EE/CprE 465 -- Fall 2015

Lab 8: System Area Optimization

In the last lab, you learned how pipelining can improve your design's throughput. In this lab, you will optimize the area of your design. Before that, let us first study an example. For the same example you have gotten in Lab 7:

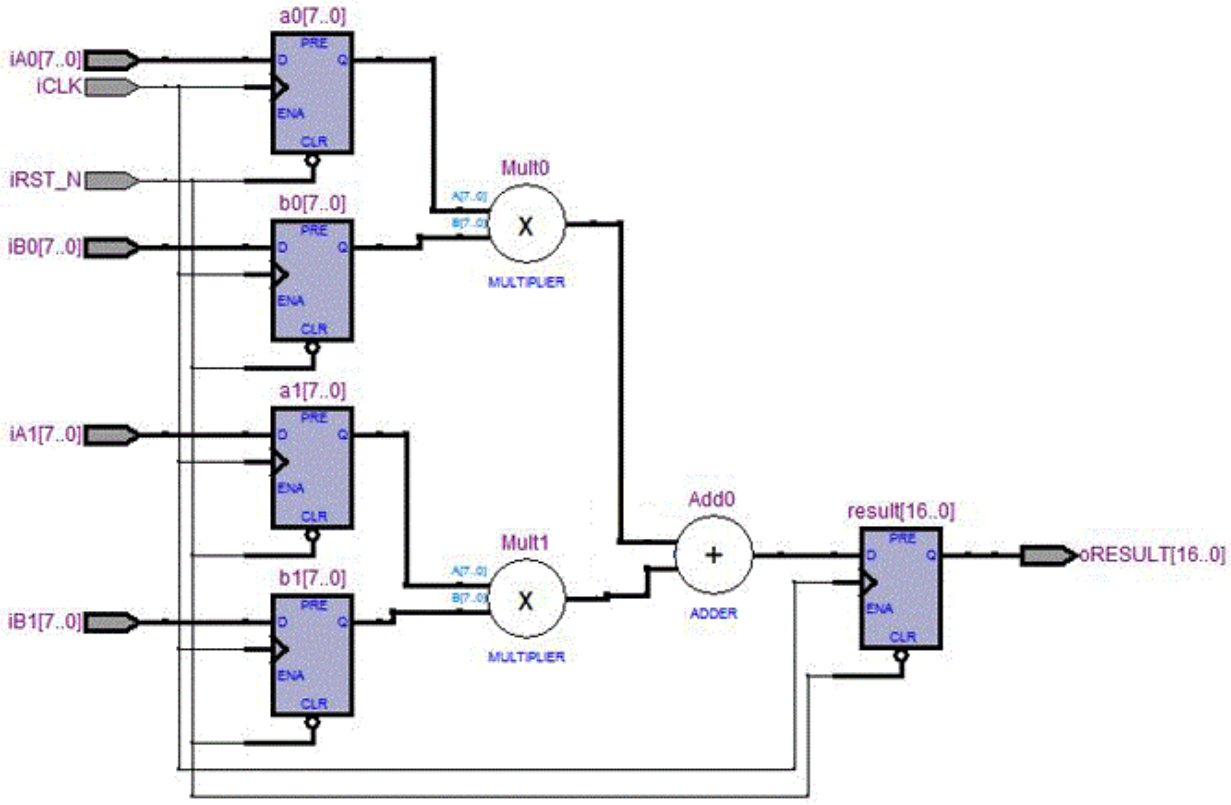

Fig. 1: An example design.

How can we save chip area for this circuit? Please note that we have 2 multipliers in the circuit and they occupy a large area. If we can save one of them by reusing the other one, we will save part of the area of the whole circuit. In order to use a single multiplier, the two multiplications should be performed in two different clock cycles. That means we need to temporarily store the operands / result of one multiplication when the multiplier is handling the other multiplication. In Task 1 of the last lab, we learned that the value of a signal can be stored and delayed by one clock cycle by inserting a register. We can make use of registers in a similar manner to store and delay the signals when eliminating one multiplier:

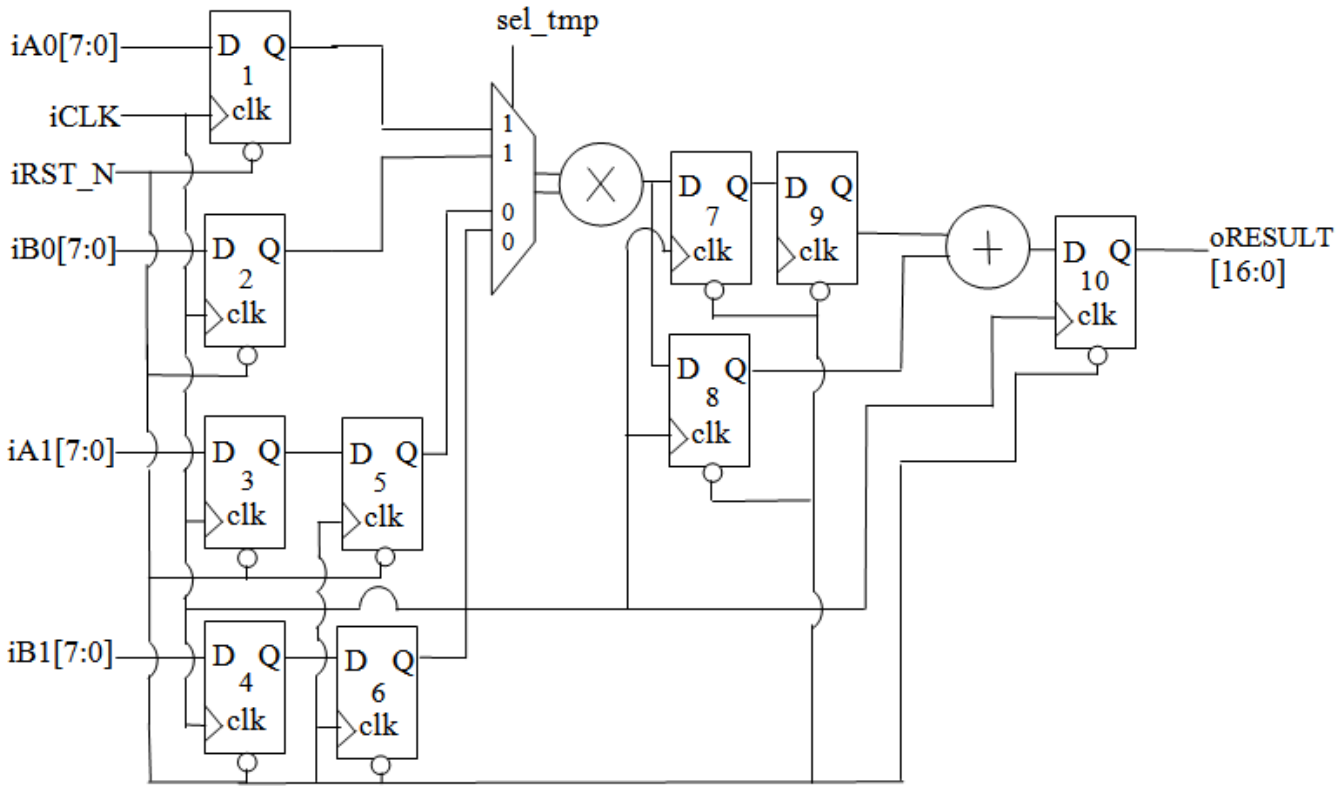

Fig. 2: Area optimized circuit for the example design.

By inserting two more registers after the input ports of iA1 and iB1, and inserting a 16-bit 2-to-1 MUX, we can delay the iA1 and iB1 signals for one clock cycle. Then we can reuse the upper multiplier. Note that we also need to add registers at the output of the multiplier to temporarily store the results. The details are explained below:

- At the 1st rising edge of clock, sel_tmp should be set to 1. The inputs of the multiplier are iA0 and iB0.
- At the 2nd rising edge of clock, the output of the multiplier is captured into register #7 and register #8 at its output. At the same time, sel_tmp should be set to 0. The inputs of the multiplier are then iA1 and iB1.
- At the 3rd rising edge of clock, the current output of the multiplier is captured into register #7 and register #8. The previous output of the multiplier is moved to register #9. Hence, the inputs of the adder now are the previous output and the current output of the multiplier.
- At the 4th rising edge of clock, the output of adder is captured into register #10 at its output.

In this way, we saved the lower multiplier by reusing the upper one. An important point to note is that this circuit would have been pipelined if we were not reusing the multiplier. However, after multiplying iA0 and iB0 in one clock cycle, the multiplier needs to multiply iA1 and iB1 and hence is not available to multiple iA0 and iB0 of another input in the next cycle. Therefore, pipelining does not work as before. The circuit cannot take in a new set of inputs in every clock cycle.

Now it is time to try the idea on our design. Please recall the design we did in Lab 4. It is the circuit below:

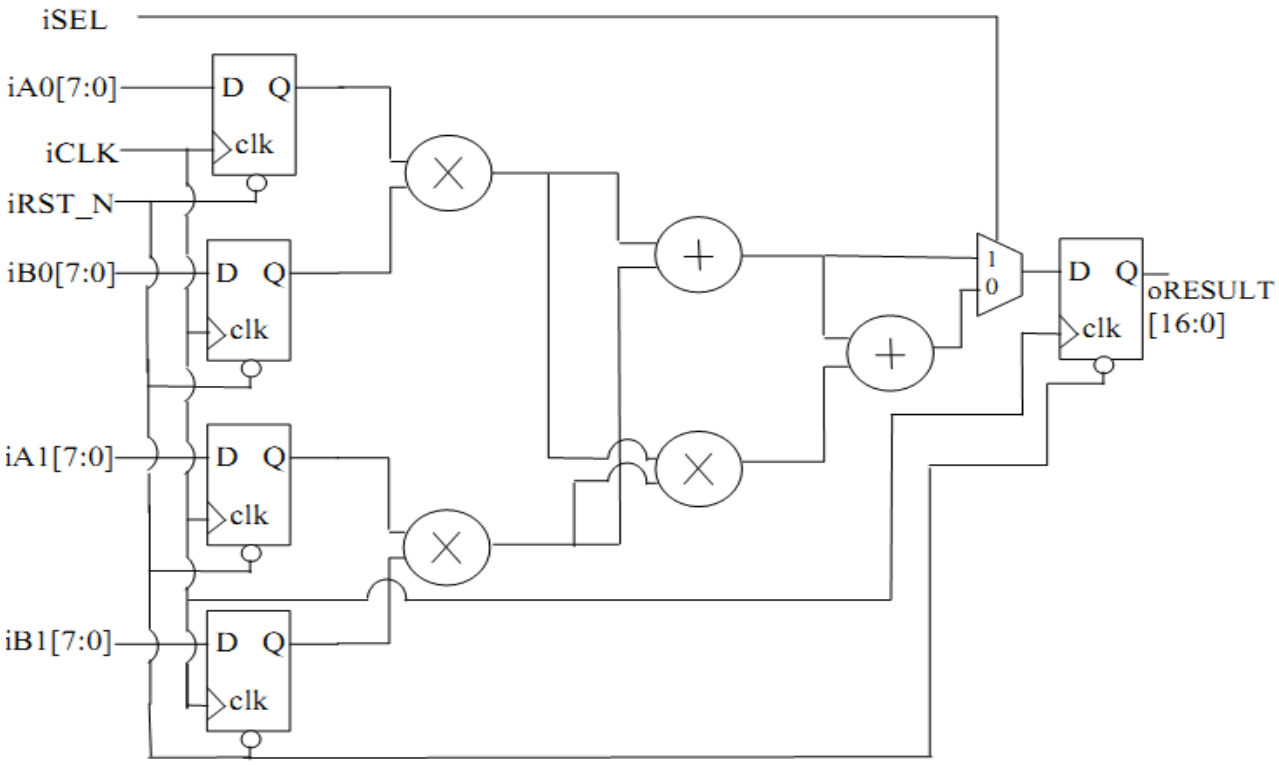

Fig. 3: Our original design from Lab 4.

Lab Task:

1. In Fig. 2, one of the registers can actually be eliminated. Please figure out which register is not necessary and modify the circuit diagram to eliminate it. You may just draw the modified circuit by hand but draw it clearly.
2. Please draw the modified circuit diagram of our design in Fig. 3 to save the multiplier at the lower left corner. You may just draw it by hand but draw them clearly.
3. Modify your Verilog code of Lab 4 according to the circuit diagram in Task 2 above. Verify its function in ModelSim for at least 5 sets of inputs. If you find that your circuit has bugs, please correct both your circuit diagram and your Verilog code. You should notice that the circuit will not function correctly if a new set of inputs is provided in every clock cycle. How many clock cycles does the circuit need to wait before it can accept another set of inputs? Please do simulation in ModelSim such that sets of inputs are supplied as frequent as possible so as to maximize the throughput of the circuit.
4. Go through the flow of synthesis, and compare the area result based on the same constraints with your pipelined design in Lab 7. Please compare both the versions without retiming and with retiming. Please note that the area saving also depends on how many new registers are introduced. Please also note that the throughput of our design may decrease. Basically, we trade it for area.

Bonus (+25 points): Note that we have totally 3 multipliers and 2 adders in our original design, and now we saved one multiplier. Could you save another multiplier and one of the adders? This will be more complicated. If you make it, please show the circuit diagram in your lab report and the synthesized area comparison based on the same timing constraints.