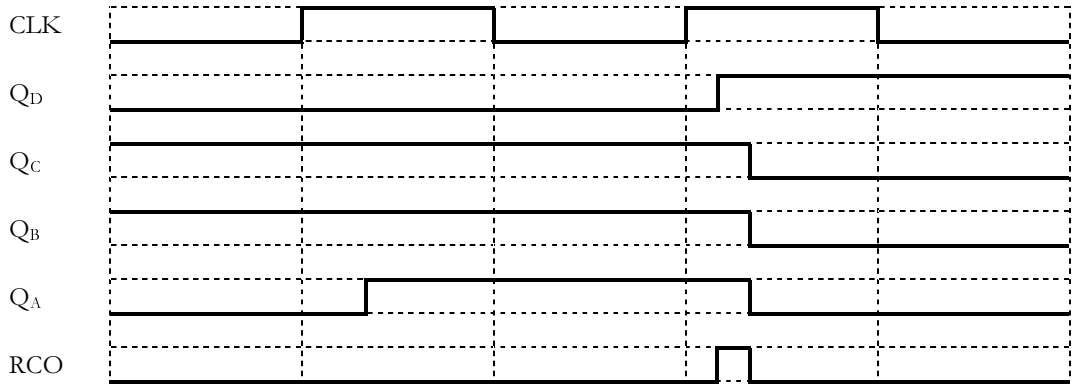
Solutions by Chris Eng, cje@cory.eecs.berkeley.edu

Problem 1 (20 pts)

Looking at the schematic for the x74_163 component in the Xilinx manual, we see that the logic equation given for the ripple carry output is RCO = $Q_D Q_C Q_B Q_A$ENT. Examining the truth table for the counter, we see that the counter only increments when ENT is high, so we are interested in looking only at Q_D through Q_A.

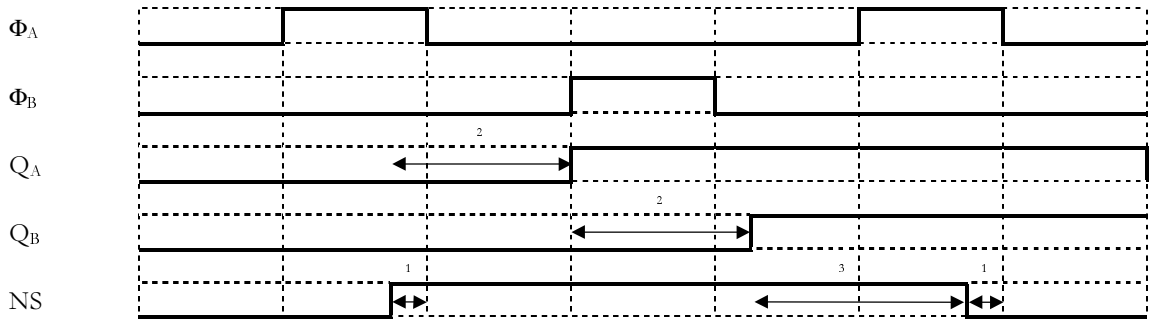
How could glitches be generated by this counter if the four inputs to the AND gate arrived at different times? For example, what if Q_D were to arrive before Q_C, Q_B, and Q_A on the transition from 0111 → 1000? For a short period of time, the AND gate would see all high bits and it would output a 1 momentarily.



Notice that RCO glitches high during the short period in which all four Q bits are high. If this signal were used to drive the clock input of another counter, that counter would produce incorrect results. The counter being driven should only be clocked once for each 16 cycles of the x74_163. With the glitch shown above, however, the counter would actually be clocked twice in every 16-cycle period.

Problem 2 (30 pts)

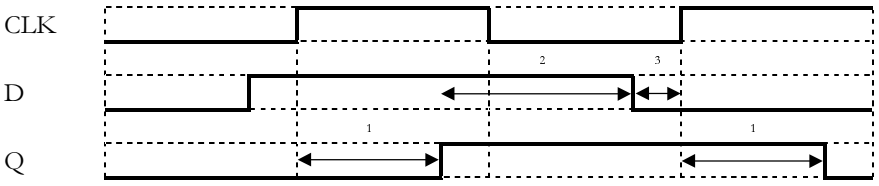
a. For simplicity, we will assume that the two non-overlapping clocks Φ_A and Φ_B have a pulse width of T/4; that is, the time period during which either clock is high is equal to the time period when both clocks are low (see timing diagram).



The labeled delays, not drawn to scale, correspond to: 1) t_{hold} = 5 ns, 2) t_{latch} = 20 ns, and 3) t_{NS} = 20 ns. We see that the critical delay occurs between the rising edge of Φ_B and the falling edge of Φ_A. During this time, a value must be latched into Q_B (20 ns) and the next state decoder must calculate its output (20 ns), which must be ready at least t_{hold} (5 ns) before Φ_A goes low. Since this duration is equal to ¾ of the clock period, we can write $\frac{3}{4}T \geq 45$ ns, or $T \geq 60$ ns. Based on this, we find that $f \leq 16.67$ MHz.

The reason we need two latches instead of one is to ensure that there is no race condition. If the second latch were not present, the output Q_A could propagate through the next state decoder and through the latch again while the clock was still high. Adding a second latch solves this problem by ensuring that the next state is not computed until Φ_A is low *(see Katz p. 304).*

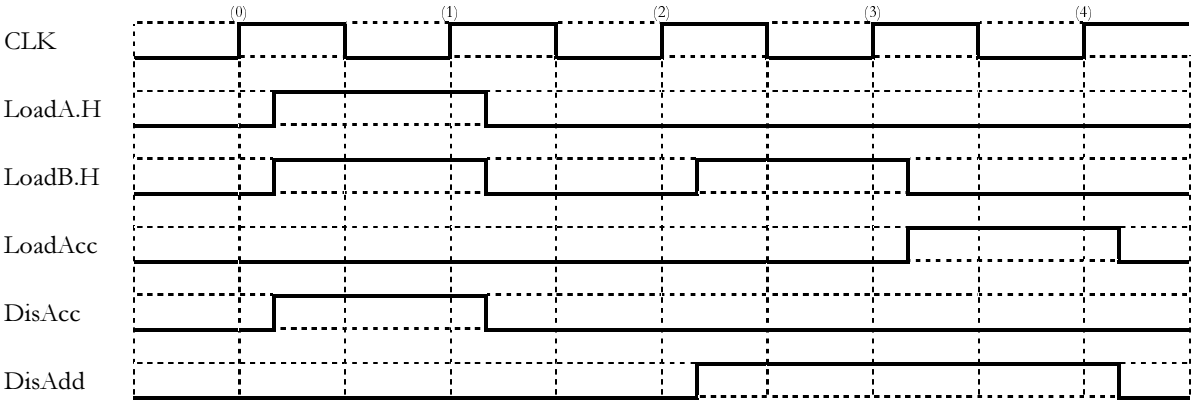
b. The state machine that uses an edge-triggered state register would have the following timing diagram.



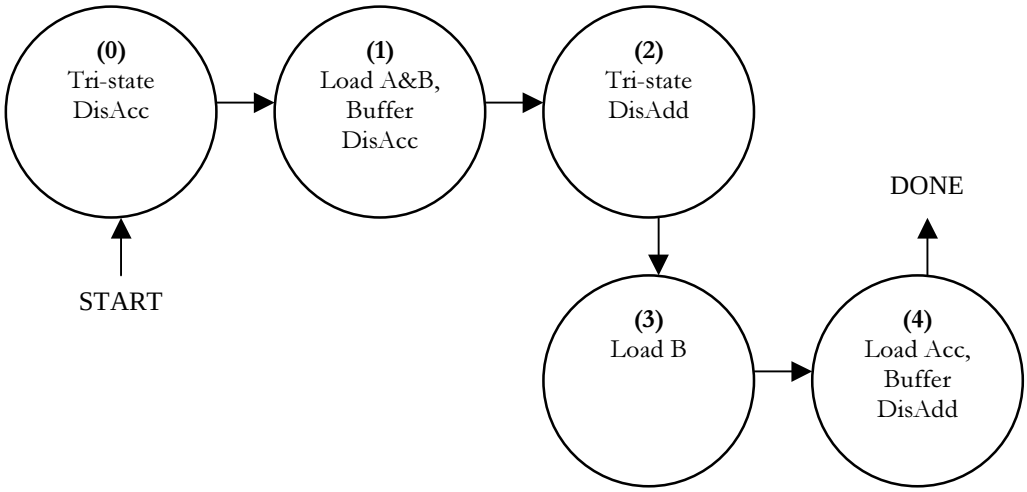
The labeled delays in this diagram, not drawn to scale, correspond to: 1) t_{CLKO} = 20 ns, 2) t_{NS} = 20 ns, and 3) t_{setup} = 5 ns. Thus, between rising edges of the clock, at least 30 ns must elapse. This can be written as $T \geq 45$ ns, or $f \leq 22.22$ MHz.

Problem 3 (50 pts)

a. The steps we want to carry out in this operation are as follows: 1) load A and B with the contents of the accumulator, 2) perform the add and store the result in B, 3) perform the second add and store the final result back in the accumulator. The rising edges of the clock are numbered 0 through 4 for use in part b.



b. Here is the state diagram for the controller. In each circle, the numbers correspond to each rising edge of the clock (as labeled in the timing diagram), and the words correspond to the actions performed at that particular clock edge.



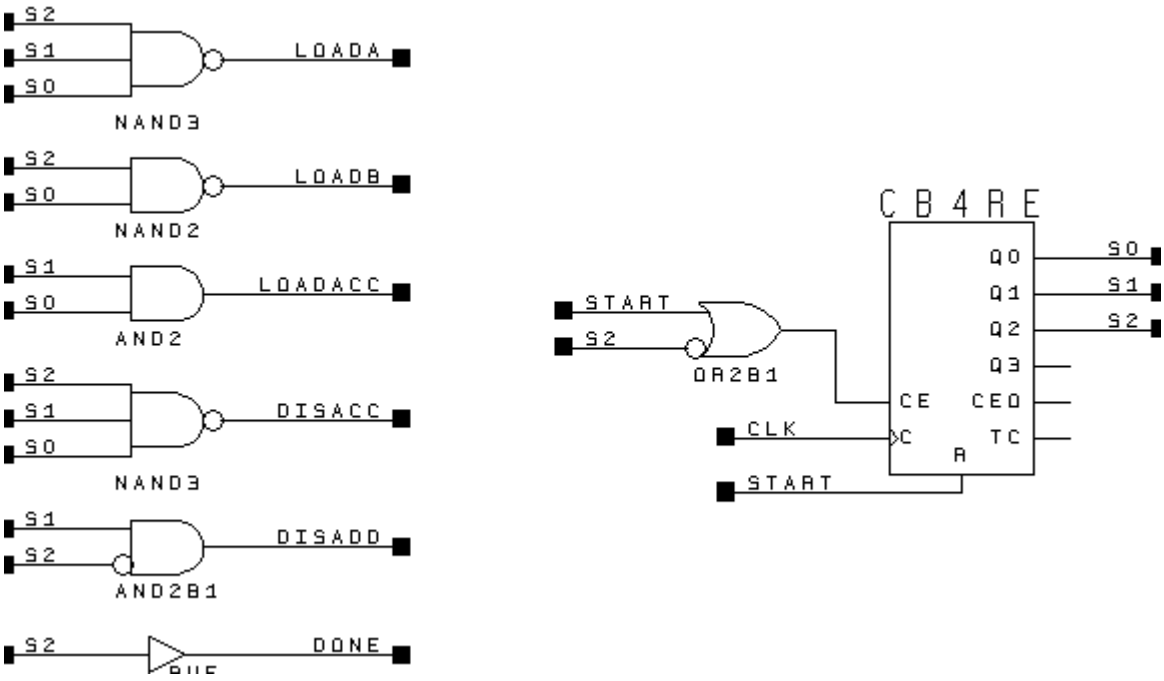
We will map states 0 through 4 to a 3-bit state (i.e. 000, 001, 010, 011, 100). The values of the control signals in each of the states are shown here.

S_2	S_1	S_0	LoadA	LoadB	LoadAcc	DisAcc	DisAdd	Done
0	0	0	1	1	0	1	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	1	0
0	1	1	0	0	1	0	1	0
1	0	0	0	0	0	0	0	1
1	0	1	DON'T CARE					
1	1	0						
1	1	1						

Note that the last three rows of the table are not used because they do not correspond to actual states in the state diagram. In addition, we have added a Done signal, which is asserted on rising edge #4, the same time when the product is latched into the accumulator.

We can express the 6 control signals in terms of the state bits as follows.

$$LoadA = \overline{S_2}\,\overline{S_1}\,\overline{S_0} \qquad LoadB = \overline{S_2}\,\overline{S_0} \qquad LoadAcc = S_1 S_0$$

$$DisAcc = \overline{S_2}\,\overline{S_1}\,\overline{S_0} \qquad DisAdd = \overline{S_2} S_1 \qquad Done = S_2$$

The final circuit is shown below. The START signal resets the counter to 000, cycles through the states, and stops when it reaches 100, asserting DONE at that point. There is a BUF component between S2 and DONE because ViewDraw does not allow the same net to be labeled with two different names.