

Circuit Design and Simulation with VHDL

2nd edition

Volnei A. Pedroni

MIT Press, 2010

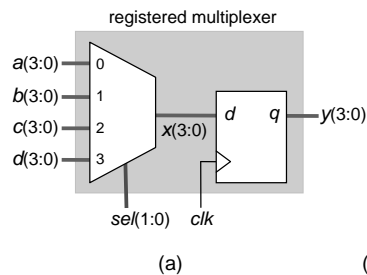
Book web: www.vhdl.us

Appendix B

Quartus II Tutorial

(Quartus II 13.0 sp1)

This tutorial is based on **Quartus II 13.0 sp1 Web Edition** (free at www.altera.com). The circuit used in the tutorial is the registered multiplexer of figure 1a, with the VHDL code of figure 1b.



(b)

```
1  -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  -----
5  ENTITY reg_mux IS
6  PORT (
7      a, b, c, d: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
8      sel: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
9      clk: IN STD_LOGIC;
10     y: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
11 END ENTITY;
12 -----
13 ARCHITECTURE reg_mux OF reg_mux IS
14     SIGNAL x: STD_LOGIC_VECTOR(3 DOWNTO 0);
15 BEGIN
16     x <= a WHEN sel="00" ELSE
17         b WHEN sel="01" ELSE
18         c WHEN sel="10" ELSE
19         d;
20
21     PROCESS(clk)
22     BEGIN
23         IF rising_edge(clk) THEN
24             y <= x;
25         END IF;
26     END PROCESS;
27 END ARCHITECTURE;
```

Figure 1


1. Starting a New Project

a) Launch Quartus II, which opens the screen of figure 2. Click **Create a New Project**, which leads to figure 3. *Note:* If Quartus II is already open, do **File > New Project Wizard** to get the screen of figure 3.

b) Set the design location and name (see figure 3):

- Enter the name of the directory where all project files should be located (*reg_mux*, in this tutorial). If the directory does not exist yet, just type in the desired name (with the proper path) and Quartus II will create it for you.
- Enter the project's name (preferably, use the same name for the directory, for the project, and for the entity).
- Note that the entity's name is filled in automatically.

Click **Finish** until the Project Navigator (figure 4a) is displayed.

c) Enter the VHDL code of figure 1b. To do so, open the VHDL editor by clicking  or by selecting **File > New**, which prompts the list to figure 4b. Select **VHDL File** and click **OK**. After typing the VHDL code, save it as *reg_mux.vhd*.

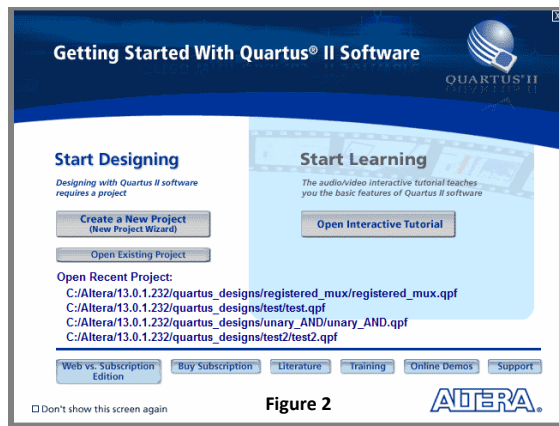


Figure 2

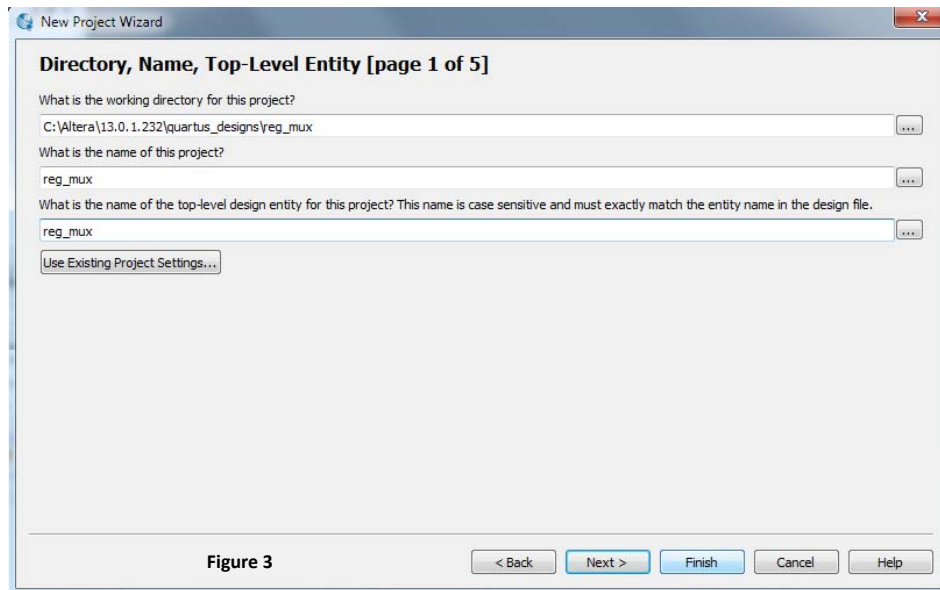
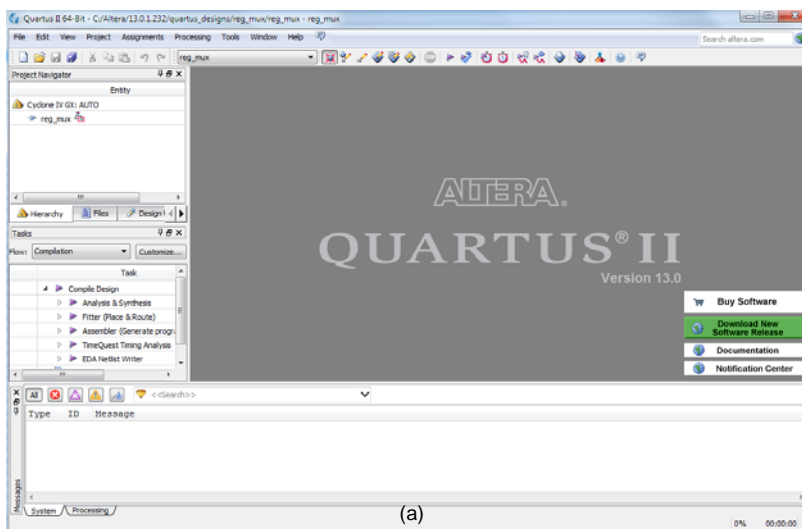
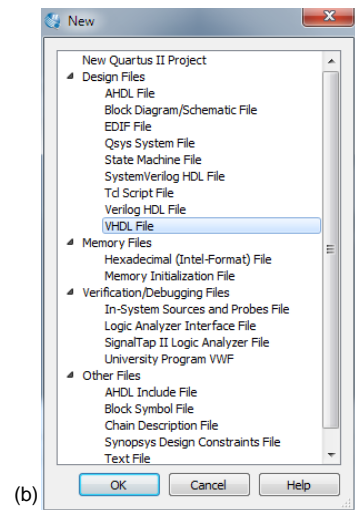


Figure 3



(a)






(b)

Figure 4

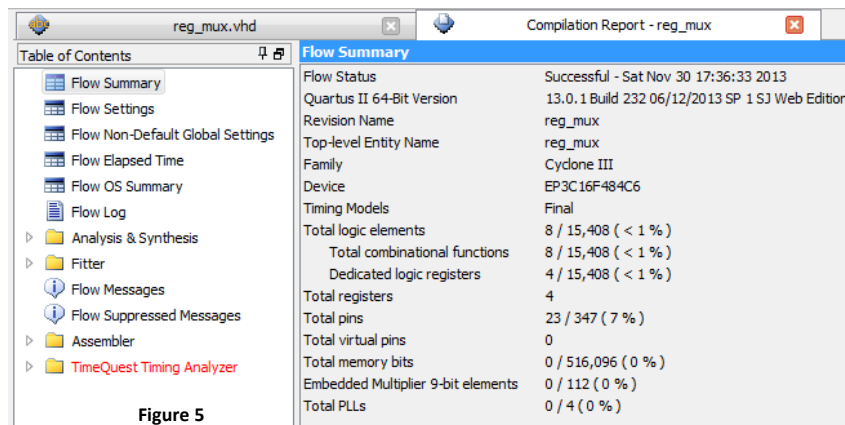
2. Synthesizing the Design

a) Define the device in which the circuit should be implemented by selecting **Assignments > Device**. In the Family field, select **Cyclone III**. In the Target Device field, mark **Specific device selected in 'Available devices' list** and select EP3C16F484C6 (FPGA of the DE0 board).

b) Compile the code by clicking  or by selecting **Processing > Start Compilation**.

Note: For a faster compilation, particularly useful while the VHDL code is still being debugged, click  or select **Processing > Start > Start Analysis and Synthesis**. In this case, no timing information is recorded. After the code is working properly, full compilation () should then be performed.

c) When the compilation ends, the Compilation Report of figure 5 is exhibited, which contains several pieces of valuable information, some of which are described in the next section.



Flow Summary	
Flow Status	Successful - Sat Nov 30 17:36:33 2013
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	reg_mux
Top-level Entity Name	reg_mux
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	8 / 15,408 (< 1 %)
Total combinational functions	8 / 15,408 (< 1 %)
Dedicated logic registers	4 / 15,408 (< 1 %)
Total registers	4
Total pins	23 / 347 (7 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 5

3. Inspecting Synthesis Results

This section describes some of the results produced by the compiler.

a) *Device type and number of pins:* Check in figure 5 if the device type is the intended one (Cyclone III EP3C16F484C6). Check also if the total number of pins is as expected ($4 \times 4 + 2 + 1 = 19$ inputs + 4 outputs = 23 pins).

b) *Number of logic elements:* Figure 5 shows also the amount of logic needed to implement the circuit. In this case, 8 logic elements were needed, out of 15,408 LEs available in that device.

c) *Number of registers:* Since y is a 4-bit signal (figure 1a), 4 flip-flops are expected to be inferred by the synthesizer. This also can be checked in figure 5, which shows a total of 4 registers.

d) *RTL View:* This tool shows how the code was interpreted by the compiler (before optimization and fitting). Select **Tools > Netlist Viewers > RTL Viewer**, which exhibits the circuit of figure 6 (or equivalent). Because 2-input multiplexers are available in the device, a total of 3 units were used to implement the 4-input mux (the other three blocks at the input are for processing sel). Note also the 4-bit register (4 D-type flip-flops —DFFs) at the output.

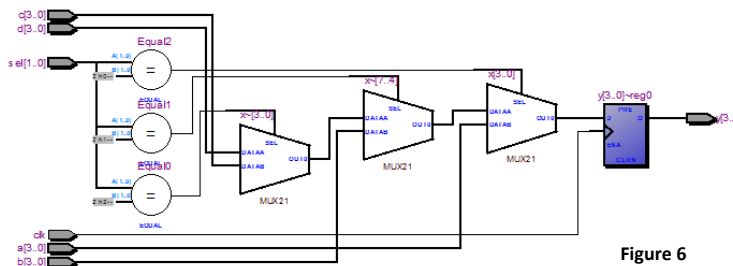



Figure 6

e) *Equations*: They represent the actual circuit implemented by the compiler. In the Compilation Report, select **Fitter > Equations** (if this option is not available, go to **Tools > Options > General > Processing** and mark **Automatically generate equation files during compilation**, then recompile the code). To interpret the equations, see first section 7. For example, “A1L41Q = DFFEAS(A1L60, GLOBAL(A1L2),...);” means that the output of cell A1L41Q is produced by a DFF whose data input comes from cell A1L60 and whose clock input is fed by a global signal from cell A1L2. Confirm in the equations that the total number of DFFs is indeed 4.

f) *Timing analysis*: In the Compilation Report, select **TimeQuest Timing Analyzer > Slow 1200mv 85C Model > Worst-Case Timing Paths** and check the minimum clock pulse width.

g) *Pin assignments*: This will be treated in section 5.

4. Simulating the Circuit

a) To perform manual graphical simulation (which is the only kind supported by Quartus II), we must first draw the input waveforms, based on which the simulator will calculate and plot the output waveform. Click  or select **File > New**, which will prompt again the list of figure 4b.

b) Select **Verification/Debugging Files > University Program VWF** and click **OK**. This opens the wave pane of figure 7.

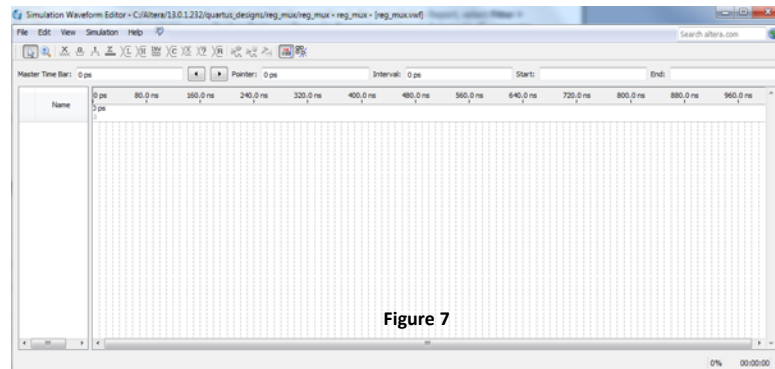
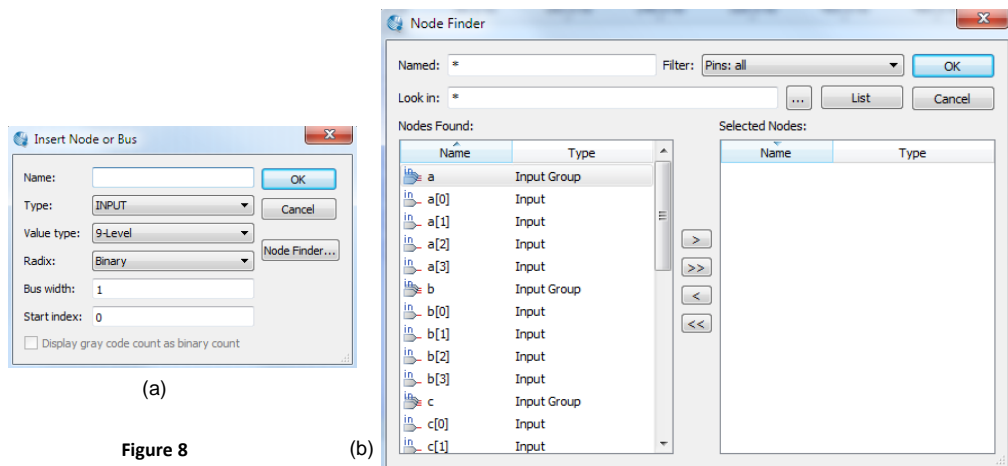



Figure 7

c) The time axis in figure 7 goes from 0 to 1 us. If a different end time is needed, select **Edit > Set End Time** and enter the desired value.

d) The grid can also be adjusted as needed by selecting **Edit > Grid Size**. Keep the default value (10 ns).

e) Now add signals to the waveform editor. Press the right mouse button in the white area under Name (in figure 7) and select **Insert > Insert Node or Bus**, which leads to figure 8a.



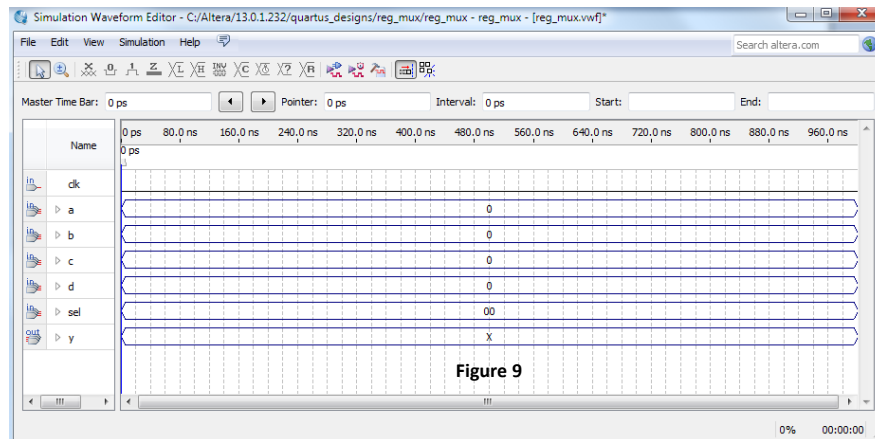
(f) Click **Node Finder**, which opens the screen of figure 8b (but empty). In the Filter field, select **Pins: all & Registers: post-fitting** (a simpler option is **Pins: all**), then click **List**. The left column will be filled with the signals (partially) shown in figure 8b. Click on the desired signals to select them and pass them to the column on the right by clicking . Click **OK** twice, which will fill the waveforms window with the signals of figure 9.

g) To change the position (order) of a signal in figure 9, click on its name to select it, then click again and hold the mouse button, dragging next the signal to the desired position. Do as in figure 9, with all inputs first and the clock at the top.

h) Before drawing the waveforms, set the desired representation radix.


- Click the left mouse button on *clk* (figure 9) to select it, then click the right mouse button and select **Radix > Binary**.

- Click the left mouse button on *a*, *b*, *c*, *d*, *sel*, and *y* to select them, then click the right mouse button and select **Radix > Unsigned Decimal**.




i) Now we must draw the input waveforms (*clk*, *a*, *b*, *c*, *d*, *sel*), after which the simulator will compute and draw the output waveform (*y*). The stimuli of figure 10 are adopted.

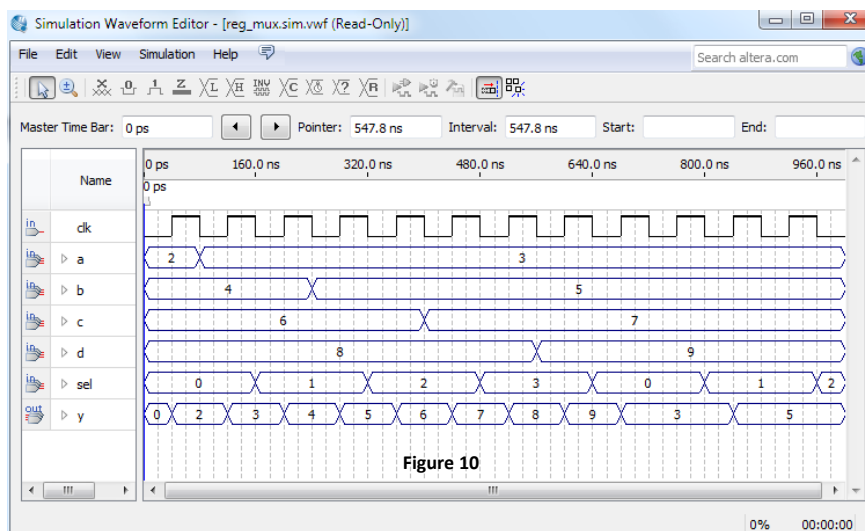
- Click on *clk* to select it. Click then the clock icon  and enter 80 ns for the period and 0 for the offset.



- Click on *a* to select it. Then click the arbitrary value icon  and enter 2. Now select line *a* only from 80 ns up and enter 3 ns, using again the arbitrary value icon.

- Repeat the process above for *b*, *c*, and *d*, entering the values shown in figure 10.

- Click on *sel* to highlight it. Then click the counter icon  and enter Start value = 0, Increment = 1, Count type = Binary, and Count every = 160 ns.


j) Save the file with the same name as the entity's name and extension *.vwf* (vector waveform file), that is, *reg_mux.vwf*. Recall that the last waveform (*y*) will be filled by the simulator.



- k) Choose the simulator in **Simulation > Options** (ModelSim or Quartus II Simulator).
- l) Run the simulation:
- For *functional simulation*, click  or select **Simulation > Run Functional Simulation**.
 - For *timing simulation*, click  or select **Simulation > Run Timing Simulation**.
- m) Inspect the results (compare them to those in figure 10).

5. Making Pin Assignments

Case 1: Manual pin assignments

- a) Select **Assignments > Pin Planner** or click , which opens the window of figure 11.
- b) In the Location column, enter the desired pin numbers.
- c) Recompile the code.

Note: Assignments can be deleted with **Assignments > Remove Assignments**.

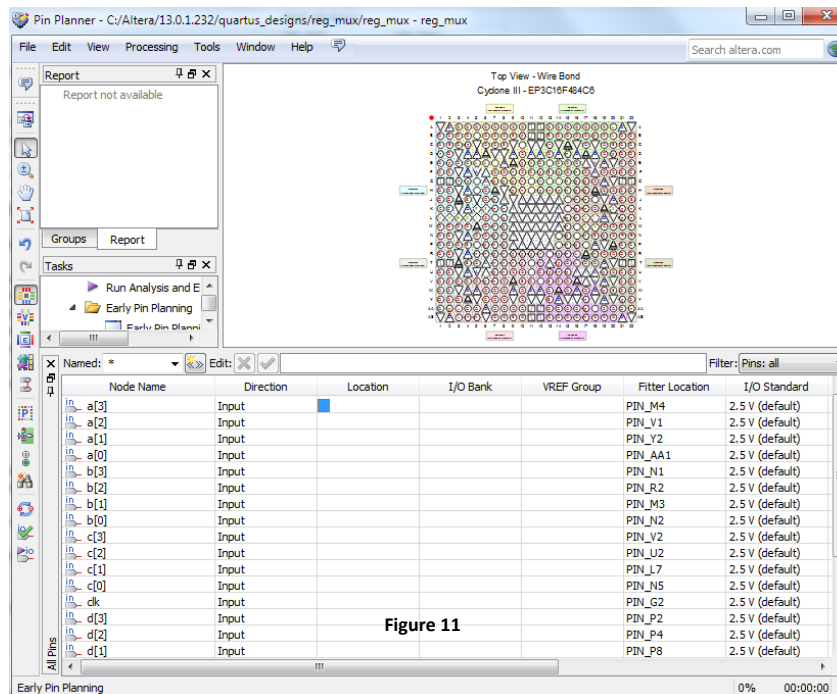


Figure 11

Case 2: Importing pin assignments with a CSV file

Pin assignments can be imported and exported using CSV (comma separated value) files. The initial pin assignments, made automatically by the compiler, cannot be exported.

- a) Select **Assignments > Import Assignments** and choose the corresponding .csv file.
- b) Recompile the code.

Note: To export a pin assignment, select **Assignments > Pin Planner**, then **File > Export**.

6. Downloading the Design to the FPGA

a) Connect the FPGA board to an USB port of your computer and turn the board's power on.

Note: If this is the first time that you are using a board that interfaces using an USB port, the USB-Blaster Driver must be installed (generally done automatically).

b) In Quartus II, click the Programmer icon  or select **Tools > Programmer**, which opens the window of figure 12. Note the following:

- The hardware driver is USB-Blaster.
- The mode is JTAG.
- The programmer file is *reg_mux.sof*.
- The Program/Configure box is checked.

c) Click **Start**, and the device will be programmed. Observe what happens to the board's LEDs during programming.

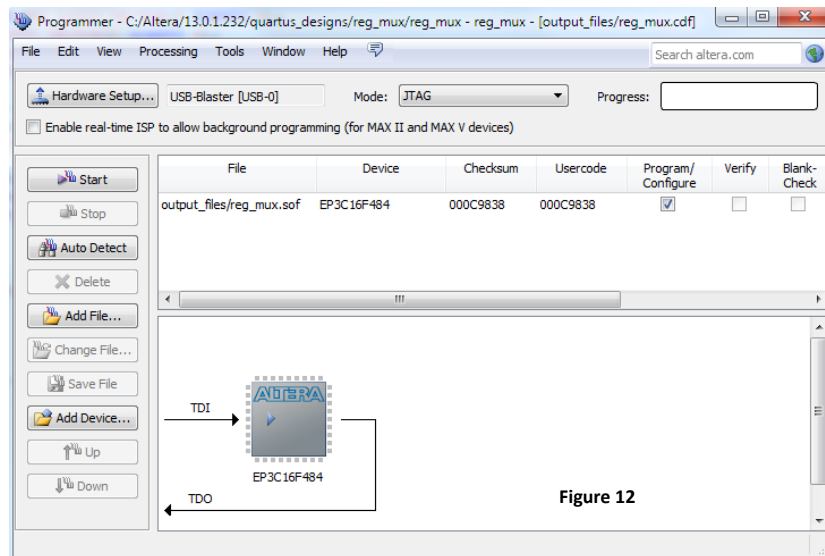


Figure 12

7. Interpreting the Fitter Equations

Below are the main symbols used in the Fitter equations.

a) Logic operators: ! (NOT), & (AND), # (OR), \$ (XOR)

b) Flip-flops:

DFF (D, CLK, CLRN, PRN) (DFF with reset and preset, both active low)

DFFE (D, CLK, CLRN, PRN, ENA) (DFF above plus enable)

DFFEA (D, CLK, CLRN, PRN, ENA, ADATA, ALOAD) (DFF above plus asynchronous data load)

DFFEAS (D, CLK, CLRN, PRN, ENA, ADATA, ALOAD) (DFF above with synchronous clear)

TFFE (T, CLK, CLRN, PRN, ENA) (TFF with reset, preset, and enable)

Note: Recall that in the context of this book/course a reset signal is called *reset* when it is *asynchronous* and it is called *clear* when it is *synchronous*.