Exercise Manual

for

Designing with Quartus II

Exercise 1

Exercise 1

Objectives:

- *Use the MegaWizard Plug-in Manager and create a multiplier*
- *Insert wires and pins into design*
- *Verify that all connections are made correctly.*

Pipelined Multiplier Design

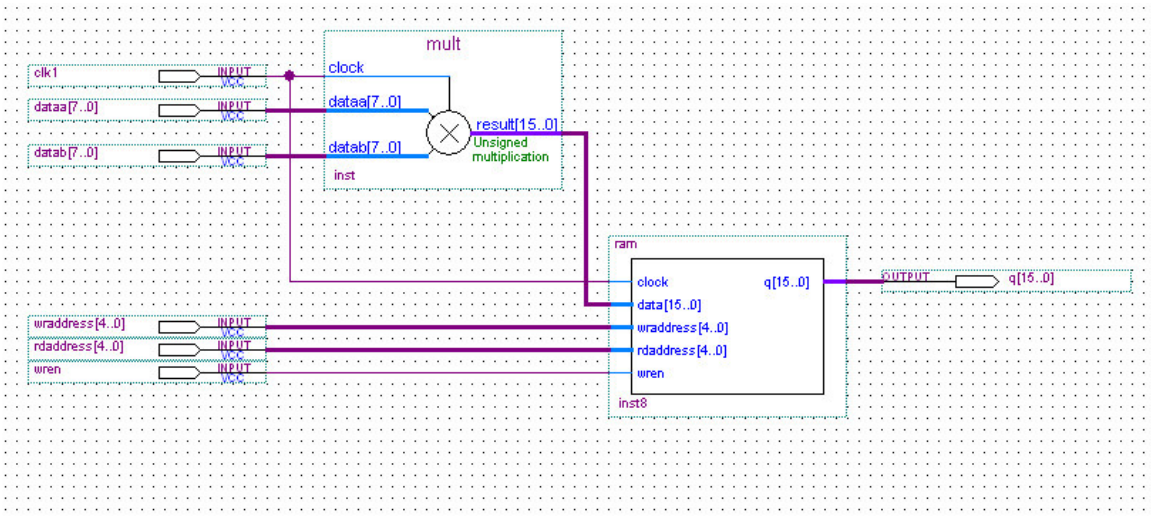


Figure 1

Step 1 (Open the project and create schematic file)

In this exercise, you will use a project which has been created for you. This exercise focuses on the schematic design capabilities in Quartus® II.

1. If it is not already started, launch the **Quartus II** software using the start menu or desktop icon if one exists on your machine.
2. Select **File ⇒ Open project…** Browse to **<Quartus II\Lab install directory>\QII4_0\Lab1\.** Select the project **pipemult.qpf** and click on **Open**.
3. Select **File ⇒ New** and select **Block Diagram/Schematic File**
4. Select **File ⇒ Save As** and save the file as

   <Quartus II\Lab install directory>\QII4_0\Lab1\ pipemult

   We will now create the design shown in Figure 1.

Step 2 (Build an 8x8 multiplier using the MegaWizard® Plug-in Manager)

1. Choose **Tools ⇒ MegaWizard Plug-In Manager**. In the window that appears, select **Create a new custom megafunction variation**. Click on **Next**.
2. On **page 2a** of the **MegaWizard**, expand the **arithmetic** folder and select **LPM_MULT**.
3. From the drop down menu select **Stratix II** for the device family. Choose **Verilog HDL** output.

   Name the **output file <Quartus II\Lab install directory>\QII4_0\Lab1\mult**.

   Click on **Next**.
4. On **page 3**, set the width of the **dataa** and **datab** buses to **8** bits. For the remaining settings in this window, use the defaults that appear. Select **Next**.
5. On **page 4**, choose **Use default implementation** under **"Which Multiplier Implementation should be used?"** Select **Next.**
6. On **page 5**, choose **Yes, I want an output latency of 2 clock cycles**. Click **Next**.
7. On **page 6**, the following check boxes should be enabled to generate output files:

   mult.v

   mult.bsf
8. Select **Finish** in the final window that appears. *The multiplier is built.*

Note that the steps you just went through could be applied to any design entry method. One possible modification you might try is to choose the same language output as your design modules. So, if you are using VHDL modules, a VHDL LPM_MULT output can be chosen.

9. In the **Graphic Editor**, **double-click** in the screen so that the **Symbol** Window appears. Inside the symbol window, **click** on  to expand the symbols defined in the **Project** folder. **Double-click** on **mult**. **Click the left mouse button** to put down the symbol inside the schematic file.

   *The symbol for **"mult"** now appears in the schematic.*

Step 3 (Create symbol for 32 word deep RAM written in VHDL)

1. From the **File** menu, **open** the file **ram.vhd**.

   *Notice this is a **VHDL** file inferring a single-port 32-bit synchronous RAM. This file could very well be written in **Verilog**. This RAM block could also have been easily created using the **MegaWizard**, selecting **LPM_RAM_DP+** in the **storage** folder or the **RAM: 2-PORT** in the **memory compiler** folder.*

2. From the **File** menu, go the **Create/Update** menu option and select **Create Symbol Files for Current File**. Click **Yes** to save changes to **pipemult.bdf**.

   ***Quartus II** now generates a symbol file based on the port declarations in the VHDL file.*

3. Once **Quartus II** is finished creating the symbol, click **OK**. Close the **ram.vhd** file.

4. In the **Graphic Editor**, **double-click** in the screen so that the **Symbol** Window appears again. **Double-click** on **ram** in the **Project** folder. **Click OK.** Move the mouse to the appropriate location to connect the RAM as shown in the figure. Click the left mouse button to put down the symbol.

   *The symbol for **"ram"** now appears in the schematic.*

Step 4 (Add Pins to the Design)

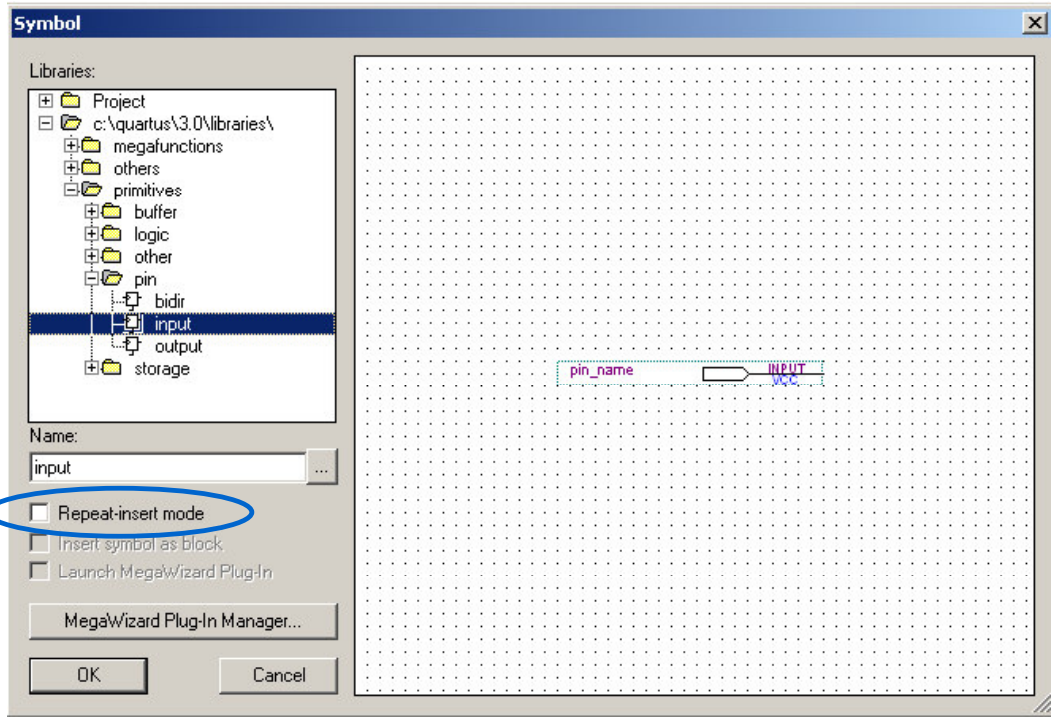
Table 1. Pin List

Input	Output
clk1	q[15..0]
dataa[7..0]	
datab[7..0]	
wraddress[4..0]	
wren	
rdaddress[4..0]	

For each of the pins listed in Table 1, you must insert a pin and change its name.

1. To place pins in the schematic file, go to **Edit ⇒ Insert ⇒ Symbol** or **double-click** in the **Graphic Editor**.
2. Browse to **libraries ⇒ primitives ⇒ pin** folder. **Double-click** on **input** or **output.**

   *Hint: To insert multiple pins select **Repeat Insert Mode**.*



3. To rename the pins **double-click on the pin name after it has been inserted.**
4. Type the name in the **Pin name(s)** field and **Click Ok**.

Step 5 (Connect the Pins and Blocks in the Schematic)

1. In the left hand tool bar click on  to draw a wire and  to draw a bus.

   If you place the cursor next to any symbol's port, the wire or bus tool will automatically appear.

2. Connect all of the pins and blocks as shown in **Figure 1**.

Step 6 (Save and check the schematic)

1. Click on the **Save** button in the toolbar  to save the schematic.

2. From the **Processing** menu select **Start** and then **Analysis & Synthesis button**  .

   Analysis and synthesis checks that all the design files are present and connections are made correctly and then synthesizes the design.

3. Click **OK** when analysis and synthesis is completed.

Step 7 (Open the RTL Viewer)

1. If you have any extra time before the start of the next section, open the **RTL Viewer** and explore the Quartus II interpretation of the design you created.

Exercise Summary

- Completed a schematic design in Quartus II
- Used the MegaWizard to create a multiplier
- Generated a symbol from previously created HDL file to incorporate into a schematic
- Used Analysis and Synthesis to check design files

END OF EXERCISE 1

Exercise 2

Exercise 2

Objectives:

- *Use an existing top-level file to set up a project*
- *Analyze various timing parameters in the design*
- *Control the use of DSP Blocks for implementing the multiplier*
- *Assign I/O Pins and Run I/O Analysis*

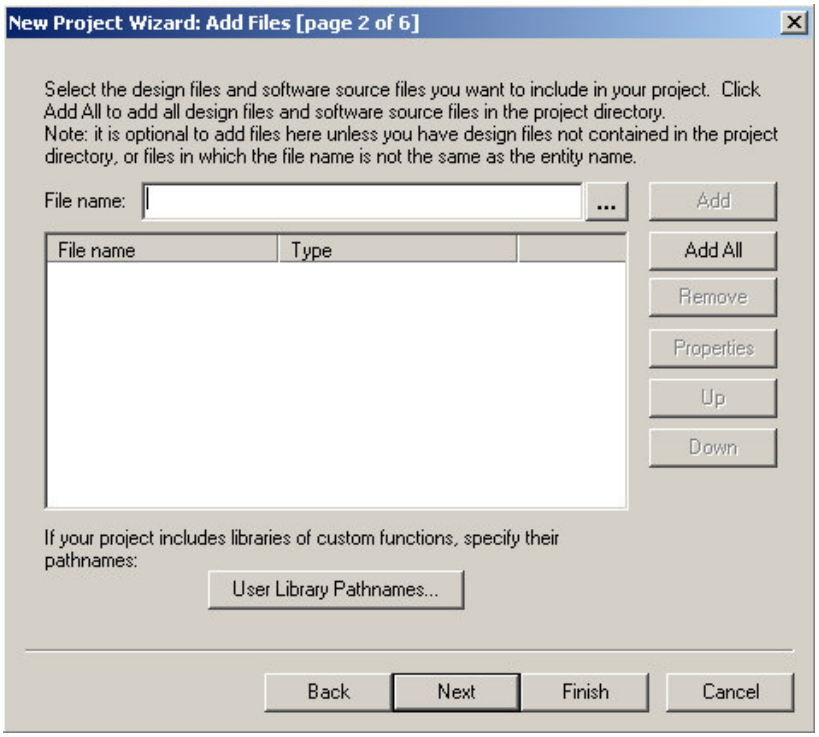
	With DSP Blocks	Without DSP Blocks
Device Name		
Total Logic Elements		
Total DSP Blocks Elements		
Total Memory Bits		
Worst Fmax		

Step 1 (Setup Project for QII4_0\Lab2)

1. Under **File**, Select **New Project Wizard…**. A new window appears. If an **Introduction** screen appears, click **Next**.
2. Page 1 of the wizard should be completed with the following:

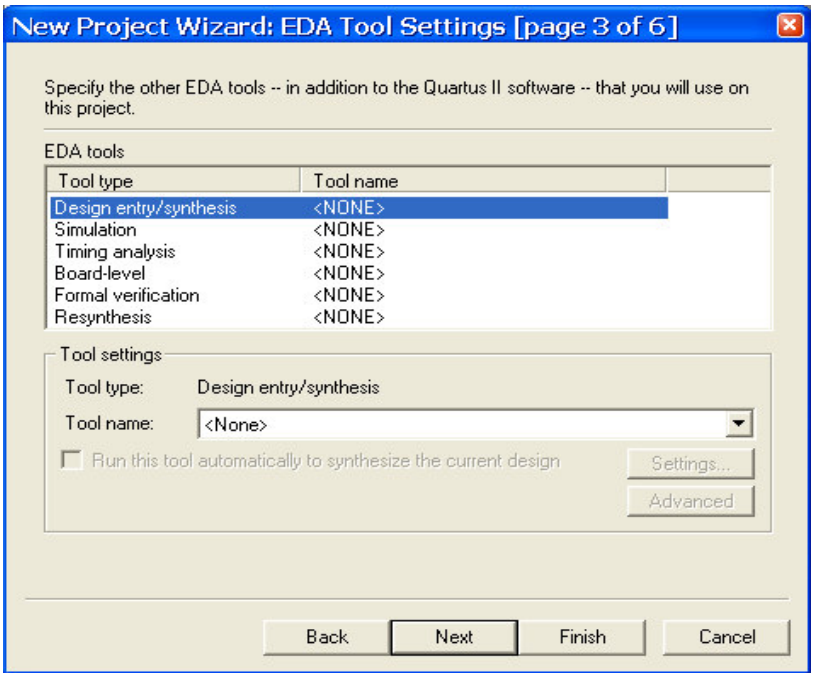
working directory for this project	**<Quartus II lab install directory>\QII4_0\Lab2_4**
name of project	**pipemult**
top-level design entity	**pipemult**

3. **Click Next** to advance to the **Project Wizard: Add Files [page 2 of 6]**.

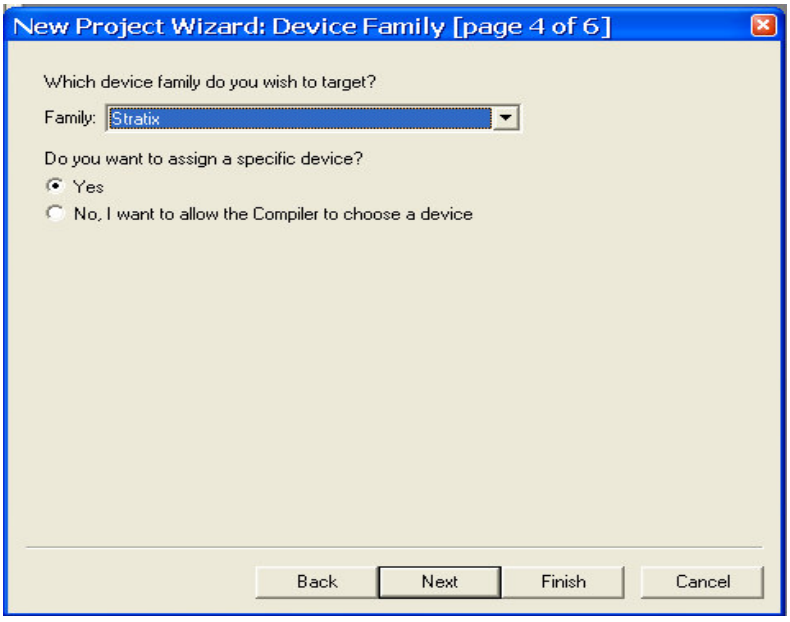


4. All of the design files are in the current project directory. As shown above, leave everything blank and **Click Next**.

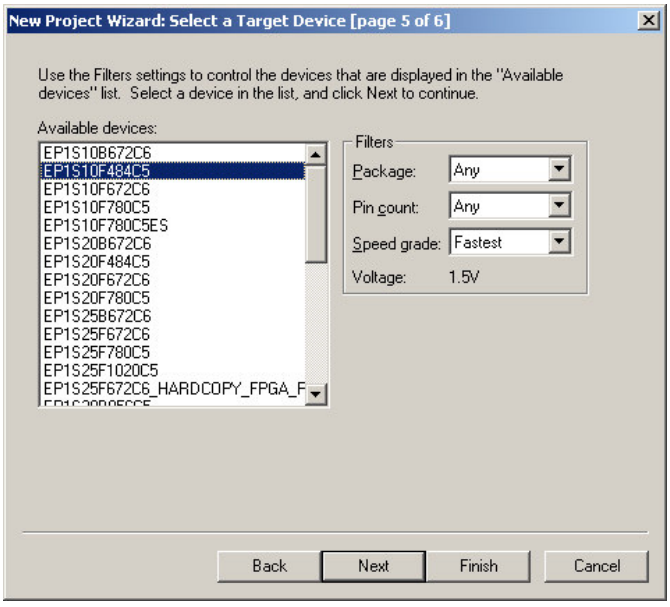
   Note: Files located in the project directory do not need to be added to this list. This screen is used for adding design files located in different locations.



5. Page 3 allows you to specify any third party EDA tools. These exercises will be done entirely within Quartus II. Click **Next.**

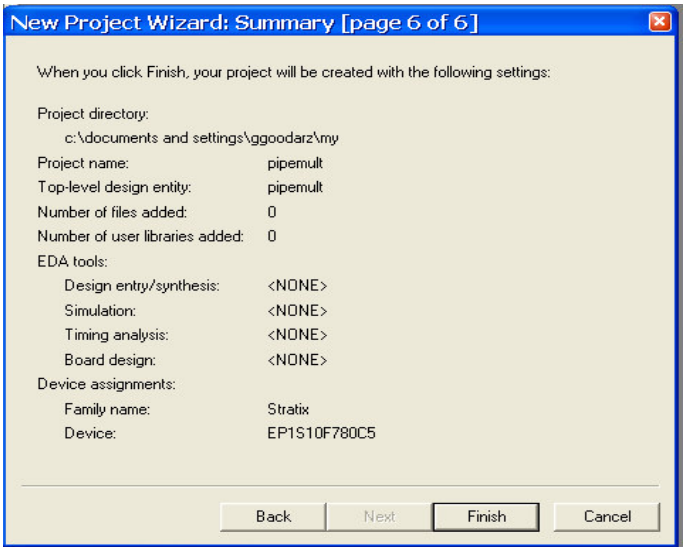


6. On page 4 (as shown above), select **Stratix** for the Family and click **Yes** for specifying the device. Click **Next.**



7. On **page 5**, make sure all of the Filters on the right side are set to **Any**. Scroll down and select the **EP1S10F484C5** device. Click **Next**.

   *The device may already be selected since **Quartus II** copies the device selected in the last open project for the new project being created.*



8. The summary screen appears as shown. **Click Finish.** The project is now created.

Step 2 (Compile the design)

1. Select **Start Compilation** from the **Processing** menu or **click** on  located on the toolbar to perform a full compilation of the design. A dialog box will appear when the compilation is complete.

2. Click **OK**.

Step 3 (Gather information from the Compilation Report File)

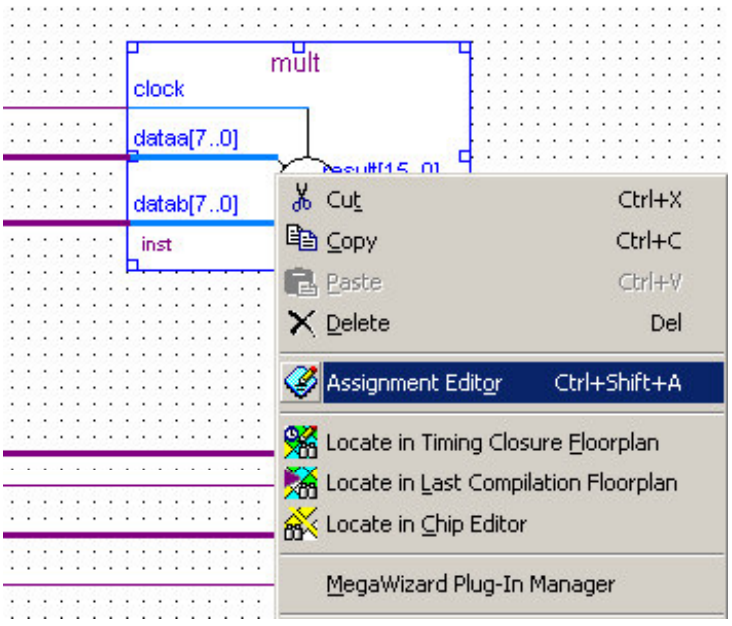
*By default, **Quartus II** opens the **Compilation Report** file and has the Summary Section selected. If this is not open, click on*  .

1. From the **Summary** section of the **Compilation Report**, record the **Device** name, **Total logic elements**, **Total memory bits**, and **DSP Block 9-bit elements** in the table above at the beginning of this exercise.
2. Expand the **Timing Analyzer** folder in the **Compilation Report**. Click on **Clock Setup: 'clk1'** table under the **Timing Analyzer** folder. In the exercise table, record the worst **Actual fmax** in the clock table.

Step 4 (Implement the multiplier in logic elements instead of a DSP Block)

***Stratix** and **Stratix II DSP Blocks** are a valuable resource for implementing multiply, multiply-add, and multiply-accumulate functions in the FPGA. They provide a better usage of resources for multiplication over logic elements. But, DSP Blocks are limited in number. If you design has many multipliers, it may be advantageous to implement smaller or non-speed critical multipliers in logic elements instead. This can be done using an option in the MegaWizard flow, or it can be done on a multiplier-by-multiplier basis using the Assignment Editor logic option.*

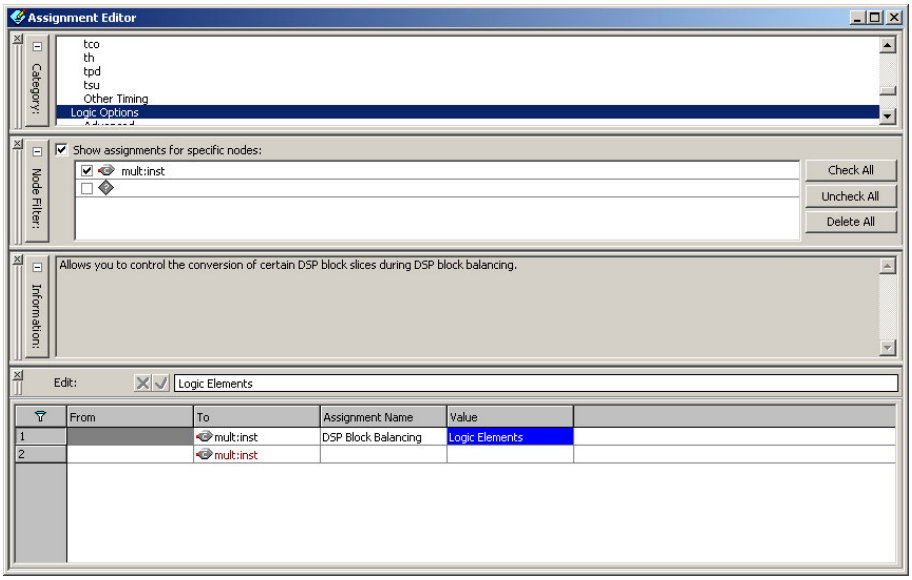
1. Bring **pipemult.bdf** to the foreground.
2. **Right mouse click** on the **mult** symbol and choose **Assignment Editor**.



3. In the **Assignment Editor** expand the **Category** section (at the top) and select **Logic Options**.

4. In the **Node Filter** section, make sure **Show assignments for specific nodes** is checked.
5. **Right-click** in the **Assignment Name** field for **mult:inst** (row 1) and select **Edit Cell**.
6. Select **DSP Block Balancing** from the drop down menu.
7. **Right-click** in the **Value** field for the **DSP Block Balancing** option and select **Edit Cell**.
8. Select **Logic Elements** from the drop down menu.
9. Save the **Assignment Editor** file.

*Your **Assignment Editor** window should look similar to below.*



Step 5 (Recompile the design)

1. Click on  to compile the design.

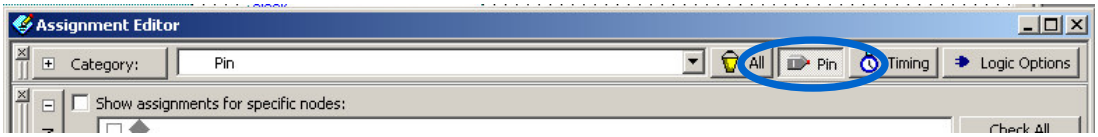
Step 6 (Determine the performance of the design)

4. From the **Summary** page inside the **Compilation Report**, record the **Device name**, **Total logic elements**, **Total memory bits**, and **DSP Block 9-bit elements** in the exercise table above.
5. Expand the **Timing Analyzer** folder in the **Compilation Report**. Click on the **Clock Setup: 'clk1'** table. In the exercise table above, record the worst fmax.

It is pretty clear the DSP Blocks can greatly improve timing, but as with many options there is a trade-off with regards to resource usage.

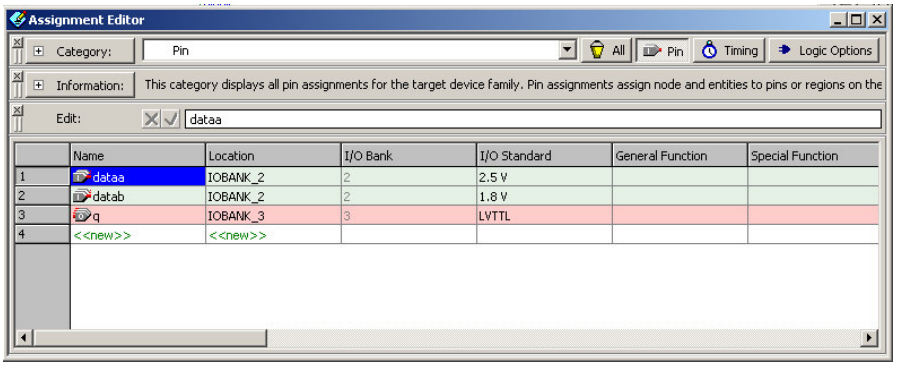
Step 7 (Use Assignment Editor to make general pin placement assignments)

1. Bring the **Assignment Editor** to the foreground.
2. In the **Assignment Editor** toolbar, make sure the **Category Bar** and **Show I/O Banks in Color** buttons are turned on. Under the **Node Filter**, uncheck the box to **Show assignments for specific nodes**.



3. **Collapse** the **Category** section of the **Assignment Editor** and then click on the **Pin** button as shown above.
4. In the **Assignment Editor**, double-click on the **To** cell in row 1 to reveal a drop-down menu of available pins. Select the entire **dataa** input bus (dataa).
5. In the **Location** cell of row 1, double-click and assign the **dataa** bus to **I/O Bank 2**.
6. In the **I/O Standard** cell of row 1, double-click to assign the **dataa** bus as **2.5 V**.
7. Double-click on the **To** cell of row 2 and select the entire **datab** input bus (datab).
8. Assign the **datab** bus to **I/O Bank 2** by double-clicking on the **Location** cell of row 2.
9. Set the value for the **datab** bus to **1.8 V** by double-clicking on the **I/O Standard** cell of row 2.
10. Double-click on the **To** cell of row 3 and select the **q** output bus.
11. Assign the **q** output bus to **I/O Bank 3**.

    *Your **Assignment Editor** window should look as shown below. Notice that the **q** output bus is shown in a different color to indicate it is being assigned to a different I/O Bank than the **dataa** and **datab** busses.*



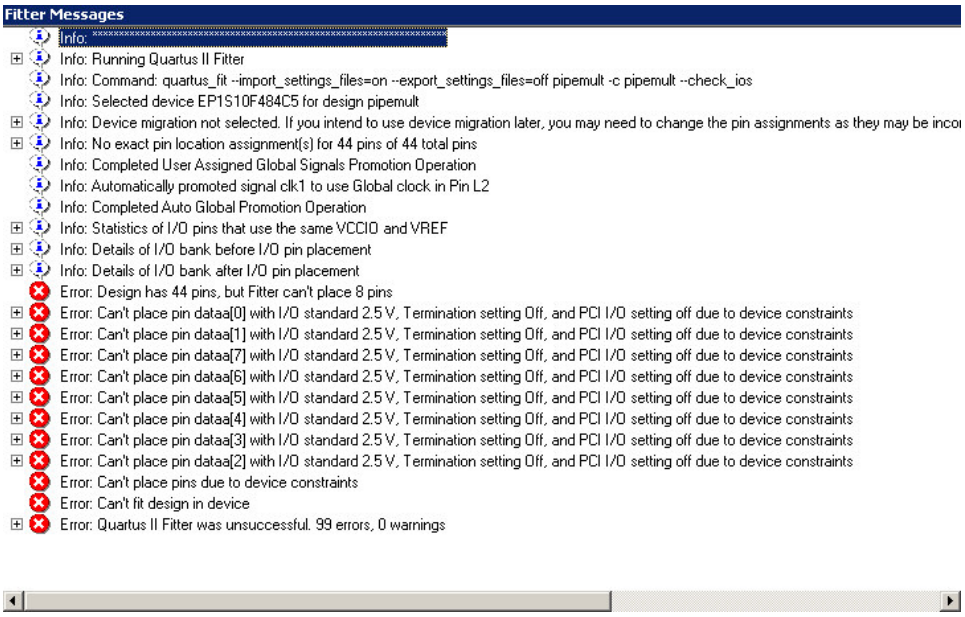
12. **Save** the **Assignment Editor** file.

Step 8 (Analyze I/O assignments)

Now you have made simple I/O assignments, you can check the validity of those assignments without running a full compilation. This way you can quickly and easily find I/O placement issues and correct them.

1. From the **Processing** menu, go to **Start** and select **Start I/O Assignment Analysis**. Click **OK** once the analysis is complete.

   *Was the analysis successful? Check the messages in the **Message** window or the **Fitter Messages** in the **Compilation Report**. They should as show below.*



2. Review the **I/O Analysis Messages** and determine the cause of the error. Expand the error messages to get more detail as to why each pin of the **dataa** bus is not being placed successfully.

   *Determining the cause of the I/O placement failure requires reading the error messages carefully and having a little understanding of **Stratix** or general FPGA I/O blocks. See if you can understand and correct the cause of the errors on your own. If you do not have a lot of **Stratix** or FPGA experience, then #'s 3-5 will show you how correct the errors.*

3. Bring the **Assignment Editor** to the foreground again.

   *Notice that you have assigned **dataa** and **datab** input busses to **I/O Bank 2**, but set different **VCCIO (1.8 & 2.5)** voltage levels for them. **Stratix**, like all Altera FPGA families, allow for only one VCCIO per I/O bank.*

4. Double-click on the **I/O Standard** cell for the **datab** input bus. Change the setting setting to **2.5 V**.

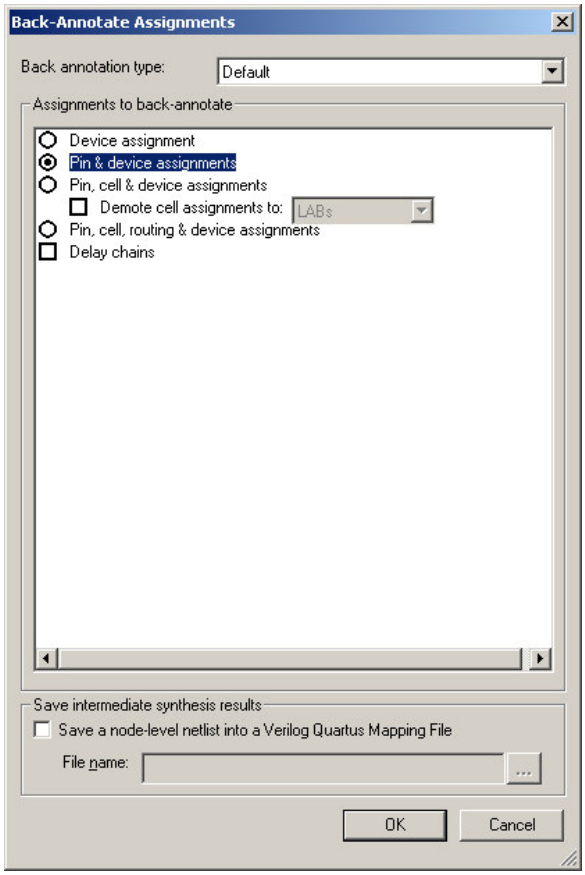
5. **Save** the **Assignment Editor** and re-run the **I/O Assignment Analysis**. Click **OK** when complete.

See how quickly and easily you can check you I/O placement assignments without running a full compilation!

Step 9 (Back-annotate pin assignments to lock placement)

*This is the step you would use once you have produced a verified pin-out to begin board design. Now you need to make sure that **Quartus II** does not move the pin locations during successive compilations.*

1. From the **Assignments** menu, select **Back-Annotate Assignments** to open the **Back-Annotate Assigments** dialog box.
2. In the **Assignments to back-annotate window**, enable **Pin & device assignments** as shown below. Click **OK**.



3. Bring the **Assignment Editor** to the foreground. Click on the **Pin Category** button again.

   Notice this time that to all I/O bank assignments have been added specific I/O pin locations. These are the locations carried from the last I/O analysis performed by the fitter. Now at this point, you have a verified pin-out to begin board design.

Exercise Summary

- Created a project
- Compiled and gathered information from the compilation report
- Used the Assignment Editor to control logic options
- Used I/O analysis to quickly verify I/O placement

END OF EXERCISE 2

Exercise 3

Exercise 3

Objectives:

- *Analyze a design for timing*

Please summarize your results in the tables as you complete the exercise:

Fmax

Worst Fmax	
Worst Fmax's Clock Period	
Register to Register Delay	
Smallest Clock Skew	
Micro Clock to Output	
Micro Setup Delay	
Solve the equation below with the above information: *Clock Period (Worst Fmax) = Register to Register Delay – Smallest Clock Skew* *+Micro Clock to Output + Micro Setup Delay*	

Tsu

Worst tsu	
Longest pin to memory delay	
Micro setup delay	
Shortest Clock Path to Destination Register	
Solve the equation below with the above information: *Setup Time (Worst tsu) = Longest pin to memory delay – Shortest Clock Path to* *Destination Register + Micro Setup Delay*	

Tco

Worst tco	

Step 1 (Recompile the design)

1. **Click** on  to do a full recompilation of the design.

   A full recompilation is necessary to re-run the place and route and to generate timing files.

Step 2 (Gather fmax timing information from the Compilation Report File)

1. Click on **Clock Setup:'clk1'** under the **Timing Analyzer** section of the **Compilation Report**. In the exercise table above, enter the **slowest fmax**.
2. Right mouse click on the slowest fmax and select **List Paths**. Go to the **Message Window** and record the **clock period** in the exercise table.
3. In the **Message** window, click on the plus sign to expand the fmax delay. Record the **Longest register to register delay**, **Smallest clock skew delay**, **Micro setup delay**, and **Micro clock to output delay** in the exercise table. Verify that:

   Clock period = Register-to-Memory – Clock Skew Delay
   + Micro Setup Delay + Micro Clock to Output Delay

4. Click on the plus sign on the **Longest register to register**. Quartus II will list the number of nodes in this register-to-register path.
5. Click on the **longest register-to-register** to hightlight. Now **right-click** and select **Locate**. In the **Floorplan View** will graphically display the critical path along with timing information..

Step 3 (Gather tsu timing information from the Compilation Report File)

1. **Click** on **tsu** under the **Timing Analyzer** section of the **Compilation Report**. A table is generated showing you the tsu for all your input and bidir pins. The **longest tsu** is listed at the very top of the table.
2. In the exercise table, enter the **longest tsu**.
3. Right mouse click on the longest tsu and select **List Paths**. The **Message Window** displays the tsu.
4. In the **Message** window, click on the plus sign to expand the tsu. Record the **Pin to memory delay**, **Micro setup delay**, and **Shortest clock path delay** in the exercise table. Verify that:

   Setup Time = Longest Pin to Register Delay + Micro Setup Delay
   – Shortest Clock Path to Destination Register

5. Click on the **Longest pin to register delay** to highlight. Now **right-click** and select **Locate**. Identify the tsu path in the Floorplan View.

Step 4 (Gather tco Timing information from the Compilation Report File)

1. Click on **tco** under the **Timing Analyzer** section of the Compilation Report File. A table is generated showing you the tco for all your output and bidir pins. The **largest tco** listed at the very top of the table.
2. In the exercise table, enter the **largest tco**.

   This number may seem large, but this is an unconstained clock to output. Much better times can be achieved by adding timing assignments to I/O. This will be discussed in the next section.

Exercise Summary

- Performed single clock timing analysis
- Viewed timing paths information and details
- Located timing information in floorplanner

END OF EXERCISE 3

Exercise 4

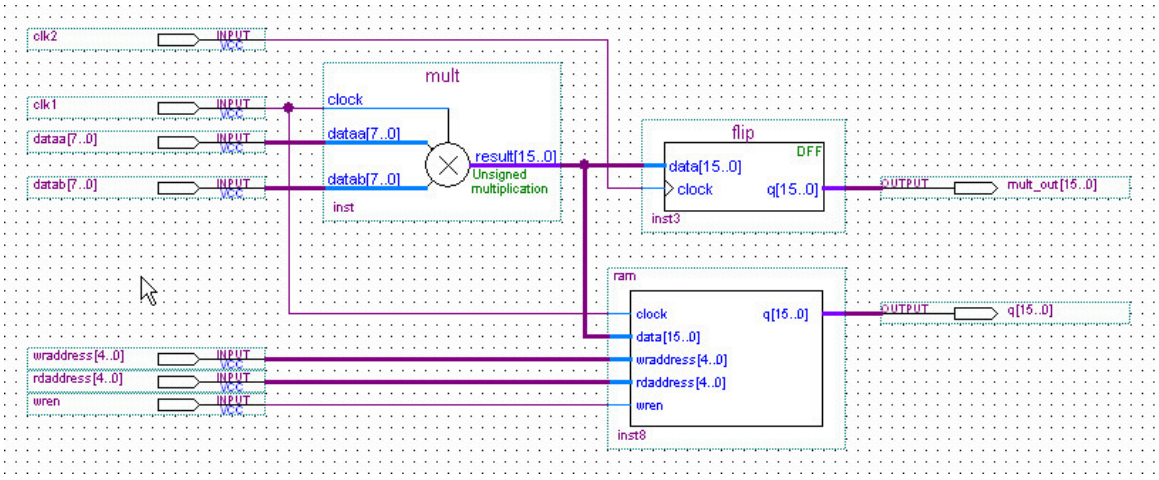
Exercise 4

Objectives:

- *Create clock settings and apply settings to clock pins*
- *Multi-cycle timing*
- *Interpret the Slack Times Table*

In addition, we'll have a better understanding of

- *Required Time*
- *The relationship between Required Time, Actual Time and Slack*



Exercise 4

Please summarize your results in this table as you complete the exercise:

Destination Clock	Required Longest P2P Time (ns)	Actual Longest P2P Time (ns)	Slack(ns)
clk1			
clk2			

What is the relationship between Required Longest P2P Time, Actual Longest P2P Time, and Slack?

_____

Why does this design have two Required Longest P2P Times?

Setting	Destination Clock	Required Setup Relationship
No Multi-cycle Assignment		
Multi-cycle = 2		

Step 1 (Add a 16-bit register and second clock to design)

1. Choose **Tools > MegaWizard Plug-In Manager**. In the window that appears, select **Create a new custom megafunction variation**. Click on **Next**.
2. In the window that appears, expand the **storage** folder and select **LPM_FF**. Choose **VHDL** output. Name the **output file <Quartus II Lab install directory>\QII4_0\Lab2_4_QII\flip**. Click on **Next**.
3. Set the number of the **flipflops** bus to be **16** and the **type of the flipflop** to be **D flipflop** bits. For the remaining settings in this window, use the defaults.
4. Select **Next**.
5. For the next window that appears, simply use the default settings by selecting **Next**.
6. Select **Finish** in the final window that appears.
7. In the **Graphic Editor**, **double-click** in the screen so that the **Symbol** Window appears. Inside the symbol window, expand the **Project** folder. Double-click on **flip**. **Move the mouse** to the appropriate location to connect the multiplier as shown in the figure. **Click the left mouse button** to put down the symbol.

   *The symbol for **"flip"** should now appear in the schematic*.
8. Add an output pin **mult_out[15..0]**, the new input clock pin **clk2** and connect the flip flop as shown in the figure.
9. **Save** the schematic.

Step 2 (Perform analysis and synthesis)

1. From the **Processing** menu select **Start** and click on the **Start Analysis and Synthesis**

   This will extract node names necessary for making timing assignments.

Step 3 (Make clock requirements for clk1 and clk2 assign to the clock nodes)

1. In the **Assignments** menu and choose **Timing Settings**.
2. In the **Clock Settings** section, click on **Settings for individual clock signals**. Click on the **Clocks…** button.
3. In the **Clocks** dialog box, click on the **New** button.
4. In the **New Clock Settings** dialog box, type **clk1** in the **Clock setting name** and **Applies to node** fields in the dialog box.

*Notice the **Node Finder** ... can be invoked to search for internally generated clocks.*

5. In the **Relationship to other clock settings** section, type the **required fmax** for **clk1** as **150 MHz**. Click **OK**.

   *You are now back at the **Clocks** dialog box you see the clock settings for the base clock (**clk1**) that you have just entered. Now, we will enter a clock setting for the second clock, **clk2**, which is derived from the base clock, **clk1**.*

6. In the **Clocks** dialog box, click on **New** again.

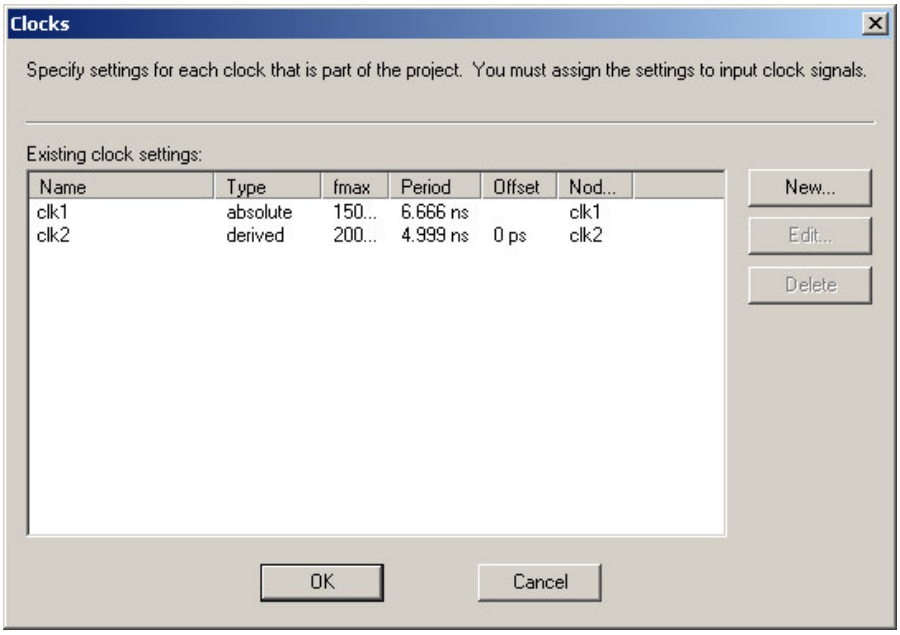
7. Type **clk2** in the **Clock setting name** and **Applies to node** fields in the dialog box.

8. This time, in the **Relationship to other clock settings** section, click **Based on** and ensure **clk1** is specified as the base clock. Click on the **Derived Clock Requirements…** button.

9. In the **Derived Clock Requirements** dialog box, specify that **clk2** must be **a 4/3 ratio** (multiply base clock frequency by 4 and divide the base clock frequency by 3) of the base clock with an offset of **0 ns**. Click **OK**.

10. In the **New Clock Settings** dialog box, click **OK**.

    You should see two clock settings as shown below, one as an absolute and the other as a derived.



11. In the **Clocks** dialog box, click **OK**. Click **OK** one last time to close the **Settings** dialog box.

Step 4 (Check the clock settings in the Assignment Editor)

The ***Timing Wizard*** *creates the clock settings and has automatically assigns them to the clock nodes in the* ***Assignment Editor****. To verify:*

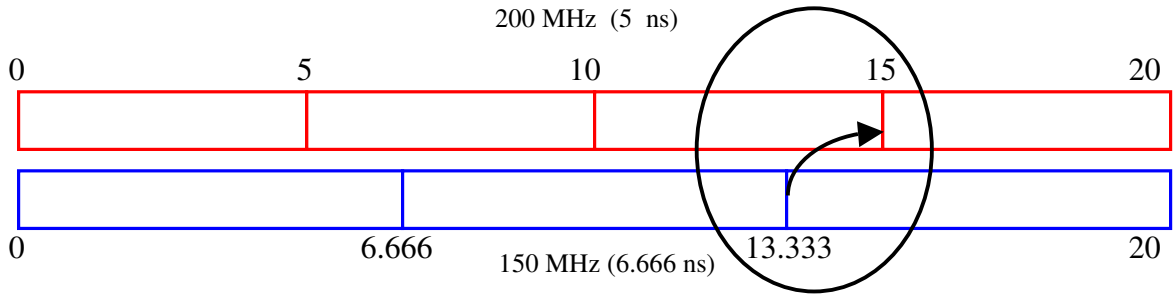
1. Bring the **Assignment Editor** to the foreground.
2. In the **Category** section, select **Timing**.

   Here you should see the two clock settings listed and being applied to input pins clk1 and clk2.

Step 5 (Analyze timing results)

1. **Compile** the design  **.**
2. Look at the clock tables for **clk1** and **clk2** in the **Timing Analyzer** section.
3. Look over the **Required Longest P2P Times** listed in the table. For the **worst case signals** of each clock, **Record** the **Required Longest P2P Time, Actual Longest P2P Time,** and **Slack** in the exercise table.
4. What is the relationship between **Required Longest P2P Time**, **Actual Longest P2P Time**, and **Slack**? Please answer this question below the exercise table.
5. Can you explain why this design has two (2) **Required P2P Times**? Please answer this question below the exercise table. Hint: Look at the design again.
6. **Record** the **Required Setup Relationship** for **clk2** in the second exercise table.

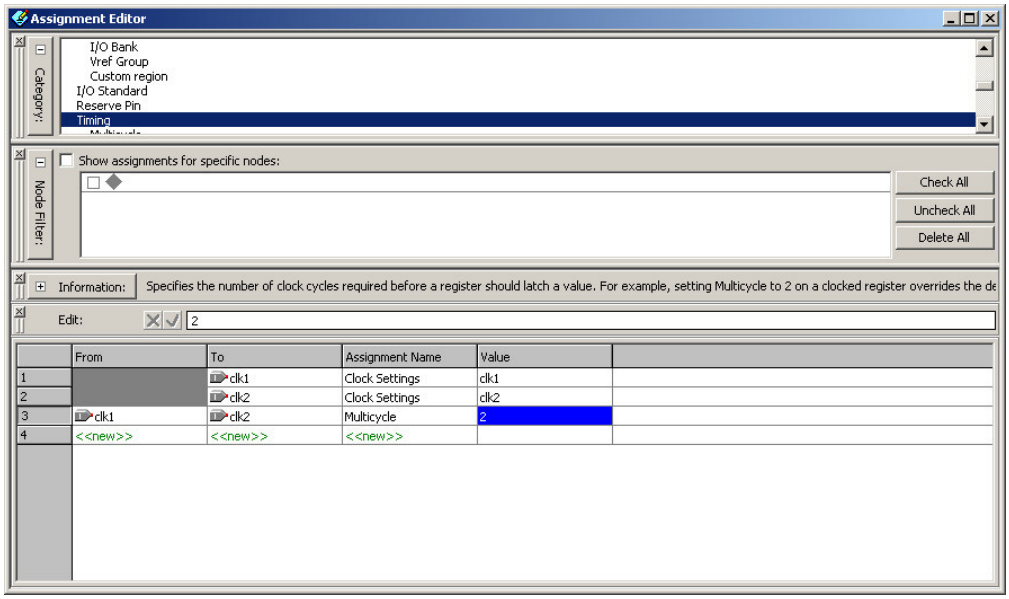
   The ***Required Setup Relationship*** *corresponds to the shortest setup time from* ***clk1*** *to* ***clk2,*** *based upon the Clock Settings ratio, as circled in the diagram below.*



The ***Required Longest P2P Time*** *subtracts the source register's clock-to-output time and the destination register's setup time from the* ***Required Setup Relationship****. Analyzing any* ***clk1*** *path (****List Paths****) will show this. Notice that for the* ***clk2*** *requirement, the values are in red to indicate that all of the registers driven by* ***clk2*** *will incur setup violations and the circuit will fail at this time.*

Step 6 (Make multi-cycle assignment)

1. Bring the **Assignment Editor** to the foreground.
2. Select **Timing** in the **category** section and click on <<new>> in the **To** field. Right click and select **Node Finder**.
3. In the **Node Finder** dialog box make sure that **Pins:all** has been selected in Filter field. Click on **List**.
4. **Double-click** on **clk2** from under ***Nodes Found:*** to add it to the ***Selected Nodes:*** window. Click **OK**.
5. Under the **From** field, right-click and select **Node Finder**.
6. In the **Node Finder** dialog box make sure that **Pins:all** has been selected in Filter field Click on **List**.
7. **Double-click** on **clk1** from under ***Nodes Found:*** to add it to the ***Selected Nodes:*** window. Click **OK**.
8. In the **Options** field of the **Assignment Editor** right-click and select **Edit Cell**. From the drop down menu select **Multicycle**
9. In the **Value** field right click and select **Edit Cell** and enter **2**



10. **Save** the **Assignment Editor** settings.
11. Re-compile the design by clicking the compile button  .

Step 7 (Analyze timing results)

1. After compilation has finished, expand the **Timing Analyzer** section. Select **Clock Setup: 'clk2'** and record the **Required Setup Relationship** in the second table.

   *Notice that one whole destination clock period has been added to the **Required Setup Relationship**. Remember that this design should already need to be accounting for the clock domain transfers using synchronizers, hand-shaking or some other control logic. Creating a multi-cycle assignment merely tells the timing analyzer what you have already accounted for in logic.*

   So, were you yet able to meet timing? No? Notice the amount of slack by which you are missing your timing. Now you will try another Quartus II option to improve your results.

Step 8 (Enable physical synthesis)

1. From the **Assignments** menu, select **Settings**. Go to **Physical Synthesis Optimizations** found in the **Fitter Settings** options.
2. In the **Fitter optimizations** box, enable **Perform register retiming**. Click **OK**.

   This feature will now to try to balance the combinatorial logic between registers to improve the system performance.

3. Re-compile the design again by clicking the compile button  .
4. Review the slack analysis tables for **clk1** & **clk2** to see if timing has now been met.

   How was timing able to be met now? How is that possible? Think of the multiplier and where the pipeline registers should be for maximum performance.

Exercise Summary

- Enabled multi-clock timing analysis
  - Slack analysis
- Used the Timing Wizard to define clocks
- Made Multi-cycle timing assignments
- Enabled register retiming (physical synthesis)

END OF EXERCISE 4

Exercise 5

Exercise 5

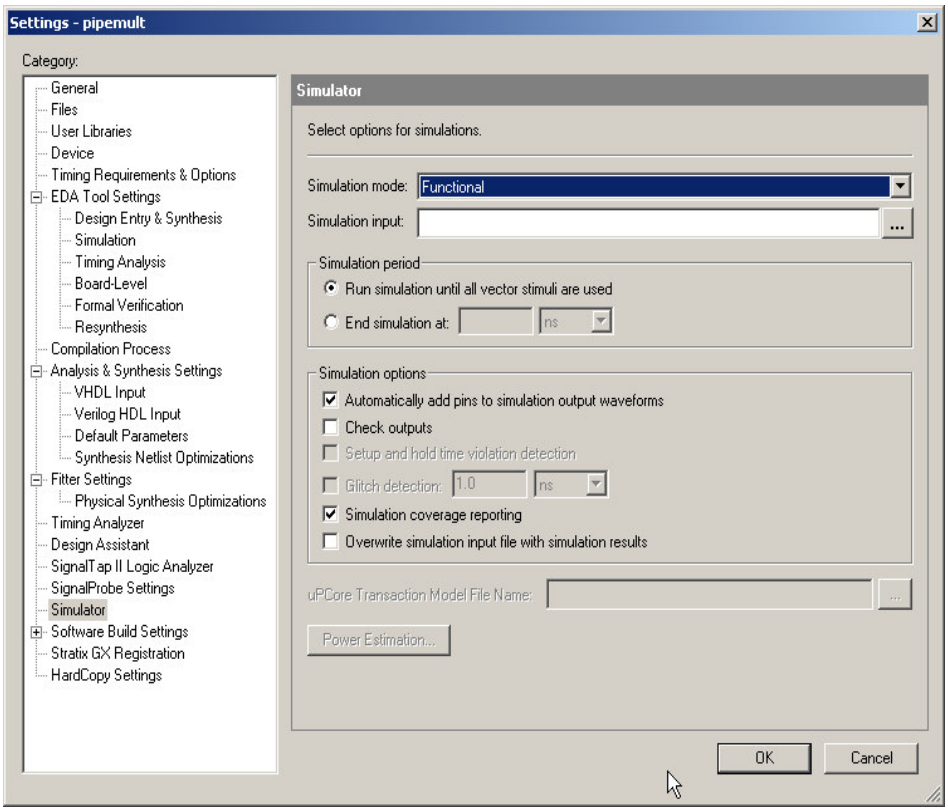
Objectives:

- Create a Vector Waveform (.vwf) file input stimulus file
- Run a functional simulation
- Examine the simulation output for verification of the design

Exercise 5

Step 1 (Set up Simulator)

1. **Open** the project **<*Quartus II lab install*>\QII4_0\Lab5\pipemult.qpf**.
2. From the **Assignments** menu select **Settings.**
3. In the **Simulator** category, click on the **Simulator mode**. Select **Functional** from the drop-down menu.
4. For simulation input, type **pipemult.vwf**. Click **OK**.



5. From the **Processing menu** select **Generate Functional Simulation Netlist**.

Step 2 (Create .vwf file)

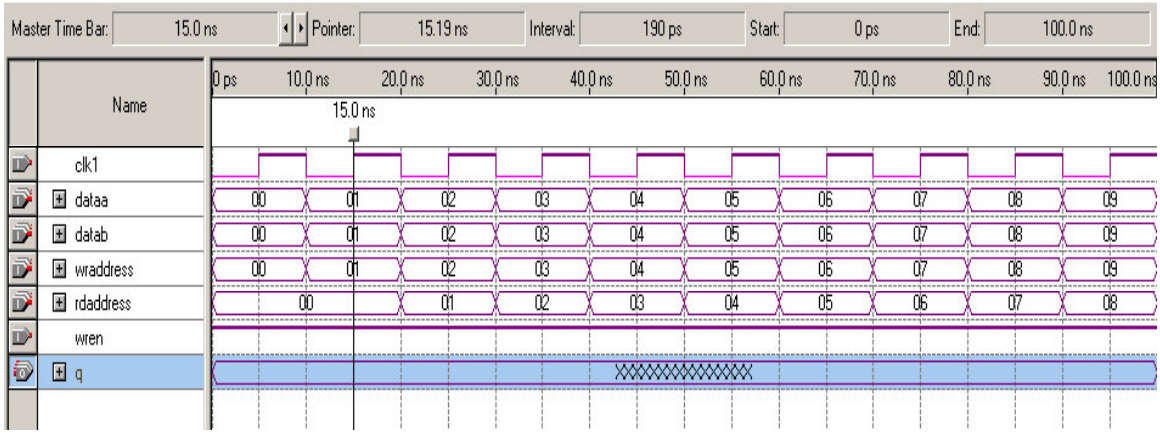
1. From the **File** menu select **New**. From the **New** dialog box select the **Other Files** tab. In the **Other Files** tab, select **Vector Waveform File** and click on **OK**.

Step 3 (Enter Signals)

1. From the **Edit** menu select **Insert Node or Bus**. In the **Insert Node or Bus** dialog box click on the **Node Finder** Button.
2. In the **Node Finder**, go to the **Filter** box and select **Pins: all** from the drop down menu. Now click on the **List** button.
3. Select **clk1, dataa, datab, wraddress, rdaddress**s, wren, and q. Click on the **>** button to copy these pins to the Selected Nodes area. Click **OK**. Click **OK** again in the **Insert Node or Bus** dialog box.
4. In the .vwf file, highlight **dataa**, **datab**, **wraddress**, **rdaddress** and **q**. Right-click and select **Properties**. Change the radix from **Binary** to **Hexadecimal**. Click **OK**.
5. From the **Edit** menu, set the **End Time** to **100 ns**.

Step 4 (Set Stimulus)

1. The .vwf file should contain all the nodes you have selected.
2. **Enter the input waveforms for clk1, dataa, datab, wraddress, rdaddress and wren as shown below.**

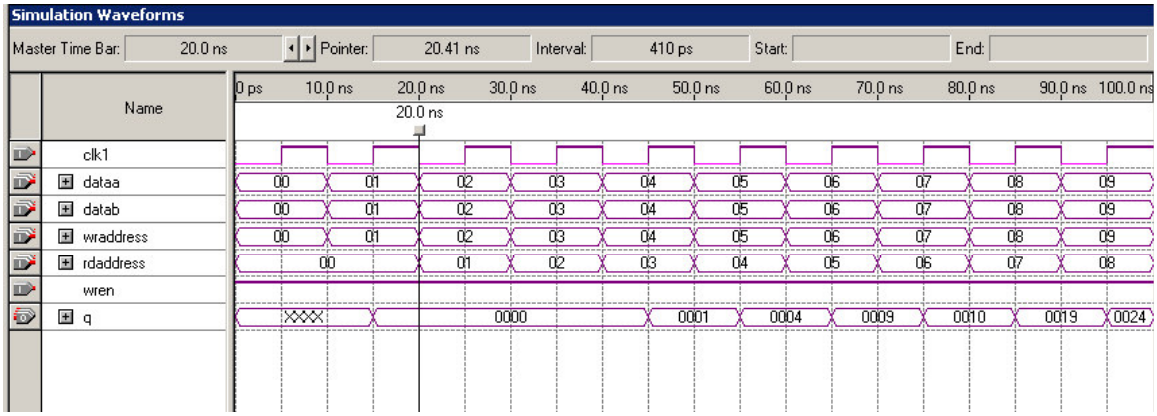


Step 5 (Save and simulate stimulus file)

1. **Save** simulation file as **pipemult.vwf**.
2. From the **Processing** menu select **Start Simulation**  .
3. Once simulation is complete, a box appears “Simulation was Successful”. Click **OK**.

Step 6 (Examine result)

1. The **Simulation Report** automatically opens when simulation begins.
2. The **Simulation Waveform** in the simulator report should match the image below.



Exercise Summary

- Created a .VWF file
- Performed functional simulation
- Used simulation report to view simulation results

END OF EXERCISE 5

Exercise 6

(Optional Cubic Cyclonium Programming Lab)

Exercise 6

Objectives:

- Create a chain description file (CDF) to use in programming a JTAG chain
- Use Programmer tool to configure a device

Exercise 6

This exercise is a quick demonstration of how to set up the programmer and program a device.

Step 1 (Connect Cubic Cyclonium Board & Open Quartus II Project)

1. Take out your **Cubic Cyclonium** board and connect it to the **USB** port of your PC.

   *This will power the board and cause the default hardware and software images to be loaded. These images interface with the **Cubic Cyclonium** application loaded on your PC. But before you get to experiment with that, you will load a simple counter program to demonstrate the **Quartus II** programmer tool.*

   ***Note:** If the Found New Hardware Wizard opens, then please alert your instructor. The USB drivers might not have been loaded on your PC.*

2. **Open** the project **<*Quartus II lab install directory*>\QII4_0\Lab6\counter_cc\counter_100.qpf**.

   This project contains only a simple 2-digit decimal counter.

Step 2 (Open and Set Up Programmer to Configure Device)

1. From the **Tools** menu or toolbar  , select **Programmer**.

   *This will open a blank **CDF** as shown below. The **CDF** file lists all of the devices in your configuration or JTAG chain along with their associated programming or configuration files.*

2. If not already named, save the CDF as **counter_100.cdf**.

3. Locate the **Hardware Setup** button at the top of the **CDF** file. The field to the right of this button should read **Cubic Cyclonium [USB-0]**.

   *If this window reads anything else, click on the **Hardware Setup** button and select **Cubic Cyclonium** from the list of **Available Hardware**. Click on the **Select Hardware** button and then **Close**.*

   *If **Cubic Cyclonium** is not listed in the **Available Hrdware** window, please let the instructor know as the drivers may not have been loaded correctly onto your PC.*

4. Back in the **CDF** file, use the drop-down **Mode** field to choose **JTAG**.

   How does this work? USB control logic has been loaded into the MAX 7000 device. The MAX 7000 device thus allows you to communicate with the JTAG chain via the USB cable.

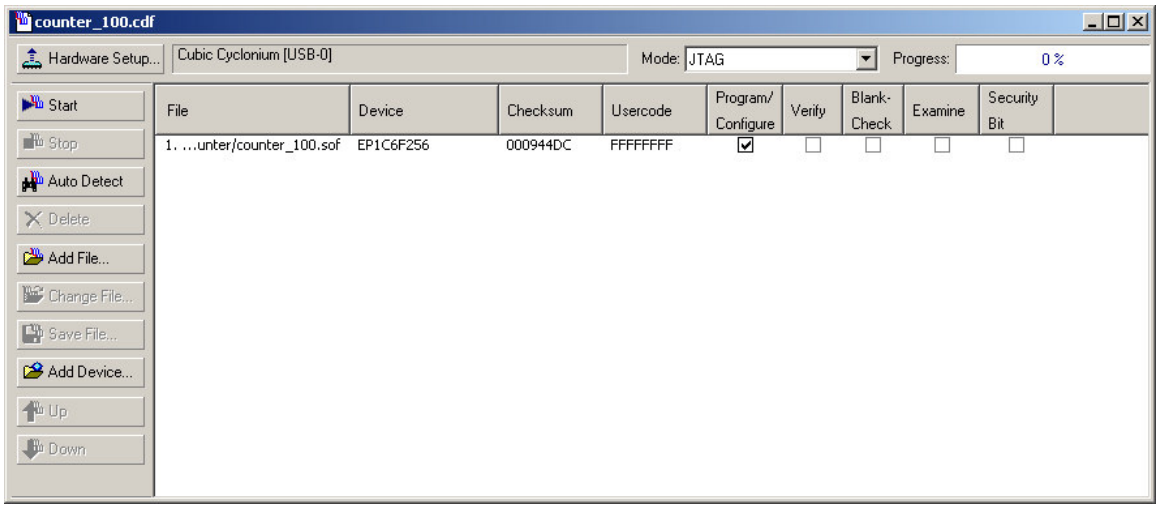
5. In the main programming window, you should see the file **counter_100.sof** listed along with its target device, the **EP1C6F256**.

*If you do not see the **counter_100.sof** configuration file listed in the programmer window, click on the **Add File** button and select it.*

6. Enable the **Program/Configure** option for the **counter_100.sof** file.

   Remember if you want to bypass any devices in your JTAG chain, you can simply leave this option unchecked for those devices.

7. Your **counter_100.cdf** file should look as below. If so, click on the **Start** button to begin configuration.



The LED Matrix should begin counting from 0 to 99.

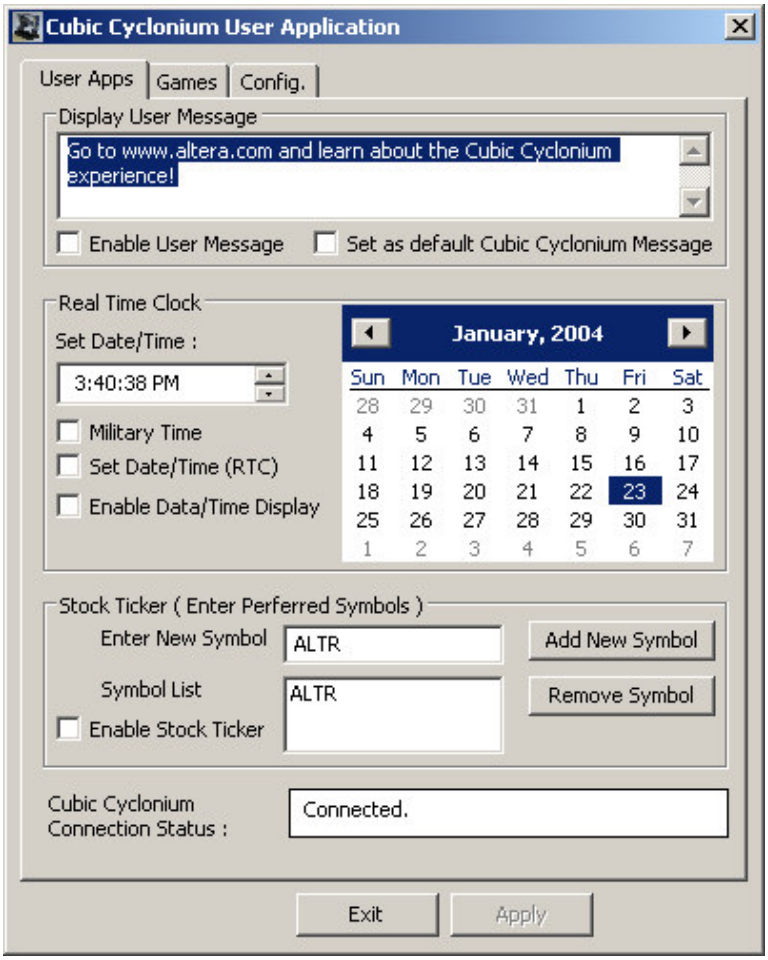
Step 3 (Optional : Program the Cubic Cyclonium Demo)

*You have officially completed the final exercise. If you have some extra time and want to see an example of a **Nios** embedded processor running on a **Cyclone** device, follow the steps below. You will be loading the original design back into the device.*

1. In your **CDF** file, highlight the **counter_100.sof** file.
2. Click on the **Change File** button. Browse to the ***<Quartus II lab install directory>*\QII4_0\Lab6\cubic cyclonium\cc_quartus_project** directory and select the **thrifty_32.sof** file.
3. Click on the **Start** button to program the **Cyclone** device.

Step 3 (Optional : Run the Cubic Cyclonium User Application)

1. Using Windows Explorer, locate the **Cubic Cyclonium.exe** file. It should be located in the ***<Quartus II lab install directory>*\QII4_0\Lab6\cubic cyclonium** directory. After a splash screen, the window below should be available.



2. Once the application has begun, feel free to do any of the following. Click **Apply** to have your information transmitted to the **Nios** processor.

   Enable User Messages (User Apps tab)

   Set Date/Time Display (User Apps tab)

   Tetris (Games tab)

   LED Savers (Games tab)

Exercise Summary

- Created a Chain Description File (CDF)
- Programmed a Cyclone device

END OF EXERCISE 6

Exercise 7

(Optional Nios Development Board Programming Lab)

Exercise 7

Objectives:

- Create a chain description file (CDF) to use in programming a JTAG chain
- Use Programmer tool to configure a device

Exercise 7

This exercise is a quick demonstration of how to set up the programmer and program a device.

Step 1 (Connect Cubic Cyclonium Board & Open Quartus II Project)

1. Take out your **Nios Stratix** or **Cyclone** development board. Power the board and connect the **ByteBlasterII** download cable to the parallel port of your PC or laptop. Connect the 10-pin female connector of the **ByteBlasterII** download cable to the development board. You will find the 10-pin header near the top left corner of the board.
2. **Open** the project **counter.qpf** located in the ***<Quartus II lab install directory>*\QII4_0\Lab7\counter_nios** directory.

   This project contains a simple 2-digit decimal counter.
3. From the **Project** menu, select **Revisions**. Choose either **counter_stratix** or **counter_cyclone** depending on your Nios board. If not already checked, click on the **Set Current** button to place a check next to the correct revision. Click on **Close** when finished.

Step 2 (Open and Set Up Programmer to Configure Device)

1. From the **Tools** menu or toolbar  , select **Programmer**.

   *This will open a **CDF** as shown below. The **CDF** file lists all of the devices in your configuration or JTAG chain along with their associated programming or configuration files.*
2. If not already named, save the CDF as **counter_nios.cdf**.
3. Locate the **Hardware Setup** button at the top of the **CDF** file. The field to the right of this button should read **ByteBlasterII**.

   *If this window reads anything else, click on the **Hardware Setup** button and select **ByteBlasterII** from the list of **Available Hardware**. Click on the **Select Hardware** button and then **Close**.*

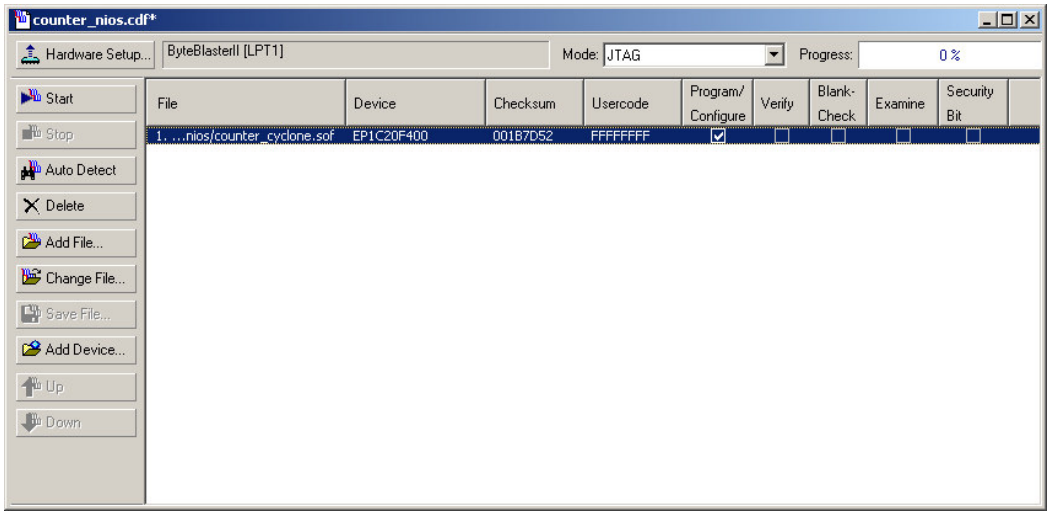
   *If **ByteBlasterII** cable is not listed in the **Available Hardware** window, please let the instructor know as the drivers may not have been loaded correctly onto your PC.*
4. Back in the **CDF** file, use the drop-down **Mode** field to choose **JTAG**.
5. In the main programming window, you should see the file **counter_stratix.sof** or **stratix_cyclone.sof** listed along with its target device.

   *If you do not see the **counter_<device>.sof** configuration file listed in the programmer window, click on the **Add File** button and select it.*

6. Enable the **Program/Configure** option for the configuration file and target device.

   Remember if you want to bypass any devices in your JTAG chain, you can simply leave this option unchecked for those devices.

7. Your **CDF** file should look similar to the figure below. If so, click on the **Start** button to begin configuration.



The LED display should begin counting from 0 to 99 when configuration is complete.

Exercise Summary

- Created a Chain Description File (CDF)
- Programmed an FPGA device

END OF EXERCISE 7