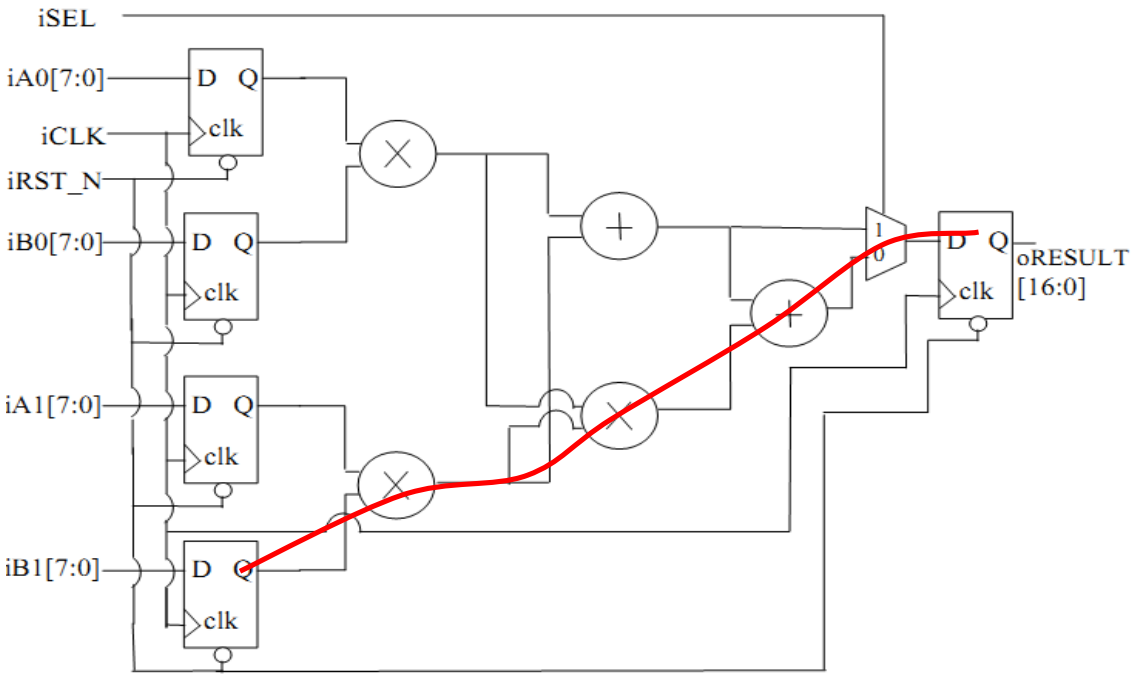
EE/CprE 465 -- Fall 2015

Lab 7: System Timing Optimization

In previous labs, you have already learned basic digital circuit design skills. In this lab, you will learn a fundamental skill to optimize the system timing. Please recall the design in Lab 4. It was the circuit below:

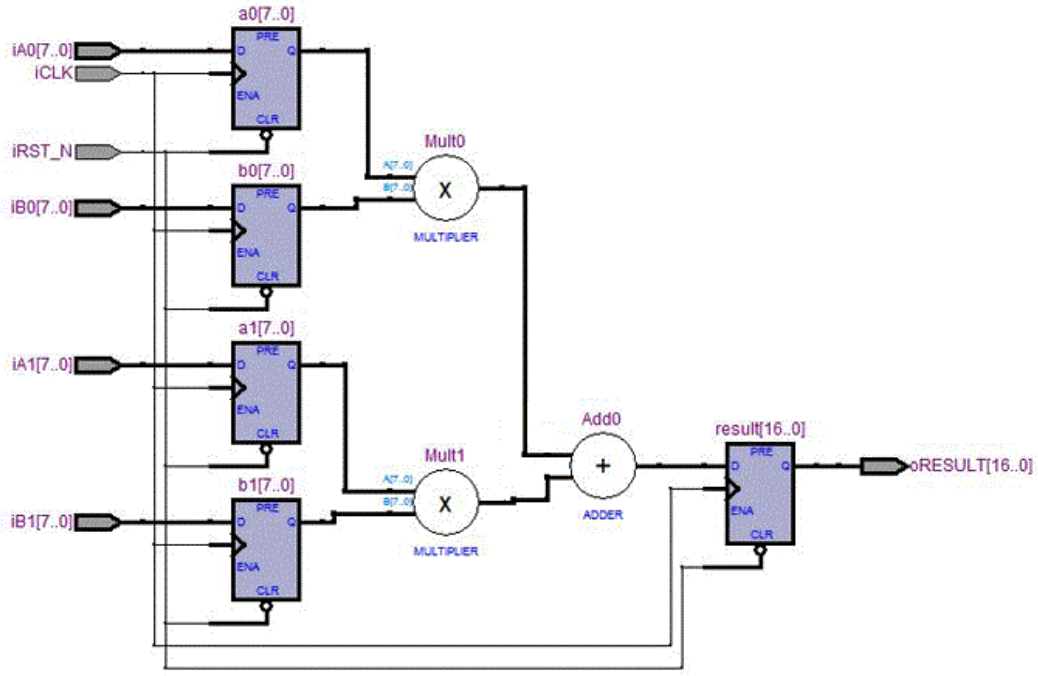


Do you remember the worst timing path that you reported in Lab 4? It should be the one marked in red in the diagram above. Now what is the highest clock frequency your circuit can run with based on timing reports after placement and routing? It determines the throughput of your circuit. For example, if the minimum clock period (which equals 1/ highest clock frequency) of your circuit is 5ns, it means for every 5ns, your circuit can take in a new set of inputs and output the result of last inputs. Now if your circuit could take in a new set of inputs and output the result of last inputs for every 4 or even 3ns, the throughput would be increased. But given that the circuit has already been adequately optimized by synthesis and layout tools, how can it be achieved? We will learn a very commonly used technique called pipelining in this lab.

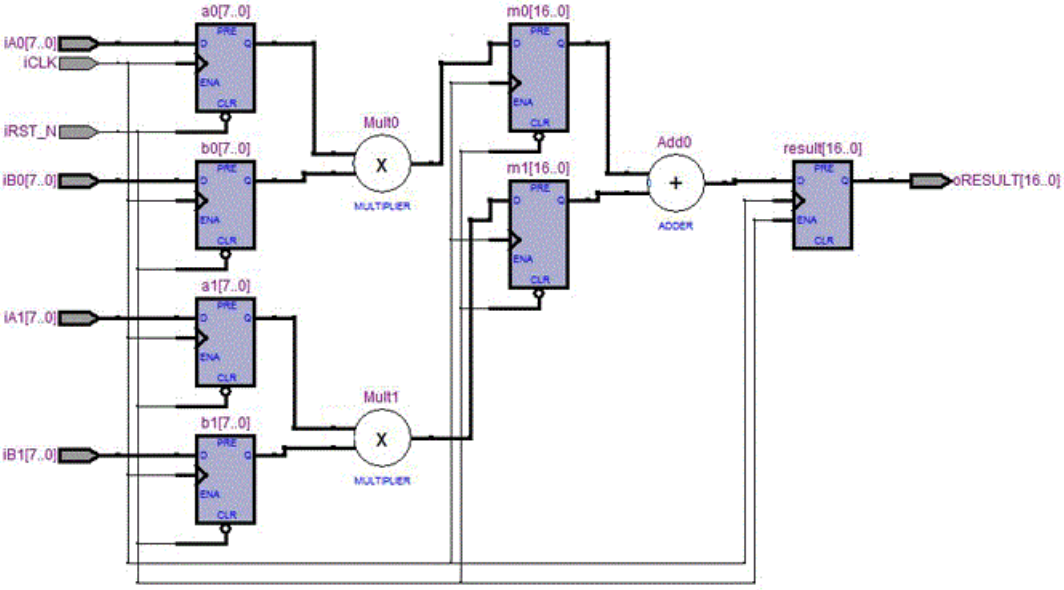
Pipelining is a technique to partition the worst timing path of a circuit into several shorter paths by inserting extra DFFs. In other words, the computation is broken down into several clock cycles (or stages). If DFFs are inserted in appropriate locations, the delay of the worst timing path can be significantly reduced. Hence the clock frequency and the throughput can be significantly increased. But there is a caveat: To ensure the circuit will function correctly after pipelining, the number of DFFs inserted along any path from an input DFF to an output DFF in the original circuit should be the same. Otherwise, the partial results corresponding to inputs at different clock cycles would be mixed together.

The cost of pipelining is the increase in area, power and latency (the time it takes for a data to travel from input to output) due to the extra DFFs. The benefit and the cost of pipelining are determined by how many stages are used and where the DFFs are inserted. Thus pipelining should be applied carefully.

Here is an example which is very similar to our design:



After applying pipelining, the circuit is changed to:



Please find the Verilog files posted for this example, and run simulation by ModelSim and synthesis by RTL Compiler to learn about how it works. Note that the input ports and output ports of this example are similar but not the same as our design in Lab 4. So you can reuse your synthesis scripts by making minor modifications to them. Also make sure you change the Verilog file name in the scripts to read in the right file.

You should find out that, the design with pipelining can run with higher clock frequency and therefore has higher throughput. After you learned the pipeline technique, it is time for you to try it yourself on the design in Lab 4.

Lab Tasks:

1. Before you can change the design in Lab 4 into a pipelined one, we want to point out a problem with the design. First, please make sure the number and locations of DFFs in your design are EXACTLY the same as the diagram at the beginning of this document. Do you realize that iSEL is not controlled by iCLK but the input data iA0, iA1, iB0 and iB1 are controlled by iCLK? When iSEL is set up in a certain clock cycle, does it affect the computation of the input data of the same clock cycle?

   Please observe this problem by performing the following experiment.

   Using your original Verilog codes (not the output result of synthesis) in Lab 4 and make sure your circuit has all DFFs at input ports iA0, iA1, iB0, and iB1, but not at iSEL. Modify your test bench to use the input values below:

   - Reset your circuit by iRST_N
   - At rising clock edge #1, iA0, iA1, iB0, and iB1 are all set to binary '00000011', and iSEL ='0'
   - At rising clock edge #2, iA0, iA1, iB0, and iB1 are all set to binary '00000001', and iSEL ='1'

   Run simulation until rising clock edge #3. What are the values of oRESULT after each clock edge? Are the output values what you expect? Please explain any difference.

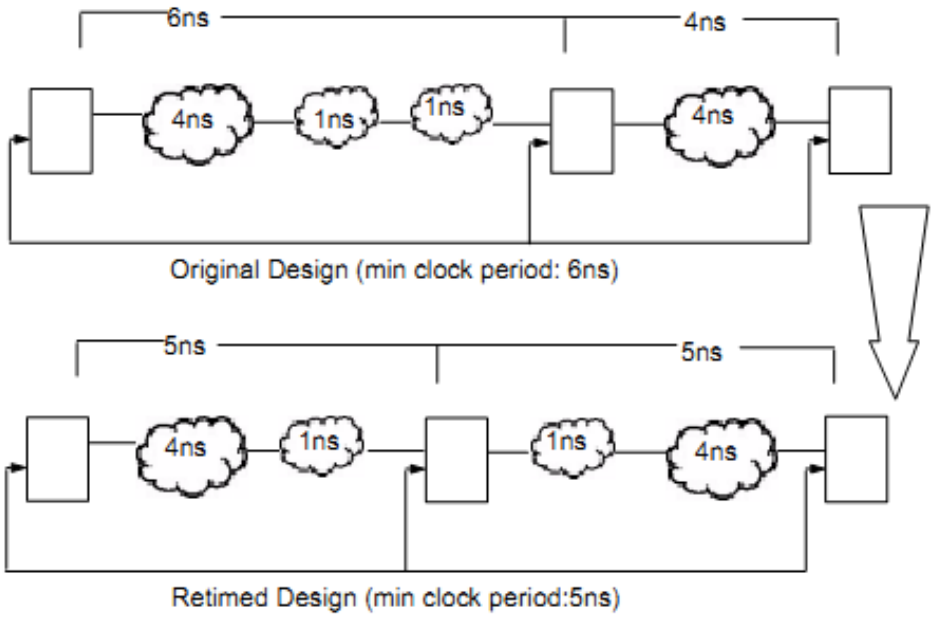
   Please fix this problem by modifying the iSEL signal in your Verilog code.

2. Apply pipelining to the design after the fix in Task 1 by modifying your Verilog code. (Notice that you may see a similar problem to the one in Task 1 above if you do not insert the same number of DFFs to all input-output paths.)

3. Go through the flows of functional simulation as we did in Lab 1 and synthesis as we did in Lab 4. Try your best to achieve the fastest timing.

4. Compare the timing, area and power reports of the design which has pipelining with the design which does not have. And most importantly, compare the throughputs of these two designs.

5. Retiming: Now think about if you can improve the throughput any more. Did you insert the DFFs in the right places of the timing paths? If you did not insert the DFFs in the best places, your design still has potential for improvement. For the example below, just move DFF a little bit to the left in the timing path, you can easily decrease the clock period by 1ns, which is 17%! We call this technique “retiming”.



Retiming is not easy for us to do manually because we need to know the timing information for every path. Fortunately, the synthesis software can do it. Just insert the following command after "elaborate" in your RTL-Compiler script named run_synth.tcl located in /rc/syn/scripts directory.

```
set_attribute   retime    true    /ALT_MULTADD_pipe
```

Please note that `ALT_MULTADD_pipe` is the design module name. You need to change it to yours. Then run synthesis again to see if you get improvement on timing. And also report area and power. Compare them with the results when you are not doing retiming. Also check the generated schematic in RTL-Compiler to see what has been done to your circuit by the software. Does it really move your inserted DFFs? Does it change the number of DFFs that you use?