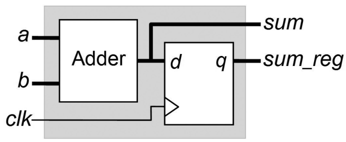
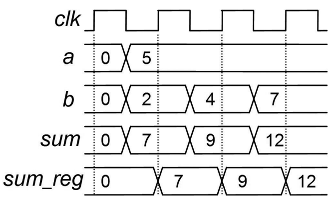
# Appendix B: Quartus Prime Tutorial

## 1. Introduction

This tutorial is based on *Quartus Prime 18.0 Lite Edition* (free at intel.com). The circuit used in the tutorial is the registered unsigned adder of figure B.1a, synthesized with the VHDL code of figure B.1b and simulated with the stimuli of figure B.1c. The adder inputs (*a*, *b*) are 3-bit signals, while its output (*sum*) is a 4-bit signal, so overflow never occurs.




(a) Registered adder.




(c) Stimuli (and expected responses) employed in the simulations.

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

entity registered_adder is
  port (
    clk: in std_logic;
    a, b: in std_logic_vector(2 downto 0);
    sum, sum_reg: out std_logic_vector(3 downto 0));
end entity;

architecture rtl of registered_adder is
begin
  sum <= std_logic_vector(('0' & unsigned(a)) + unsigned(b));
  process(clk)
    begin
      if rising_edge(clk) then
        sum_reg <= sum;
      end if;
  end process;
end architecture;
```

(b) VHDL design file.

**Figure B.1**

## 2. Synthesizing the Design

a) Start Quartus Prime, which opens the screen of figure B.2a.

b) Click **New Project Wizard**, then **Next**, which opens the screen of figure B.2b. Enter the project location and name (as shown in the figure) and click **Finish**, which leads to figure B.3a.

c) If this is your first access to Quartus Prime, do the following:

- Set up the default directory for all of your projects at **Tools > Options > General**. At the bottom of the screen, enter, for example, **Default file location:** *C:/ … /my_designs*.

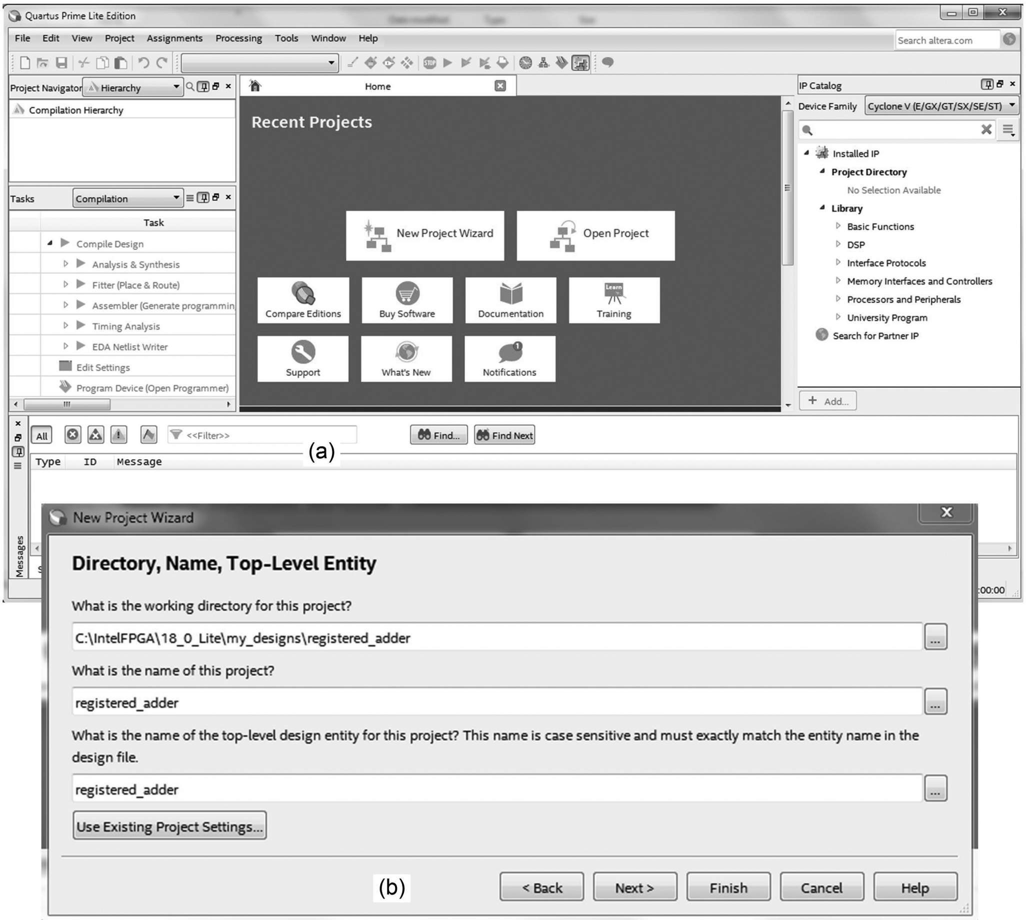



**Figure B.2**

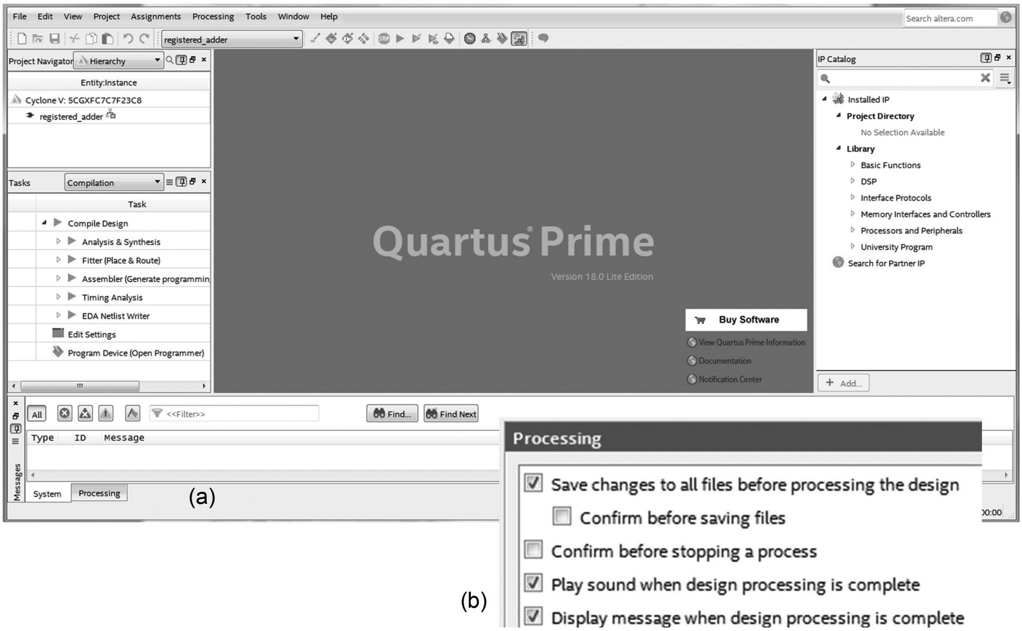



**Figure B.3**

- Set up the local feedback options by going to **Tools > Options > Processing** and making the selections shown in figure B.3b.

d) To enter the VHDL code (figure B.1b), open the VHDL editor by clicking on 🗋 or by selecting **File > New** and choosing **VHDL File** as the entry type. After typing the code, save it as *registered_adder.vhd*.

e) Define the VHDL version by selecting **Assignments > Settings > Compiler Settings > VHDL Input** and marking **VHDL 2008**.

f) Select the FPGA in **Assignments > Device**. If the design is not going to be downloaded to any device, choose a simpler FPGA family (for example, Cyclone IV over Cyclone V or 10) because compilation is usually faster (and there is no direct timing simulation support for the latter two). The EP4CE10F17C6 device is employed here.

g) Compile the code by clicking ▶ or by selecting **Processing > Start Compilation**. When the compilation ends, the Compilation Report of figure B.4 is exhibited. This report contains several pieces of valuable information, some of which are described in the next section.

*Note:* You can also run just **Analysis and Synthesis** ( ) until the syntax has been checked/fixed.

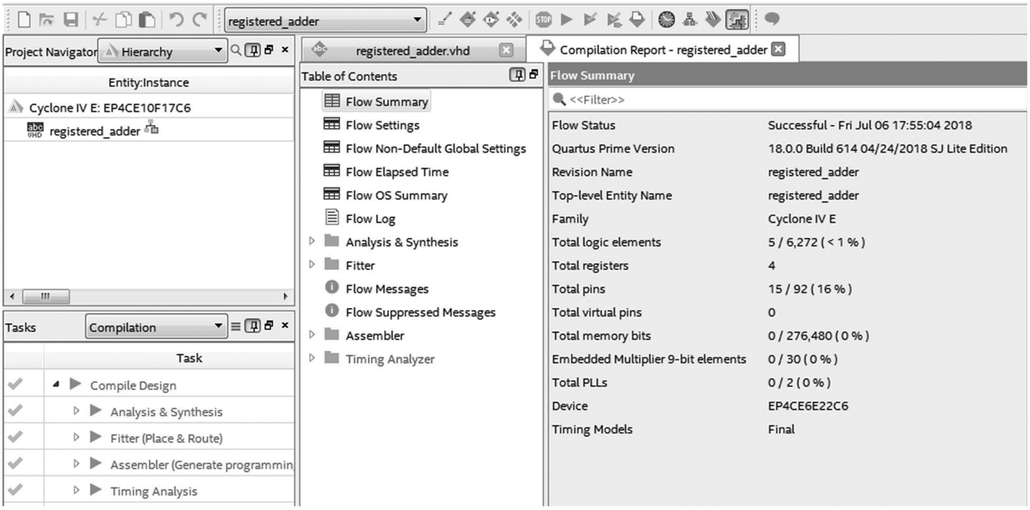



**Figure B.4**

## 3. Inspecting Synthesis Results

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

a) *Device type and number of pins:* Check in figure B.4 if the device is the intended one (in case one was selected). Check also whether the total number of pins is as expected (3 + 3 + 1 = 7 inputs and 4 + 4 = 8 outputs, totaling 15 pins).

b) *Resources usage:* It can be given in number of logic elements (LEs), number of LUTs, number of adaptive logic modules (ALMs), and so on and depends on the FPGA. Note that it is given in “number of LEs” in figure B.4, with 5 used out of > 6k.

c) *Number of registers:* This is the number of D-type flip-flops (DFFs) inferred by the compiler. Since *sum_reg* is a 4-bit signal (figure B.1a), 4 flip-flops are needed, which is indeed what figure B.4 says.

d) *RTL View:* This tool shows how the code was *understood* by the compiler. Select **Tools > Netlist Viewers > RTL Viewer**, which exhibits the circuit of figure B.5 (or equivalent). Note that it is in perfect agreement with figure B.1a.

e) *Implemented circuit:* The actual implementation can be seen at **Tools > Netlist Viewers > Technology Map Viewer (Post-Fitting)**. The compiler always tries to make simplifications, so in some cases this view is slightly different from the RTL view (but with the same functionality).

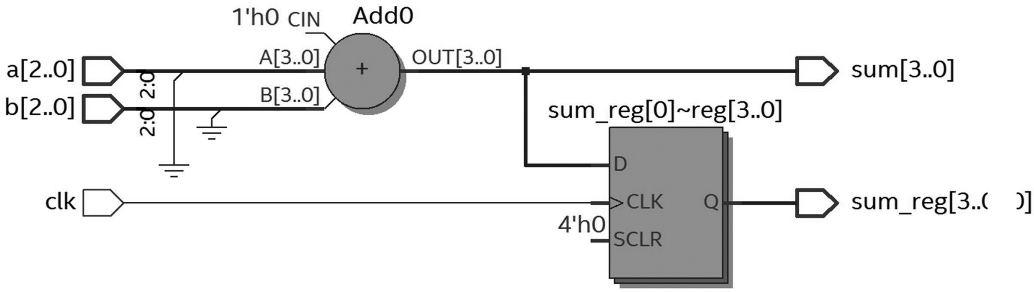



**Figure B.5**

f) *Equations:* They correspond to the implemented circuit. To see them, select **Table of Contents** > **Fitter** > **Equations**. If the equations are not shown, go to **Tools** > **Options** > **General** > **Processing** and mark **Automatically generate equation files during compilation**, then recompile the code. See also this appendix's part 7 below (Interpreting Fitter Equations).

## 4. Simulating the Circuit

This is a manual graphical simulation; we draw the input waveforms, based on what the simulator calculates and how it plots the output waveforms. For an automated simulation (with testbenches or Tcl scripts), use the ModelSim simulator (Appendix C), which is provided with the Quartus Prime software, directly.

*Note:* Quartus Prime 18.0 does not support direct timing simulation for Cyclone V and 10 FPGAs (the timing simulation shown here is for a Cyclone IV E device).

a) Click 🗋 or select **File** > **New**. Select then **Verification/Debugging Files** > **University Program VWF** and click **OK**. This opens the wave pane of figure B.6a. Note the default end time of 1 μs.

b) Now add signals to the waveform editor. To do so, press the right mouse button in the white area under **Name** (figure B.6a) and select **Insert** > **Insert Node or Bus**, which leads to figure B.7a.

c) Click **Node Finder**, which opens the screen of figure B.7b (only the screen will be empty in your version). In the **Filter** field, select **Pins: all & Registers: post-fitting** (or **Pins: all** if only the circuit ports must be exhibited), then click **List**. The left column will be filled with all design signals, partially shown in figure B.7b. Click [>>] to send all signals to the right or select just the desired signals and click [>]. Click OK twice, which leads to figure B.6b.

d) Note in figure B.6b that the signals are arranged in alphabetical order. Move *clk* to the top by clicking on it and holding, then dragging it to the desired position.

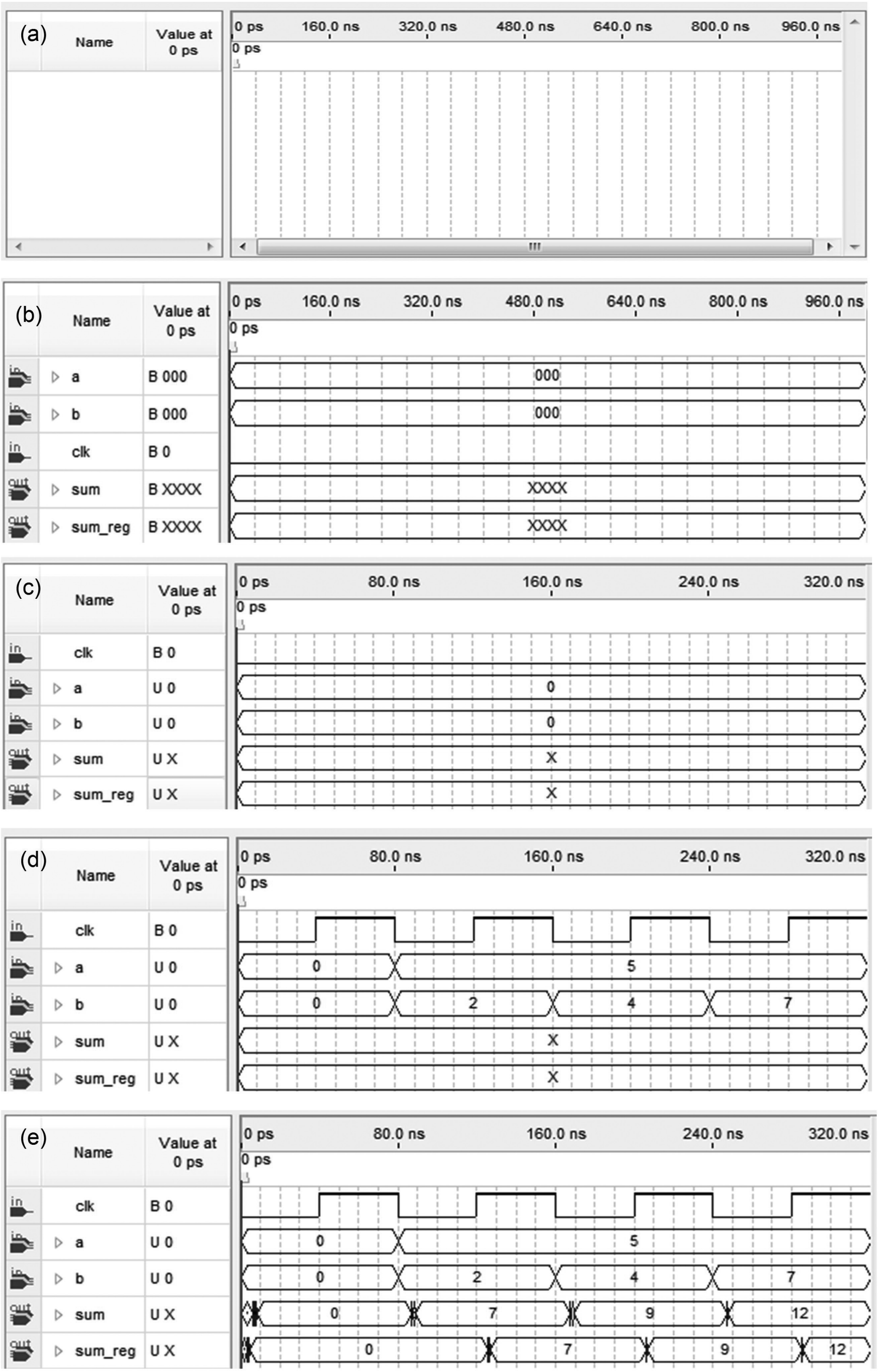

(a)
Name
Value at 0 ps
0 ps
160.0 ns
320.0 ns
480.0 ns
640.0 ns
800.0 ns
960.0 ns
(b)
a
B 000
b
B 000
clk
B 0
sum
B XXXX
sum_reg
B XXXX
000
XXXX
(c)
80.0 ns
240.0 ns
U 0
U X
(d)
(e)

**Figure B.6**

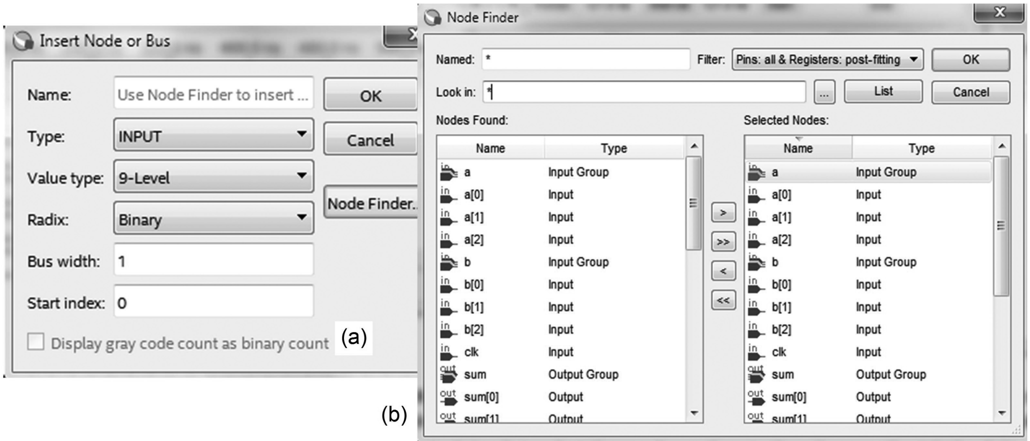



**Figure B.7**

e) Note that the default radix for all signals is binary (denoted by the B letter in the **Value** column of figure B.6b). Change the radix of all signals except *clk* to unsigned decimal. To do so, select the signals, click the right mouse button, and select **Radix** > **Unsigned Decimal**.

f) Change the time range by selecting **Edit** > **Set End Time** and entering 320 ns, which leads to the wave pane of figure B.6c.

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

h) Now we must draw the input waveforms for *clk*, *a*, and *b*, after which the simulator will compute and draw the output waveforms (*sum* and *sum_reg*). The stimuli of figure B.1c 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.

- Select the portion of *a* between 80 ns and 320 ns, then click the arbitrary value icon and enter 5.

- Repeat the process above for *b* by selecting the respective time intervals and entering 2, 4, and 7.

The result is shown in figure B.6d and is now ready for simulation.

i) Save the file with the extension *.vwf* (vector waveform file). You can use the default name (*waveform.vwf*).

j) Finally, simulate the circuit by clicking for functional simulation or for timing simulation (note that the latter is the simulation shown in figure B.6e). Equivalently, select **Simulation** > **Run Functional Simulation** or **Simulation** > **Run Timing Simulation**.

k) Inspect the results and compare them to those in figure B.1c.

## 5. Making Pin Assignments

*Notes:*

1) Pin assignments are only allowed if a specific device was selected in part B.2(f).
2) Pin assignments can be deleted with **Assignments > Remove Assignments**.

a) Select **Assignments > Pin Planner** or click , which opens the window of figure B.8.

b) In each line of the **Location** column, enter the desired pin number (for example, L2 for pin PIN_L2, as shown in the figure for signal *a*(2)).

c) When done, recompile the code.

## 6. Programming the FPGA

*Note:* Two files for programming the FPGA can be produced during compilation. That with extension *.pof* (programmer object file) is stored in an external nonvolatile memory from which the FPGA automatically retrieves the program (for self-programming) at power up. The other, with extension *.sof* (SRAM object file), programs the FPGA directly, so the configuration is not recovered when the power is turned off. The latter should be used in the experiments.

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

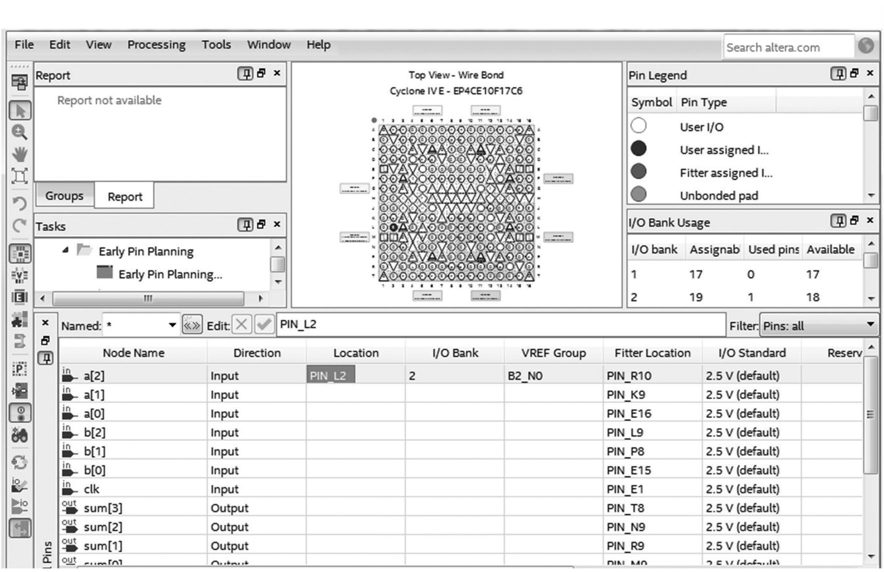



**Figure B.8**

*Note:* If it is the first time that you are using that board, the USB-Blaster Driver must be installed. This generally occurs automatically when the board is powered up.

b) Click the Programmer icon or select **Tools > Programmer**, which opens the window of figure B.9.

c) Observe the following in figure B.9: The hardware driver is **USB-Blaster**; the mode is **JTAG**; the programmer file is *registered_adder.sof* (if the file does not show up, click **Add File** and select it in the *output_files* subdirectory); and the **Program/Configure** box is checked.

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

## 7. Interpreting 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)

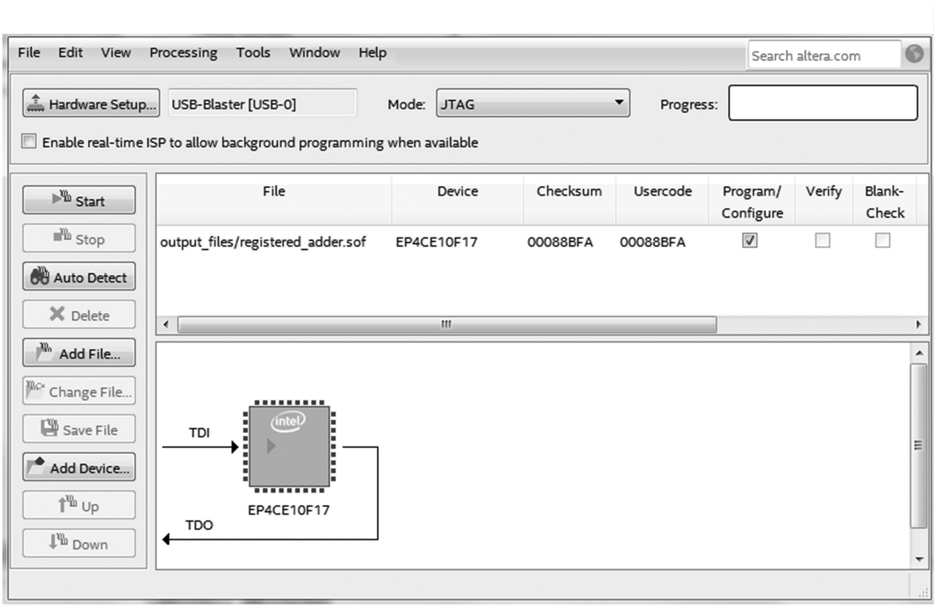



**Figure B.9**

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 a reset signal is called *reset* when it is *asynchronous* and it is called *clear* when it is *synchronous*.