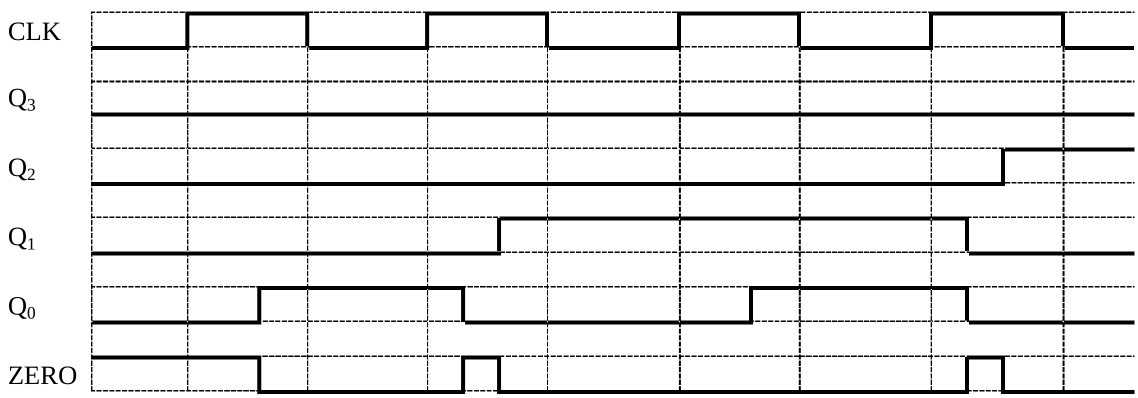
Professor Fearing **CS 150 Fall 1998**

Problem Set 5 Solutions

by Nadeem Aboobaker (nma@cory)

1. Glitches (20 pts)

Glitches can occur when Q_3, Q_2, Q_1, or Q_0 don't change at exactly the same times. For a short time during these transitions, all four signals could be low, causing ZERO to be asserted. ZERO is supposed to be asserted when $Q_3Q_2Q_1Q_0$ = 0000, but there is also a chance it will be asserted on the following transitions**: 0001→0010, 0011→0100, and 0111→1000**. The following diagram shows Q_3, Q_2, Q_1, and Q_0 signals that are delayed and the glitches that occur on ZERO at **the 0001→0010** and **0011→0100** transitions.

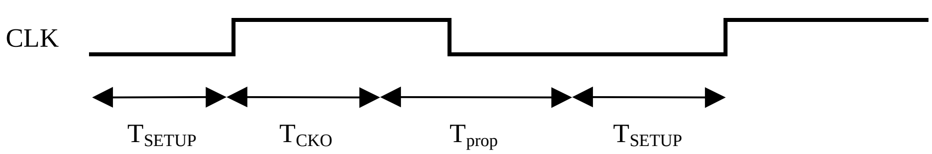


Ideally, ZERO should be asserted once every time the counter is at 0000. As shown in the above diagram, there is a chance that ZERO will be asserted additional times while it is counting from 0000 to 1111. If ZERO were used to drive the asynchronous reset input of another counter, the second counter would reset at the wrong times. Glitches on an asynchronous reset can lead to problems in your designs, so always use synchronous parts.

2. Clock Skew (30 pts)

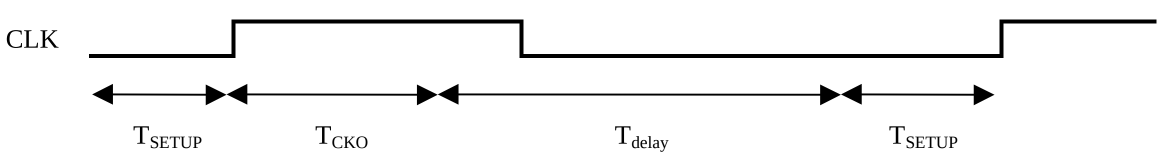
a.) Finding the maximum frequency requires finding the minimum period, which is done by adding the timing constraints and delays for each flip flop and taking the maximum of the two. The following timing diagrams (not to scale) show the timing constraints and how to calculate the clock period.

Flip Flop 0:



$T_{setup} + T_{CKO} + T_{prop}$ = 4.0 ns + 3.7 ns + 6 ns = 13.7 ns

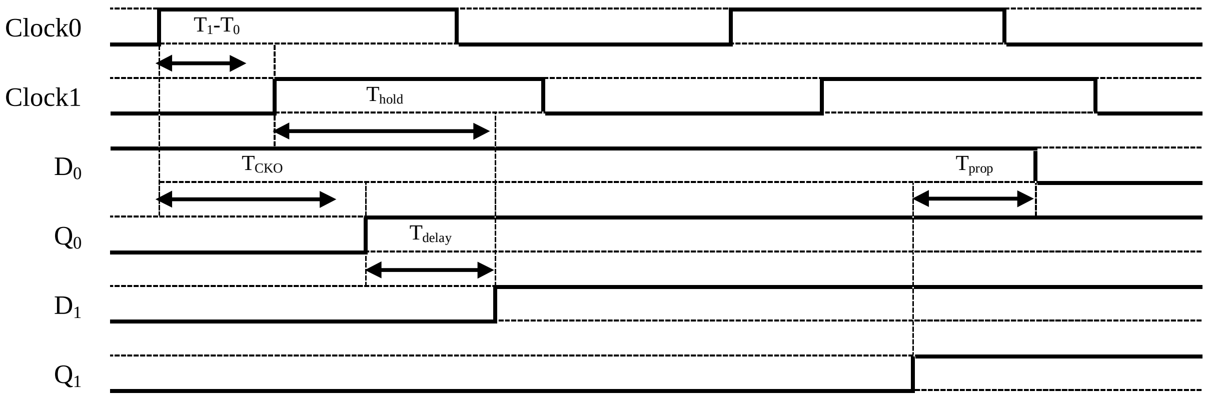
Flip Flop 1is done the same way as Flip Flop 0, except use T_{delay} instead of T_{prop}:



$T_{setup} + T_{CKO} + T_{delay}$ = 4.0 ns + 3.7 ns + 20 ns = 27.7 ns

So the minimum period is 27.7ns. Since *frequency* = *period* $^{-1}$, max frequency = 27.7ns^{-1} = 36.6 MHz.

b.) The maximum $T_1 - T_0$ is the maximum difference in delays to the two Flip Flops before the timing constraints are violated. In this case, the hold time on Flip Flop 1 will be violated if the clock on Flip Flop 1 is delayed too much compared to the delay to the clock on Flip Flop 0.

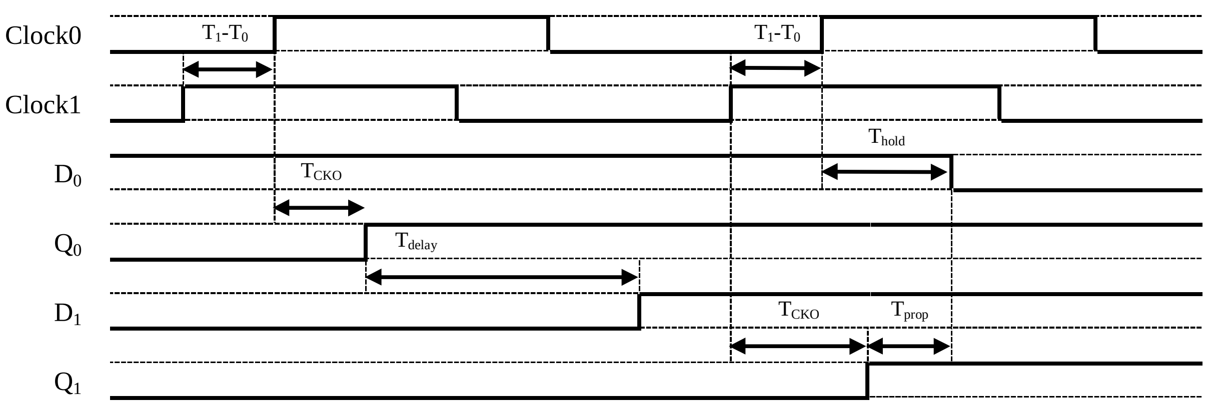


In the case in the above diagram, there are no violations of any timing constraints. But if T_1-T_0 becomes any larger, D_1 will be changing during Flip Flip 1's hold time. To calculate max T_1-T_0:

$T_{CKOmin} + T_{delay} > T_{hold} + T_1\text{-}T_0$
$T_{CKOmin} + T_{delay} - T_{hold} > T_1\text{-}T_0$
$0ns + 20ns – 0ns > T_1\text{-}T_0$

$\max(T_1\text{-}T_0) = 20ns$

c.) The minimum $T_1–T_0$ is similar to maximum in part c.), except in this case, the hold time on Flip Flop 0 will be violated if T_1 occurs is too much smaller than T_0. So $T_1–T_0 < 0$.



In this timing diagram, D0 will violate the hold time of Flip Flop 0 if Clock 1 occurs earlier:

$-(T_1–T_0) + T_{hold} > T_{CKOmin} + T_p$
$-(T_1–T_0) > T_{CKOmin} + T_p - T_{hold}$
$-(T_1–T_0) > 0ns + 6ns – 0ns$

$\min(T_1\text{-}T_0) = 6ns$, meaning Clock 1 can occur 6ns before Clock 0.

d.) To prevent delays on critical signals, the Xilinx provides BUFGS and BUFG. Each has four lines that are routed throughout the chip and can be used to prevent indeterminate delays on the clock line due to random routing.

3. Controller Design and Tri-state busses (50pts)

STATE #: OUTPUTS

0: LOADA, ENACC

1: LOADSUM, ENACC

2: TRISTATE BUS

3: LOADA, ENADD

4: LOADSUM, ENADD

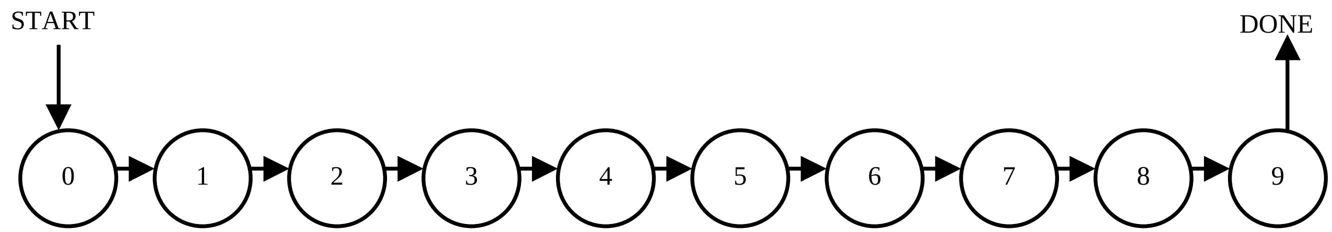
5: LOADA, ENADD

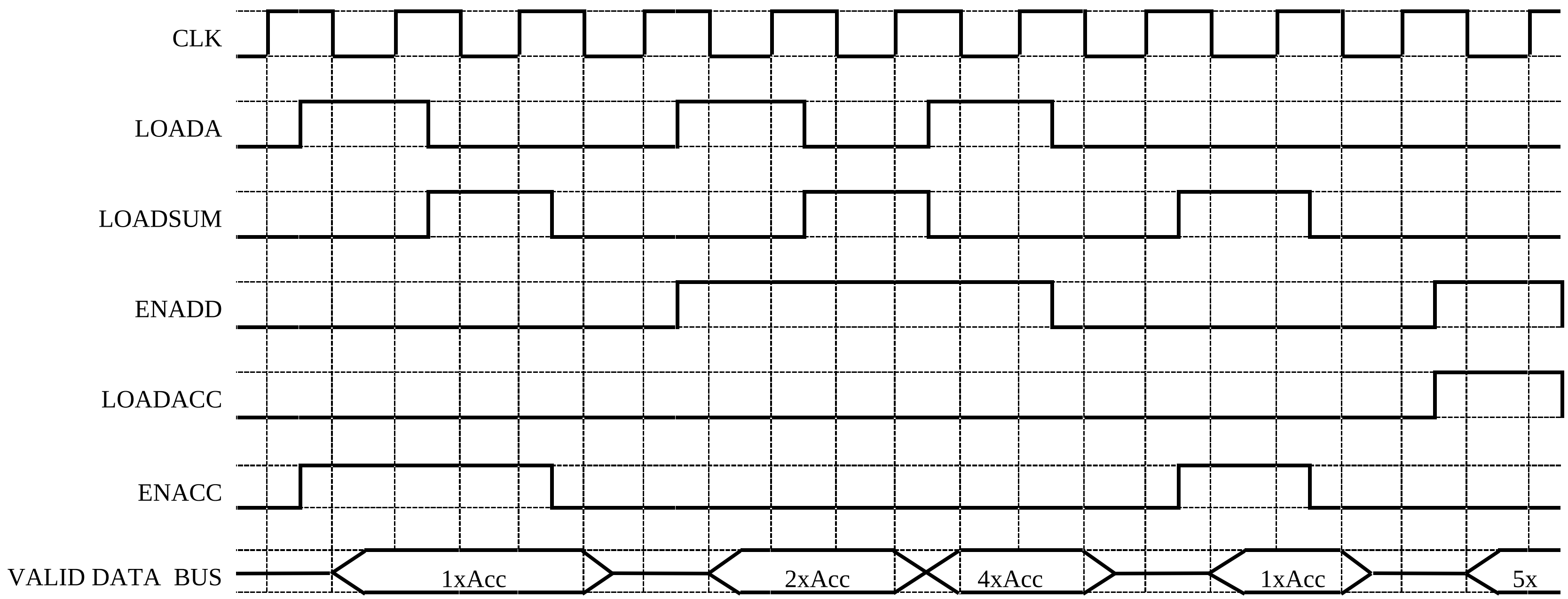
6: TRISTATE BUS

7: LOADSUM, ENACC

8: TRISTATE BUS

9: LOADACC, ENADD





a.) The steps shown on the last page are as follows: 0.) Load A with the contents of the Accumulator, 1.) perform the addition and store 2xAccumulator in the Sum register, 2.) idle, 3.) Load A with the contents of the Sum register, 4.) perform the addition and store 4xAccumulator in the Sum register, 5.) Load A with the contents of the Sum register, 6.) idle, 7.) perform the addition of 4xAccumulator with value in the Accumulator and store 5xAccumulator in the Sum register, 8.) idle, 9.) Load the accumulator with the value from the Sum register. This can be re-written in RTN form as:

0: ACC → A
1: ACC + A → SUM
2: idle
3: SUM → A
4: SUM + A → SUM
5: SUM → A
6: idle
7: ACC +A → SUM
8: idle
9: SUM → ACC

A less obvious way of doing the problem in only 6 states is to take advantage of the latching of values at the edge of the clock:

0: ACC → A
1: ACC + A → SUM
2: idle
3: SUM + A → SUM, SUM → A
4: SUM + A → SUM
5:SUM → ACC

b.) The state diagram for the controller is on the last page along with a table that shows the outputs corresponding to each state. Since the outputs are defined only by the state, this is a Moore machine. The following is a state table of states and outputs:

S3	S2	S1	S0	LOADA	LOADSUM	ENADD	LOADACC	ENACC	DONE
0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	1	0	1	0	0	0
0	1	1	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	1	0
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1	0	0
1	0	1	0	0	0	0	0	0	1
1	0	1	1	DON'T CARE SINCE STATES GO TO ONLY 1010					
1	1	0	0						
1	1	0	1						
1	1	1	0						
1	1	1	1						

Here are the logic equations taken from the above state table:

$LOADA = \overline{S_3}(\overline{S_2}\,\overline{S_1}\,\overline{S_0} + \overline{S_2}\,S_1\,S_0 + S_2\,\overline{S_1}\,S_0)$

$LOADACC = S_3\,\overline{S_2}\,\overline{S_1}\,S_0$

$LOADSUM = \overline{S_3}(\overline{S_2}\,\overline{S_1}\,S_0 + S_2\,\overline{S_1}\,\overline{S_0} + S_2\,S_1\,S_0)$

$ENACC = S_3(\overline{S_2}\,\overline{S_1}\,\overline{S_0} + \overline{S_2}\,\overline{S_1}\,S_0 + S_2\,S_1\,S_0)$

$ENADD = S_3\,\overline{S_2}\,\overline{S_1}\,S_0 + \overline{S_3}(\overline{S_2}\,S_1\,S_0 + S_2\,\overline{S_1}\,\overline{S_0} + S_2\,\overline{S_1}\,S_0)$

$DONE = S_3\overline{S_2}S_1\overline{S_0}$

The final design is shown below. The START signal resets the counter to 0000, and the counter then cycles through the states until state 1010. At this point the counter stops and DONE is asserted when 5 times the original accumulator value is latched into the accumulator register. A 4:16 decoder is used to simplify the logic and make it easily modifiable. Signals D0 – D10 on the output of the decoder are for State0 – State10.