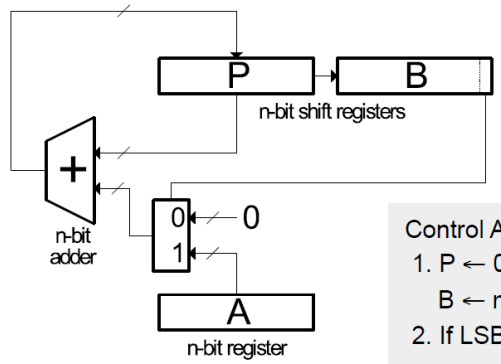
Spring 2013 Discussion 4

1. Shift and add multiplier is shown below



- Sums each partial product, one at a time.
- In binary, each partial product is shifted versions of A or 0.

- Cost α n, T = n clock cycles.
- What is the critical path for determining the min clock period?

Control Algorithm:
1. P ← 0, A ← multiplicand, B ← multiplier
2. If LSB of B==1 then add A to P else add 0
3. Shift [P][B] right 1
4. Repeat steps 2 and 3 n-1 times.
5. [P][B] has product.

How would you modify this to make it work for 2's complement signed multiplication? Describe changes made to the datapath, and design a Moore FSM controller for it.

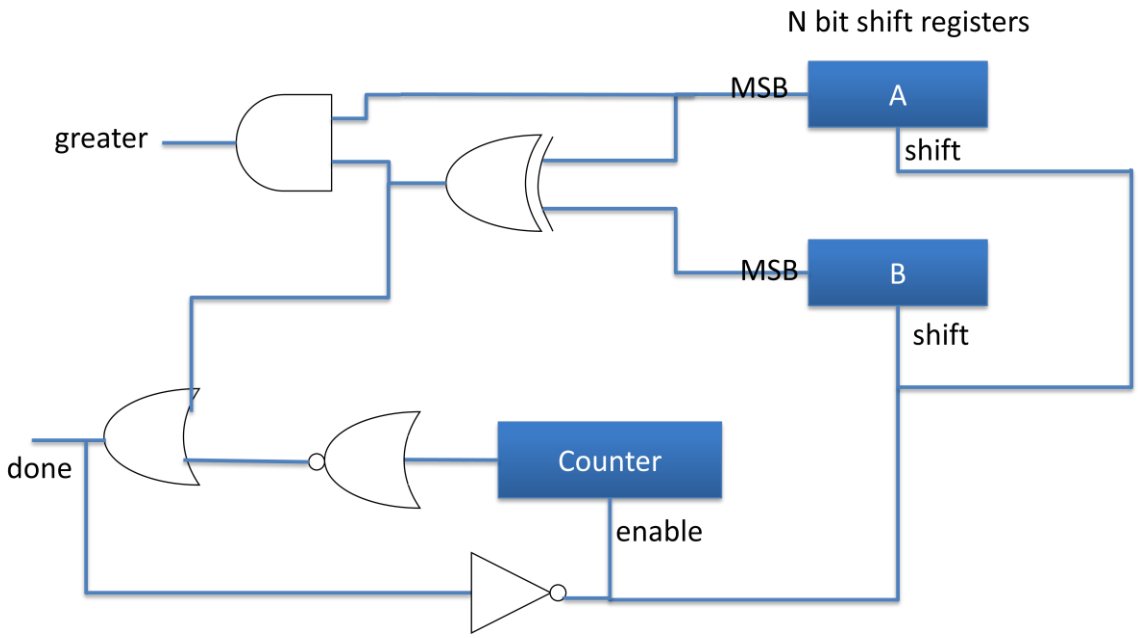
Solution:

To make it work for 2's complement signed multiply, you would need to do subtraction if the multiplier's MSB is 1.

This can be achieved by replacing the n-bit adder with a n-bit adder/subtractor which is controlled by a signal the last bit of B.

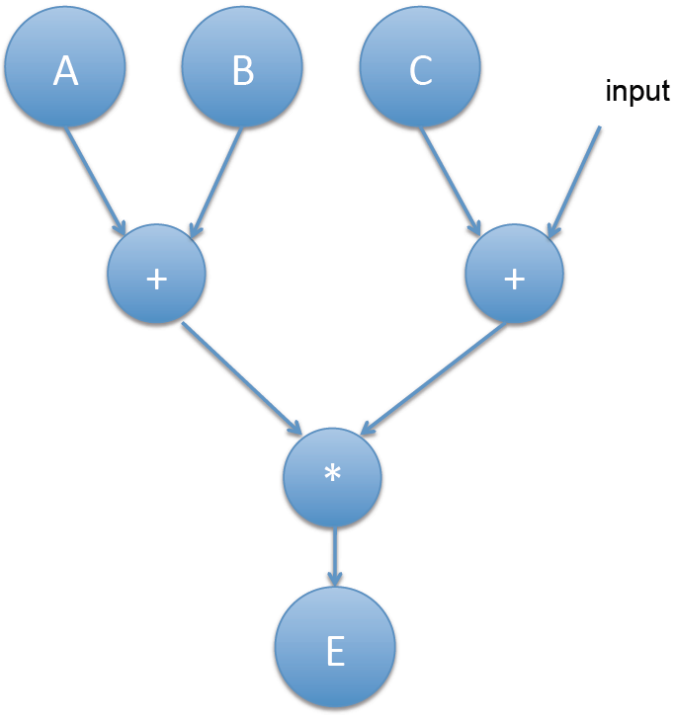
The FSM for the controller would basically check a counter value, when it reaches n-1, it would mux in the last bit of B to determine if the adder should add or subtract.

2. Design a circuit which would do unsigned comparison between 2 N bit numbers. Your circuit should finish the comparison in O(N) time, it would produce 2 output: done and greater, when done is asserted, greater would show if A is greater than B.
(You can assume you are given two N-bit shift registers)



using the diagram above, we would start off by loading A and B into the two shift registers, and N-1 into the Counter. Note the "enable" signal, when asserted, would decrement the counter each cycle, thus we use the counter to check how many bits we have already compared. When counter has value 0, the NOR gate at its output would produce 1, which would assert done, and the counter would be stopped. Also, we are always comparing the most significant bit of A and B, if they are different, then we are done, and the "greater" signal would be 1 only if A's MSB at that moment is 1.Note the A and B shift register would be shifting out their MSB when "done" is not asserted.

3. The graph below shows the computation that happens in one iteration. The circles A, B and C represent LOADs from the memory, the circle E represents a STORE operation to the memory. The other circles are labelled with the computation each represents.You may assume you have two adders, a true dual port memory (each port can support an independent read/write), and a non-pipelined multiplier, also the multiplication would take two cycles to complete while the addition and memory operations only take one cycle.



a. If you have no information about the addresses that each of the LOADs and STOREs use, how would you schedule the operations?

b. After alias analysis is done on the application, it can be safely assumed that the addresses E stores would never be loaded by A, B or C. How would you schedule the operations now? Again, show four iterations.

Mem1	A0	C0				E0	A1	C1
Mem2	B0						B1	
Adder1		A0+B0						
Adder2			C0+in					
Mult					Mult0			

Mem1	A0	C0	A1	C1	A2	C2	A3	C3	A4
Mem2	B0		B1		B2	E0	B3	E1	B4
Adder1		A0+B0		A1+B1		A2+B2		A3+B3	
Adder2			C0+in		C1+in		C2+in		C3+in
Mult				M0	M0	M1	M1	M2	M2

4. Imagine you are given a specification for a circuit that has three inputs, clk, reset, and input and an output, output. Your job is to design an FSM that detects the sequences 01 and 10. The FSM continuously inspects its input and generates a 1 at is output when the input sequence 01 or 10 has been detected,otherwise it outputs a 0. Sequences can overlap. For instance, the input sequence 0101 will output a pulse for three consecutive cycles.

a. Show the state transisiton diagrams and the state transition tables for both the Moore implementations of this circuit.
b. Show the gate level diagrams of the FSM using one-hot encoding.