

PIPELINED DATAPATHS

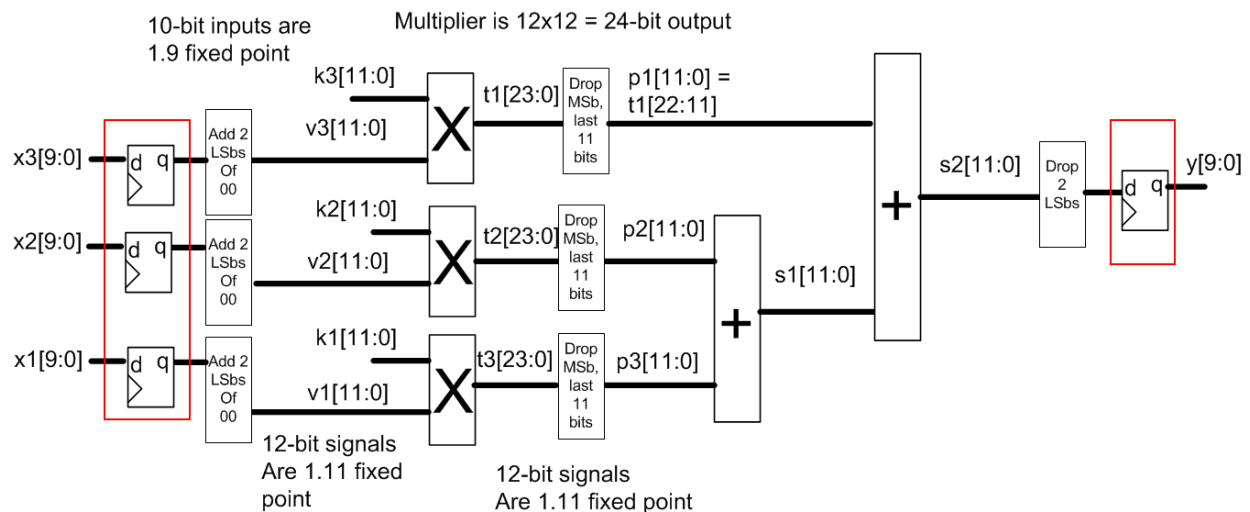
This lab has you pipeline the multiplier/adder datapath that you create for the previous lab. This does not have to be downloaded into the Basys board.

BACKGROUND

Review the notes on pipelined datapaths and implementing DFFs in Verilog.

TASK DESCRIPTION

Modify the Verilog module that you created for the previous lab by adding pipeline registers to the inputs and outputs and shown below:



After this is working, you will add additional pipeline registers to improve performance.

Procedure

1. Download the zip archive associated with the lab.
2. This contains the following files:
 - *lab5dpath.v* -- complete this module
 - *tb_lab5dpath.v* -- test bench
 - *multadd_vectors.txt* -- input vector file for testbench.
 - *lab5_computations.xls* -- spreadsheet file that contains the test vectors in decimal form, shows the intermediate calculations in hex and decimal. You can use this to help debug your design.

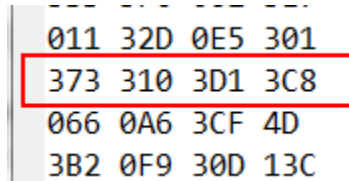
- *report.doc* – report file that needs to be filled out. **You should open this file, and fill out the requested information as you perform the remaining steps.**
3. Create a project named *lab5* and add the *lab5dpath.v* file to it as the top module. You can copy the contents of your previous lab solution to this file as you will be modifying this design. Then, add the testbench file *tb_lab5dpath.v* as the simulation file.
 4. Copy the *multadd_vectors.txt* file into the project directory. This is needed before you simulate your design.
 5. Copy the *Basys3_Master.xdc* file from the first lab to your project directory, and then add it to your project. Comment out all of the lines that have the 'set_property' command for pins. This is easy to do by opening the file in Vivado, selecting a group of lines, and then use the "Toggle Line Comment" from the right-click menu. Then add the following line to the file (does not matter where):

```
create_clock -period 20.000 -name sys_clk_pin -waveform {0.000 10.000}
-add [get_ports clk]
```

This creates a clock with a 20 nanosecond period (50 MHz). This creates a clock constraint for the synthesis tool; it will try to synthesize the logic such that it works with a maximum clock frequency of 50 MHz.

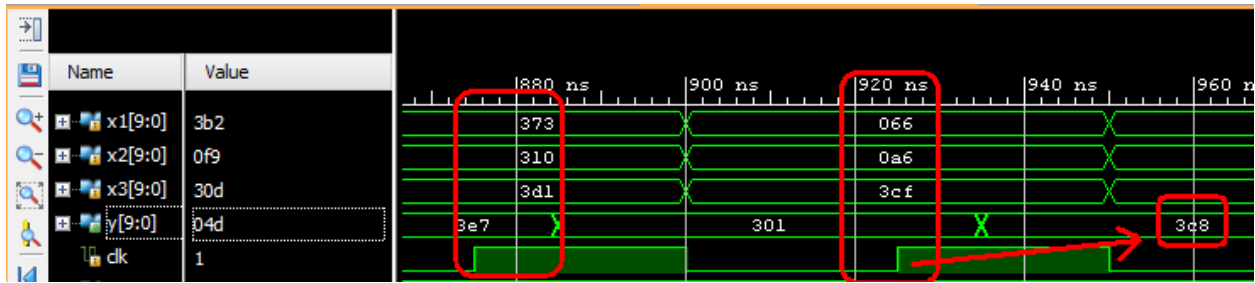
6. Modify your design from the previous lab to add the input/output pipeline registers as shown in the figure. You must use a hardmacro multiplier implementation. Using a behavioral simulation, verify that your modified design passes the vectors in the testbench. This design has a latency of 2 since there are pipeline registers on the inputs/outputs and no other pipeline registers in the design. Once the simulation is working, pick any vector and capture simulation screenshots showing that the latency is 2 (see the following screenshots, you must pick a different test vector!!):

From the input vector file (the first three values are x1, x2, x3; the last value is the expected result):



```
011 32D 0E5 301
373 310 3D1 3C8
066 0A6 3CF 4D
3B2 0F9 30D 13C
```

From the simulation, observe that two rising clock edges is necessary to get the output of 0x3C8 (the design has a latency of 2). The first rising edge clock edge clocks the input vectors into the first set of DFFs. During this clock period, the values propagate through the datapath and setup on the output DFFs. The second rising clock edge clocks the result out to the Y bus. This screenshot is from a post implementation simulation, so there is delay from the second rising clock edge until the Y bus actually settles to the correct value of 0x3C8.



7. Once the behavioral simulation is working, use the 'Run Implementation' command to map this design to the FPGA. Verify that the Post-Implementation Timing summary passes as well. You do not have to download it into your board.
8. Capture a simulation screenshot like above from the Timing simulation and annotate it to show the two clock-cycle latency. Pick a different test vector than what is shown above. Include the screenshot from the vector file with the vector line circled as shown above.
9. Copy the Utilization tables that we have used in the past to your report (Slice Logic, Summary of Registers by Type, DSP). This design has DFFs in it, so the number of registers will not be zero as was in previous labs.
10. Run the 'Report Timing Summary' tool as you did in the previous lab. In the 'Timing' Table, open the 'Intra-Clock paths for the sys_clk_pin/setup'. This will give you the longest register to register delay in your design. The 'slack' is the extra time in our clock constraint period that is not needed. In this design, we have a lot of slack and could have specified a higher clock frequency for our constraint.

Timing - Timing Summary - timing_1

Intra-Clock Paths - sys_clk_pin - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Source Clock	Destination Clock
Path 1	11.494	5		7 x2q_reg[9]/C	y_reg[7]/D	8.546	5.508	3.038	sys_clk_pin	sys_clk_pin
Path 2	11.502	5		7 x2q_reg[9]/C	y_reg[9]/D	8.538	5.500	3.038	sys_clk_pin	sys_clk_pin
Path 3	11.578	5		7 x2q_reg[9]/C	y_reg[8]/D	8.462	5.424	3.038	sys_clk_pin	sys_clk_pin
Path 4	11.598	5		7 x2q_reg[9]/C	y_reg[6]/D	8.442	5.404	3.038	sys_clk_pin	sys_clk_pin
Path 5	11.707	4		7 x2q_reg[9]/C	y_reg[5]/D	8.333	5.295	3.038	sys_clk_pin	sys_clk_pin
Path 6	11.772	4		7 x2q_reg[9]/C	y_reg[4]/D	8.268	5.230	3.038	sys_clk_pin	sys_clk_pin
Path 7	11.861	5		7 x2q_reg[9]/C	y_reg[3]/D	8.179	5.508	2.671	sys_clk_pin	sys_clk_pin

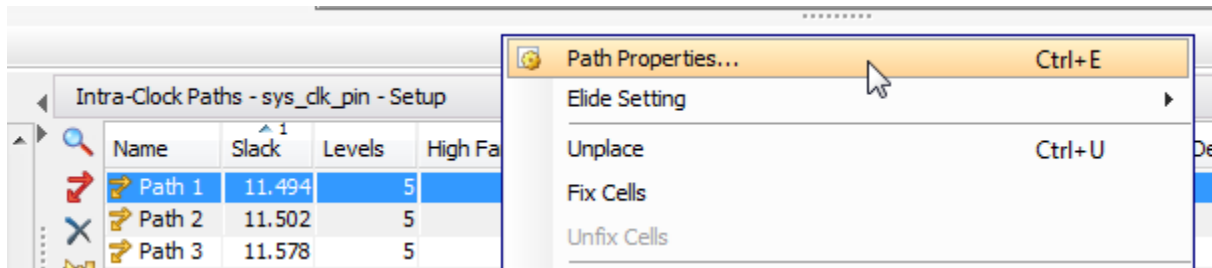
The 'sys_clk_pin/hold' paths report the shortest register-to-register path delays. The 'slack' in this case refers to the amount of extra time available to avoid a hold time violation on the destination register.

Timing - Timing Summary - timing_1

Intra-Clock Paths - sys_clk_pin - Hold

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Source Clock	Destination Clock
Path 11	1.134	3		2 x3q_reg[3]/C	y_reg[5]/D	1.305	0.844	0.461	sys_clk_pin	sys_clk_pin
Path 12	1.135	3		2 x3q_reg[3]/C	y_reg[1]/D	1.305	0.844	0.461	sys_clk_pin	sys_clk_pin
Path 13	1.136	3		2 x3q_reg[3]/C	y_reg[3]/D	1.307	0.846	0.461	sys_clk_pin	sys_clk_pin
Path 14	1.161	3		2 x1q_reg[4]/C	y_reg[4]/D	1.332	0.845	0.487	sys_clk_pin	sys_clk_pin
Path 15	1.162	3		2 x3q_reg[3]/C	y_reg[6]/D	1.334	0.850	0.484	sys_clk_pin	sys_clk_pin

11. Select the longest setup path, and use the right-click menu 'Path Properties'.



12. The Summary pane for the path has information about the starting, ending pins of this path, the number of logic levels, and the detailed paths for both the clock (Source Clock Path) and the data (Data Path). Since we only have input and output registers in this design, the path will have to run the output of an input register to the input of some output register.

Summary

Name

Path 1

Slack

11.494ns

Source

x2q_reg[9]/C (rising edge-triggered cell FDRE clocked by sys_clk_pin {rise@0.000ns fall@10.000ns period=20.000ns})

Destination

y_reg[7]/D (rising edge-triggered cell FDRE clocked by sys_clk_pin {rise@0.000ns fall@10.000ns period=20.000ns})

Path Group

sys_clk_pin

Path Type

Setup (Max at Slow Process Corner)

Requirement

20.000ns (sys_clk_pin rise@20.000ns - sys_clk_pin rise@0.000ns)

Data Path Delay

8.546ns (logic 5.508ns (64.448%) route 3.038ns (35.552%))

Logic Levels

5 (CARRY4=2 DSP48E1=1 LUT3=1 LUT4=1)

Clock Path Skew

-0.033ns

Clock Uncertainty

0.035ns

Source Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock sys_clk_pin rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: M18	clk
net (fo=0)	0.000	0.000		clk
			Site: M18	clk_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 0.938	0.938	Site: M18	clk_IBUF_inst/O
net (fo=1, routed)	1.972	2.910		clk_IBUF
			Site: BUFGCTRL_X0Y0	clk_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.096	3.006	Site: BUFGCTRL_X0Y0	clk_IBUF_BUFG_inst/O
net (fo=40, routed)	1.552	4.558		clk_IBUF_BUFG
FDRE			Site: SLICE_X12Y22	x2q_reg[9]/C

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_O)	(r) 0.518	5.076	Site: SLICE_X12Y22	x2q_reg[9]/Q
net (fo=7, routed)	0.787	5.863		u2/U0/i_mult/gDSP.gDSP_only.iDSP/B[11]
			Site: DSP48_X0Y8	u2/U0/i_mult/gDSP.gDSP_only.iDSP/multOp/B[15]
DSP48E1 (Prop_dsp48e1_B[15]_P[15])	(r) 3.656	9.519	Site: DSP48_X0Y8	u2/U0/i_mult/gDSP.gDSP_only.iDSP/multOp/P[15]
net (fo=2, routed)	1.395	10.914		p2[4]

Capture a screenshot like above for your design that gives the details for the longest register-to-register path in your design.

13. Back in the timing summary, open the 'Unconstrained Paths/sys_clk_pin to NONE' – The 'setup' times list longest clock to output times, while the 'hold' times list the shortest clock to output times. The slack is 'infinity' as we do have not put any constraints on these times.

Timing - Timing Summary - timing_1

Unconstrained Paths - sys_clk_pin - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Source Clock
Path 21	∞	1		1 y_reg[5]/C	y[5]	5.706	3.136	2.570	sys_clk_pin
Path 22	∞	1		1 y_reg[9]/C	y[9]	5.662	3.140	2.522	sys_clk_pin
Path 23	∞	1		1 y_reg[4]/C	y[4]	5.537	3.114	2.422	sys_clk_pin
Path 24	∞	1		1 y_reg[1]/C	y[1]	5.528	3.113	2.415	sys_clk_pin
Path 25	∞	1		1 y_reg[2]/C	y[2]	5.516	3.106	2.410	sys_clk_pin
Path 26	∞	1		1 y_reg[6]/C	y[6]	5.514	3.132	2.382	sys_clk_pin

Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths
sys_clk_pin to NONE
Setup (10)
Hold (10)

The 'NONE to sys_clk_pin' setup times give the longest input pin to Input register delay, while the hold times give the shortest delay times from input pin to input register.

Timing - Timing Summary - timing_1

Unconstrained Paths - NONE - sys_clk_pin - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Source Clock	Destination Clock
Path 41	∞	1		1 x1[1]	x1q_reg[1]/D	2.874	0.931	1.943	input port dock	sys_clk_pin
Path 42	∞	1		1 x1[5]	x1q_reg[5]/D	2.842	0.935	1.907	input port dock	sys_clk_pin
Path 43	∞	1		1 x1[7]	x1q_reg[7]/D	2.821	0.936	1.886	input port dock	sys_clk_pin
Path 44	∞	1		1 x1[0]	x1q_reg[0]/D	2.749	0.936	1.813	input port dock	sys_clk_pin
Path 45	∞	1		1 x1[8]	x1q_reg[8]/D	2.737	0.951	1.786	input port dock	sys_clk_pin
Path 46	∞	1		1 x1[4]	x1q_reg[4]/D	2.730	0.945	1.785	input port dock	sys_clk_pin
Path 47	∞	1		1 x1[3]	x1q_reg[3]/D	2.726	0.935	1.791	input port dock	sys_clk_pin
Path 48	∞	1		1 x1[2]	x1q_reg[2]/D	2.702	0.934	1.768	input port dock	sys_clk_pin
Path 49	m	1		1 x1[9]	x1q_reg[9]/D	2.681	0.933	1.747	input port dock	sys_clk_pin

Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths
sys_clk_pin to NONE
Setup (10)
Hold (10)
NONE to sys_clk_pin
Setup (10)
Hold (10)

Capture screenshots like the above two screenshots and include them in your report.

14. Save your Verilog file as *lab5dpath_part1.v* in your project directory.
15. In the same project, create another DSP multiplier implementation (use a different name!). This will be different from the original implementation you originally created in that it will have a latency value of 1. This means that it will have a pipeline register on the output of the multiplier. This requires you to select '1' for pipeline stages in the third screen.

Basic **Output and Control**

Output Product Range

☐ Use Custom Output Width

Output MSB: 23 [0 - 127]
Output LSB: 0 [0 - 23]

Output product width (max, min) = (23,0)

☐ Use Symmetric Rounding

Pipelining and Control Signals

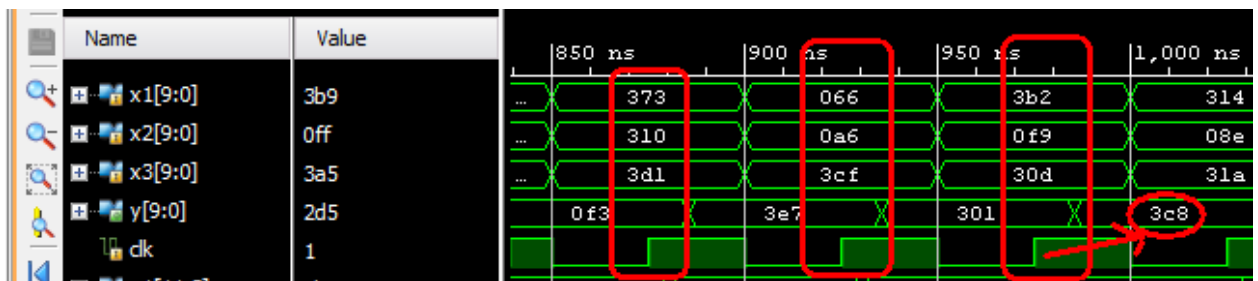
Pipeline Stages: 1 Optimum pipeline stages: 3

☐ Clock Enable ☐ Synchronous Clear

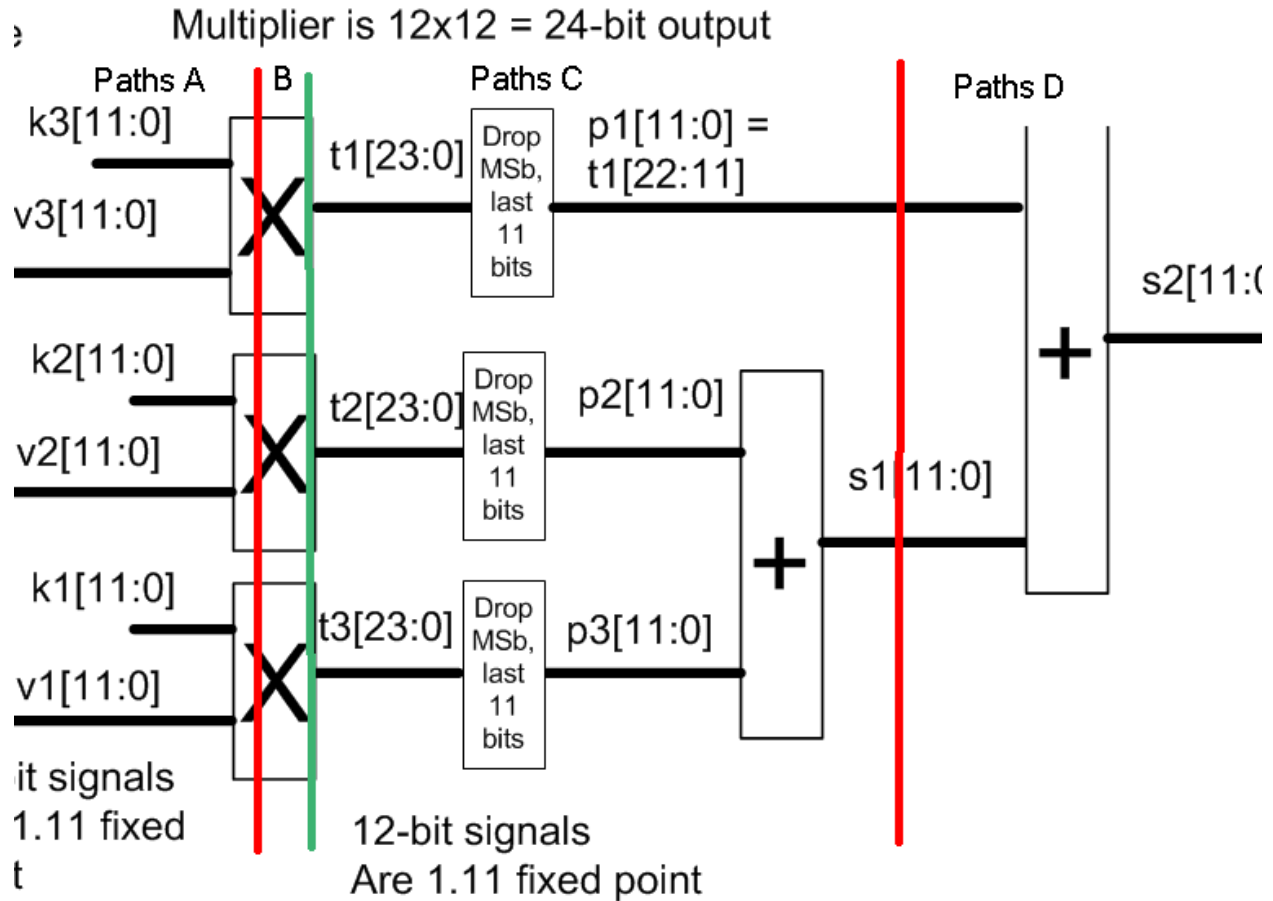
Synchronous Controls and Clock Enable(CE) Priority: SCLR Overrides CE

The effect on the overall design of putting a pipeline register on the multiplier output is that this breaks the longest path from input DFFs to output DFFs into two sections (paths from the input DFFs to the DFFs on the multiplier output, and paths from the multiplier output DFFs through the adder logic to the output DFFs).

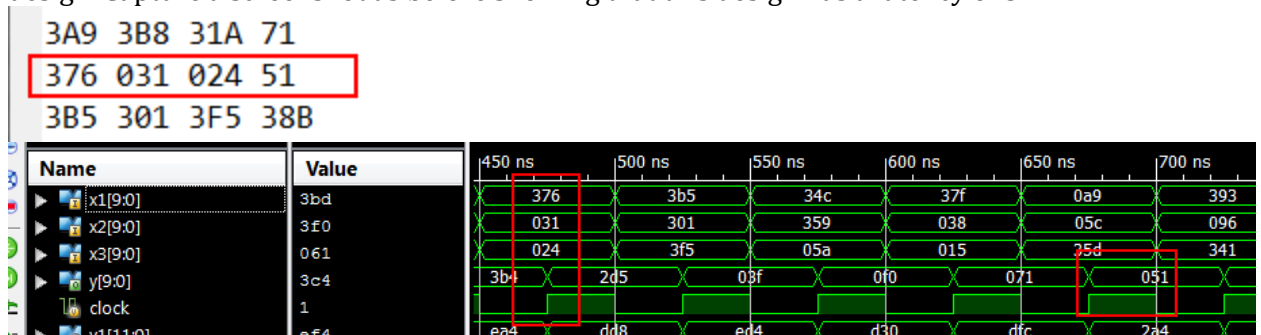
16. Modify your *lab5dpath.v* file to use this new multiplier. NOTE: The new multiplier will now have a ".CLK" input so you will have to modify your multiplier instantiations to add this new pin as well as change the module name.
17. Your new design will now have latency of 3. Edit the *tb_lab5dpath.v* testbench file and locate the LATENCY define statement. It should have a value of 2, change this to 3. This value must match the latency of the datapath for the testbench to function correctly.
18. Verify the behavioral and post implementation timing simulation of your new design. Capture a screenshot as before showing that this design has a latency of 3.



19. Copy the resource usage of your design as before. Perform the same timing analysis as before and capture a screenshot of the longest Intra-clock Path/sys_clk_pin/setup details. You should find that the total delay has decreased (slack increased) as the previously longest path is now broken by a register on the output of the multiplier. In the report, answer the question that asks you to identify whether the longest path identified in the detailed timing report is in the multiplier logic or adder logic; include information from the detailed timing report to support your answer.
20. Save your Verilog module file as *lab5dpath_part2.v*.
21. Edit your design so that two more pipeline stages are added as shown by the RED lines in the diagram below. The green line is the original pipeline stage in the multiplier. Generate a new hardmacro multiplier that has two pipeline stages (use a different name!!), the red line in the multiplier is the second pipeline stage). Also, manually add a pipeline stage in your Verilog module that breaks up the adder combinational path as shown. This means that your datapath will now have a latency of 5, do not forget to edit the testbench file before simulation. These pipeline registers break the paths in the design from input DFFs to output DFFs into four sections:
 - Paths A – from input DFFs to the first multiplier pipeline stage
 - Paths B – from the first multiplier pipeline stage to the second multiplier pipeline stage
 - Paths C – from the second multiplier pipeline stage to the adder logic pipeline stage
 - Paths D – from the adder logic pipeline stage to the output DFFs



22. Verify the behavioral simulation and post implementation timing simulations of your new design. Capture a screenshot as before showing that this design has a latency of 5.



23. Copy the resource usage of your design as before. Perform the same timing analysis as before and capture a screenshot of the longest Intra-clock Path/sys_clk_pin/setup details.. You should find that this design has a smaller delay than the previous design because the longest paths have been broken by additional pipeline registers. Look at the detailed timing report, and determine what logic section (A, B, C, or D) the longest path is in – answer the question in the report concerning this and include any information from the timing report that justifies your answer.

24. Save your final Verilog file as *lab5dpath_part3.v* and complete the rest of the report.

GRADING

The grading point distribution is specified in the report document.

SUBMISSION INSTRUCTIONS

Create a directory named 'lab5_*netid*', i.e., (lab5_rbr5).

Copy your lab5 directories to this directory.

Copy your completed report.doc to this directory.

From Windows, select the 'lab5_*netid*' folder, and from the right-click menu, use 'Send To Compressed (zipped) Folder' command to produce a ZIP archive named 'lab5_*netid*.zip'.

Upload your 'lab5_*netid*.zip' file to Blackboard using the Lab5 assignment link.