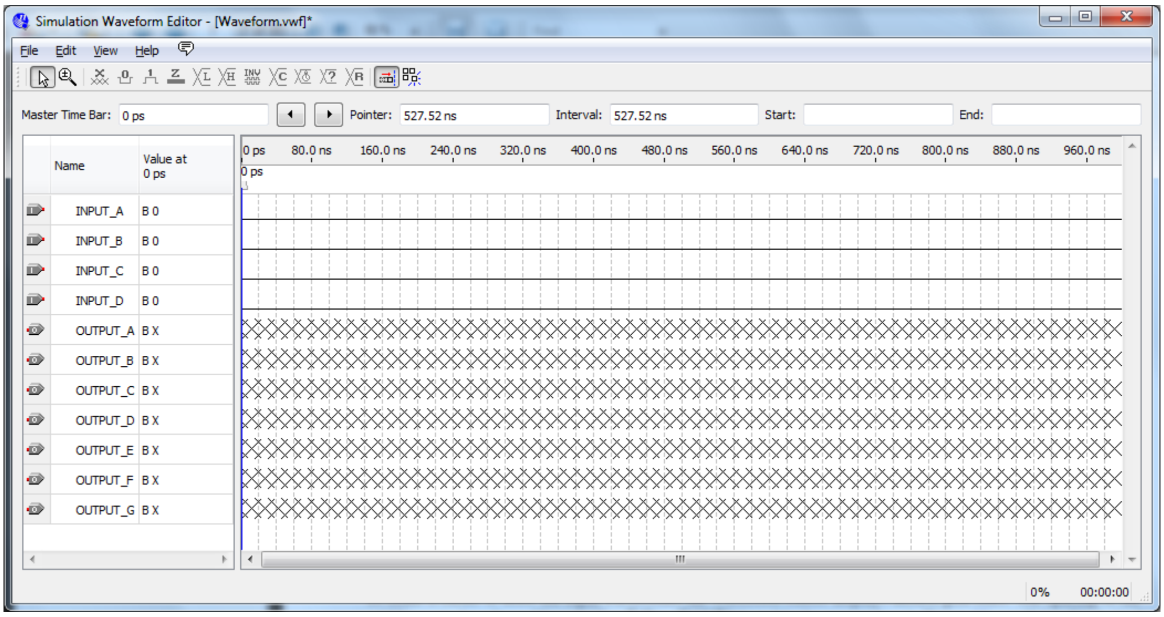
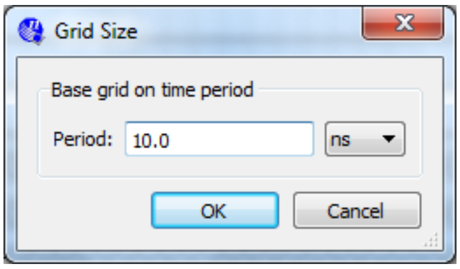


Figure 15 Waveform Editor with nodes selected

Observe that all input signals are at logic level 0 by default. The output signals are shown as undefined. Next, we have to draw the input waveforms. Then, we will simulate the circuit, which will produce the output waveform.

To make it easier to draw the input waveforms, the Waveform Editor displays dashed grid lines. The spacing of the grid lines can be adjusted by selecting **Edit > Grid Size**, and in the pop-up box specifying the desired size (Figure 16). Another convenience in drawing is to have transitions of a waveform snap on grid lines. This feature is activated by clicking on the Snap to Grid icon.



Period: 10.0 ns

Figure 16 Grid Size Selection Dialog

Input waveforms can be drawn in different ways. The most straightforward way is to indicate a specific time range and specify the value of a signal. To illustrate this approach, click the mouse on the **input_D** waveform near the 400 ns point and then drag the mouse to the 800 ns point. The selected time interval will be highlighted in blue. Change the value of the waveform to 1 by clicking on the Forcing High (1) icon (Figure 17).

In creating the waveform for input_D, we used the icon to implement the logic value 1. Another possibility is to invert the value of the signal in a selected time interval by using the Invert icon.

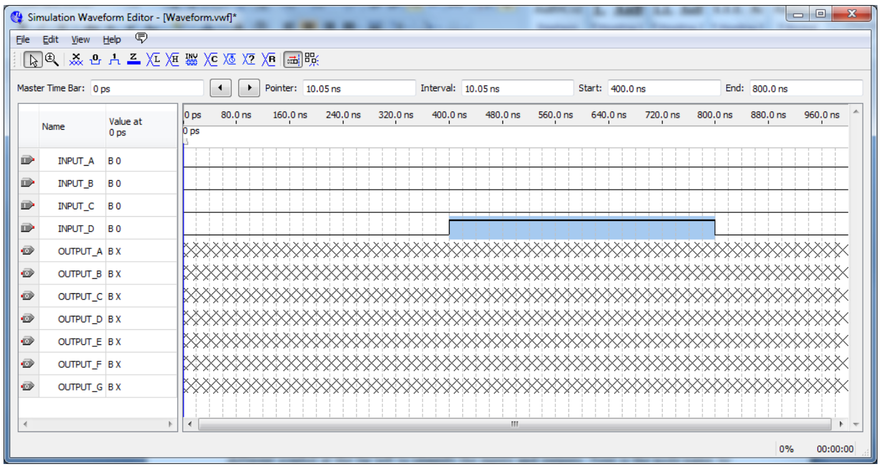


Figure 17 Input D set high

For **input_A** we will use a different approach. This signal should alternate between logic values 0 and 1 at each 50-ns interval. Such a regular pattern is indicative of a clock signal that is used in many logic circuits. Even though there is no clock signal in our example circuit, it is convenient to specify **input_A** in this manner.

Click on the **input_A** input, which selects the entire 1 us interval. Then, click on the Overwrite Clock icon (the icon that looks like a stopwatch). Specify the clock period of 100ns and the duty cycle of 50%, and click OK (Figure 18).

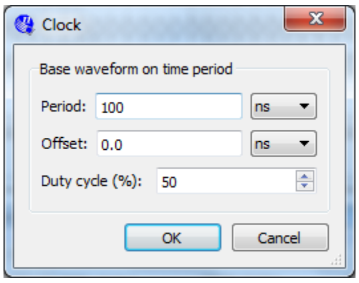

Clock
Base waveform on time period
Period: 100 ns
Offset: 0.0 ns
Duty cycle (%): 50

Figure 18 Clock Waveform Dialog

Use whichever method you think appropriate to modify the **input_B** and **input_C** inputs, to complete the waveform as shown in Figure 19.

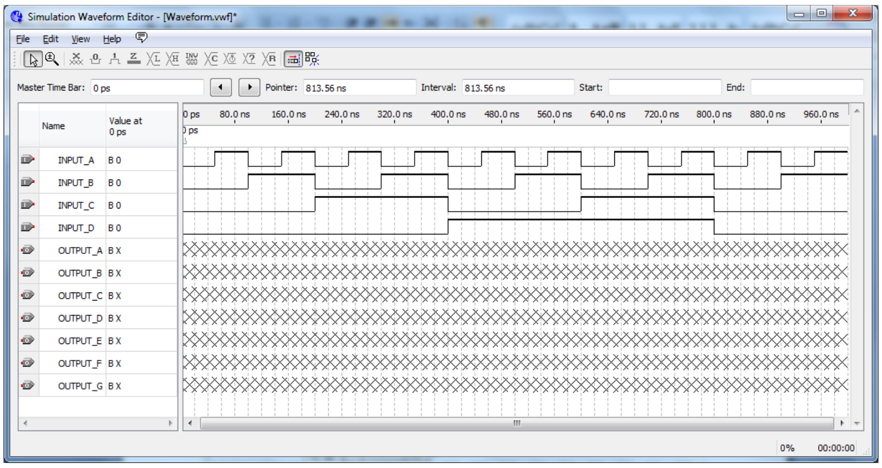


Figure 19 Completed Input Waveform

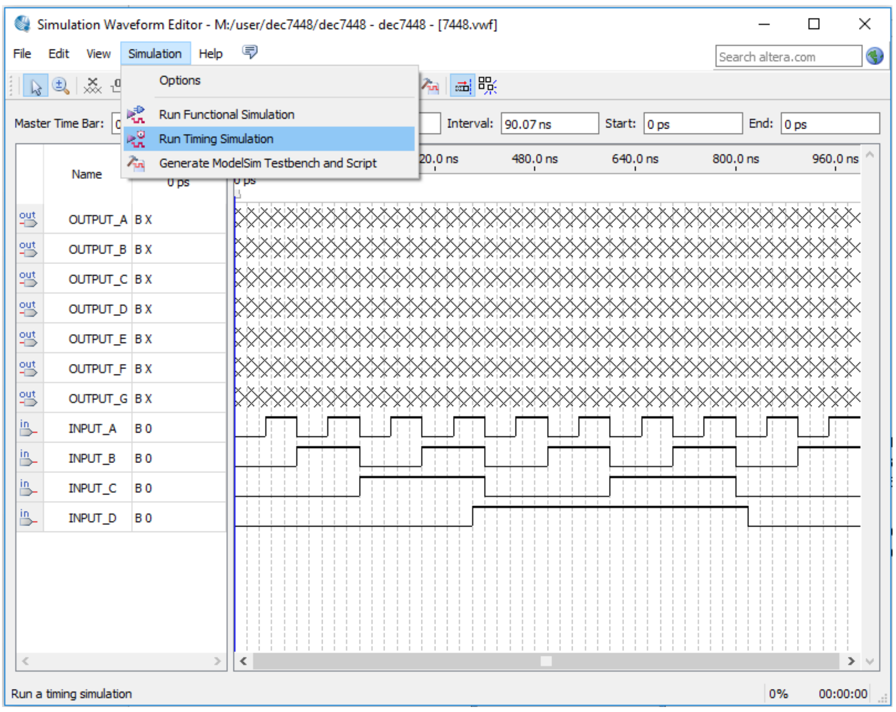
Save the waveform file with the **File** > **Save As** command using a filename which starts with your initials, in this case **YI_7448.vwf**, within your "YI_dec7448" folder created previously (assuming this is part of your work on the 7448 decoder). The suffix vwf stands for *vector waveform file*.

4.2. Simulation

Return to the simulation window.

If doing extra or preparatory simulation work at home on your own laptop/PC, under the Simulation menu, first select Options (above the selection seen in Figure 20.) Under Options, you will need to ensure "Quartus II Simulator" is selected (not ModelSim, which is not installed on your home computer) and press OK. An information box will appear warning you about approximate timings. Click OK again.

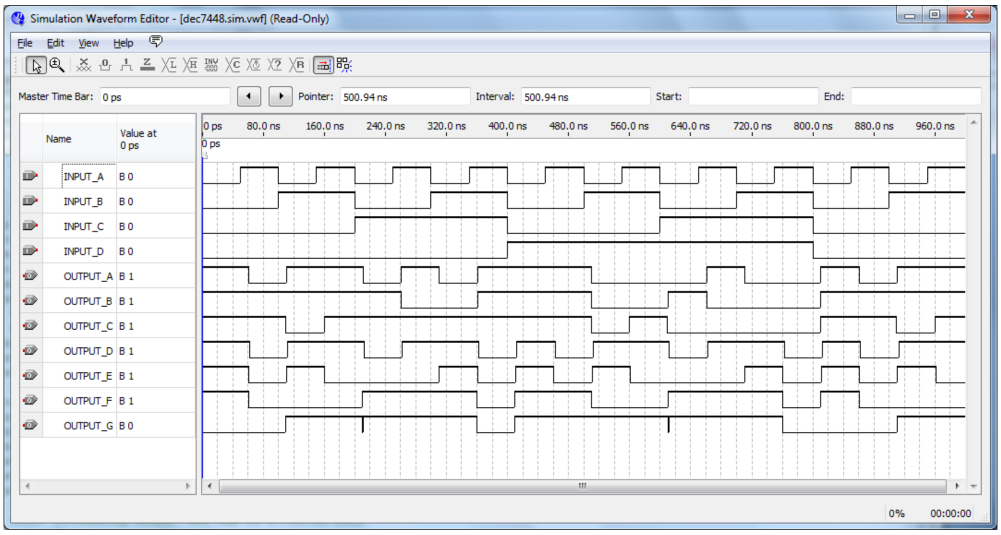
Now open the Simulation menu and select *Run **Timing** Simulation*, Figure 20.



Master Time Bar: 0 ps — Interval: 90.07 ns — Start: 0 ps — End: 0 ps

Figure 20 Timing simulation setting

The simulation tool will now display the waveforms produced in the simulation process (Figure 21).

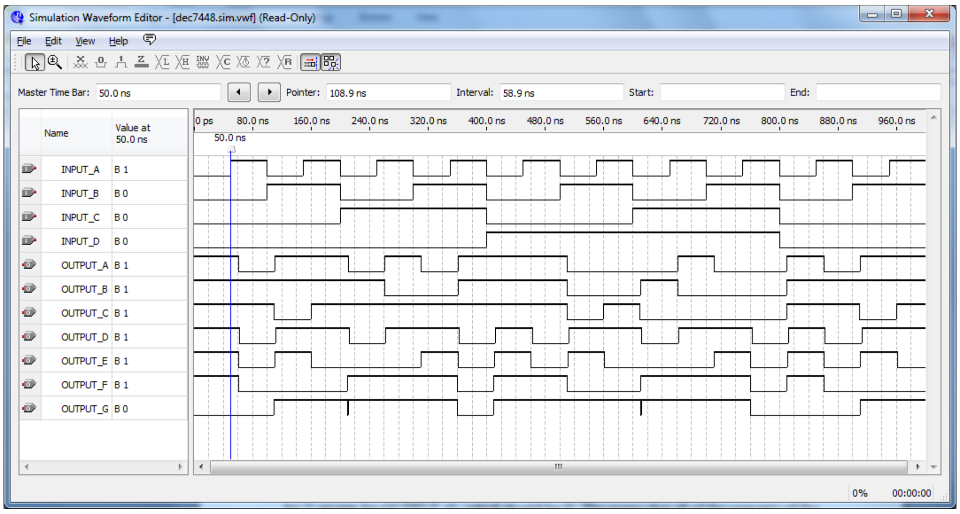


Master Time Bar: 0 ps — Pointer: 500.94 ns — Interval: 500.94 ns — Start: — End:

Figure 21 Simulation Waveform output

Note that if doing extra or preparatory work on your own laptop/PC outside the lab, the precise timings in your simulation may not precisely match what is seen in Figures 21 and 22 (e.g. changes in the Output values may not be precisely aligned). This is because some of the timing used by the Quartus II simulator is approximate. The following discussion is based around the 'ideal' option (ModelSim) available on the lab PCs. You can still compare preliminary results obtained at home to this discussion.

The reference cursor should start at 0. The inputs should be '0000' and the outputs should all be '1' except for *OUTPUT_G*, which should be '0'. Our design assumes that this will cause all of the segments of the display will be on, except for segment G, which is the centre horizontal segment of a 7 segment display. Since this is the pattern to display a '0', we can see that the outputs are correct according to our design **. Click the right arrow at the top of the window to move the reference cursor to the right. It should stop at 50ns. The inputs are now '0001', but the outputs have not changed. This might seem like an error until you realize that the simulator accurately simulates the propagation delays of the FPGA. The inputs change at 50ns, but the outputs cannot change instantly and thus remain the same (Figure 22).



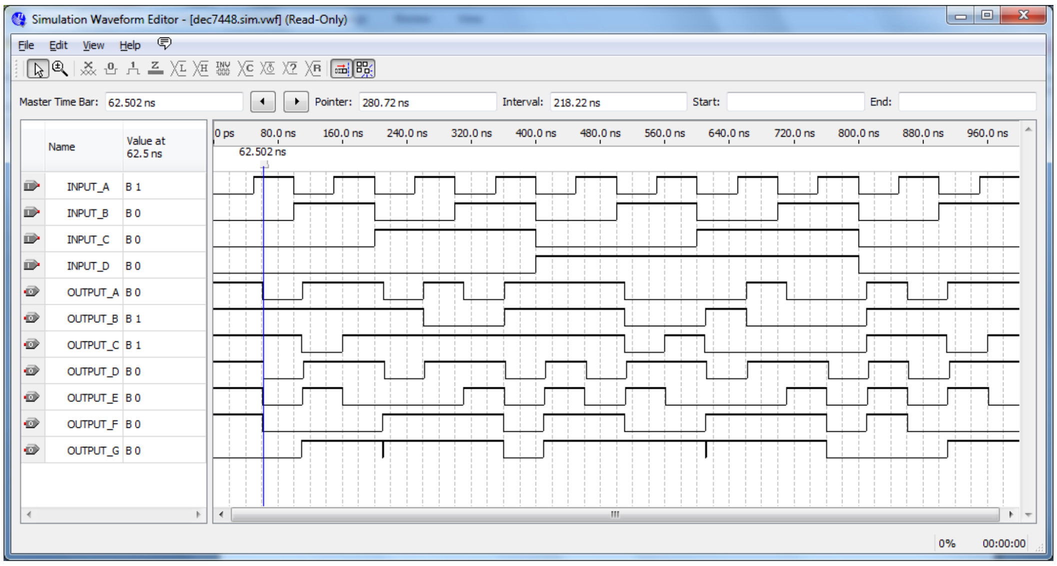
Master Time Bar: 50.0 ns Pointer: 108.9 ns Interval: 58.9 ns Start: End:

Name	Value at 50.0 ns
INPUT_A	B 1
INPUT_B	B 0
INPUT_C	B 0
INPUT_D	B 0
OUTPUT_A	B 1
OUTPUT_B	B 1
OUTPUT_C	B 1
OUTPUT_D	B 1
OUTPUT_E	B 1
OUTPUT_F	B 1
OUTPUT_G	B 0

Figure 22 Waveform at 50ns

Move the reference cursor to the right again until it stops at the falling edge transition of one of the outputs. In the example in Figure 23 (below) this was at 62.502ns but yours may be different. (If the time bar jumped to 100ns it means that it has been snapped to grid. On the **Edit** menu click on the **Snap to Transition** item to uncheck it.) The inputs will still show '0001' and the outputs will now show all zeroes except for **OUTPUT_B** and **OUTPUT_C** which will be '1'. If segments B and C of the display are on, this would be the correct pattern for a '1', so the inputs and outputs are now correct according to our design **(but our initial design is flawed see below)**. We can see that it took 12.502ns, (from 50.0ns to 62.502ns) for the outputs to settle at the correct values for the inputs. This is the propagation delay for our design. Move the reference cursor through the rest of the waveform to verify all of the outputs.

When you try to program the DE1 board to run the design based around the 7448 module, the 7-segment display will not appear as expected, although it might look spookily familiar. In fact, our design (in the main lab script) has missed a crucial fact about the 7-segment display, mentioned in Section 2.1 above, implying there's a simple error in the design. How can you correct this problem with NOT gates? **See page 18 of the main lab script for the answer!**


Simulation Waveform Editor - [dec7448.sim.vwf] (Read-Only)

Master Time Bar: 62.502 ns   Pointer: 280.72 ns   Interval: 218.22 ns   Start: End:

Name	Value at 62.5 ns
INPUT_A	B 1
INPUT_B	B 0
INPUT_C	B 0
INPUT_D	B 0
OUTPUT_A	B 0
OUTPUT_B	B 1
OUTPUT_C	B 1
OUTPUT_D	B 0
OUTPUT_E	B 0
OUTPUT_F	B 0
OUTPUT_G	B 0

Figure 23 Waveform at 65.502ns

The other available option is a **functional** simulation. Such a simulation does not wait for combinational changes in signals to propagate, but instead, simulates as if combinational changes are seen immediately. This can be useful for a simpler check on the essential logic of the system, but it may miss important propagation delays that could impact the real operation of a system, such as signals propagating too late for the desired active edge of the clock, or too late to satisfy flipflop setup times, for example. Have a play with a functional simulation of the above, and observe the differences.

Version history

Name	Date	Version
Dr D G McIntosh	April 2022	Ver. 8.1
Dr D G McIntosh	April 2021	Ver. 8
Dr D G McIntosh	April 2020	Ver. 7
Dr D G McIntosh	March 2020	Ver. 6.5
Dr M López-Benítez	September 2019	Ver. 6.4
Dr M López-Benítez	December 2017	Ver. 6.3
Dr L Esteban and Dr M López-Benítez	November 2017	Ver. 6.2
Dr A Al-Ataby	October 2014	Ver. 6.1
Dr A Al-Ataby, F Davoodi Samirmi and Prof J Smith	October 2013	Ver. 6
Dr A Al-Ataby	October 2012	Ver. 5.8
Prof J S Smith	October 2011	Ver. 5.7
Dr S Amin-Nejad	July 2011	Ver. 5.6