Name: Email: ID#

UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering and Computer Sciences
EECS150 Fall 2001

Prof. Subramanian

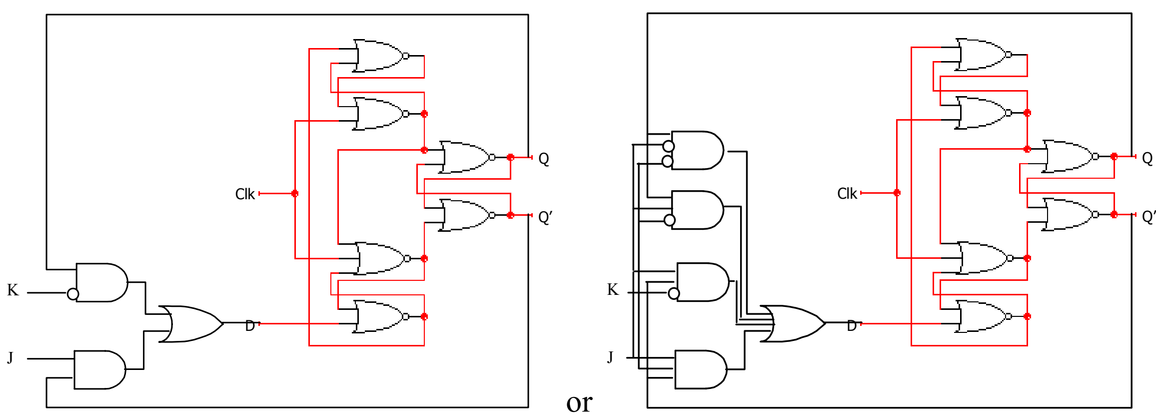
# Midterm I

1) You are given a negative edge-triggered D flip-flop as shown on page 1-19 of the notes on sequential logic.

   a) Design the combinational logic necessary to convert this flip-flop into a negative edge-triggered J-K flip-flop. Leave your solution in sum-of-products form. Write out the equation for the combinational logic block, and draw the block connected to the FF below.

| *J* | *K* | *Q* | *D* |
|---|---|---|---|
| *0* | *0* | *0* | *0* |
| *0* | *0* | *1* | *1* |
| *0* | *1* | *0* | *0* |
| *0* | *1* | *1* | *0* |
| *1* | *0* | *0* | *1* |
| *1* | *0* | *1* | *1* |
| *1* | *1* | *0* | *1* |
| *1* | *1* | *1* | *0* |

Equation: $D = \Sigma m(1,4,5,6) = J'K'Q + JK'Q' + JK'Q + JKQ'$ *( which, FYI* $= JQ' + K'Q$ *)*




   b) Assume the setup and hold times for the D-FF above are 20ns and 10ns respectively. What is the minimum propagation delay for the D-FF for it to meet correct timing criteria? Why?

$t_P > t_H \Rightarrow t_P > 10ns$. *In this way, when we cascade two or more of these D-FF's, the output of one FF can be connected as an input of another because it is stable for at least the hold time after a clock edge.*

c) Suppose the propagation delay for the D-FF is 20ns and the clock frequency is 10MHz. What is the maximum delay per gate allowed in the sum-of-products combinational logic block that you designed? Give a reason for your answer, and show your calculations. Assume that all gates in your combination block have identical delay. Also assume that J' and K' are available for free (i.e., no delay in inverting the J and K signals).

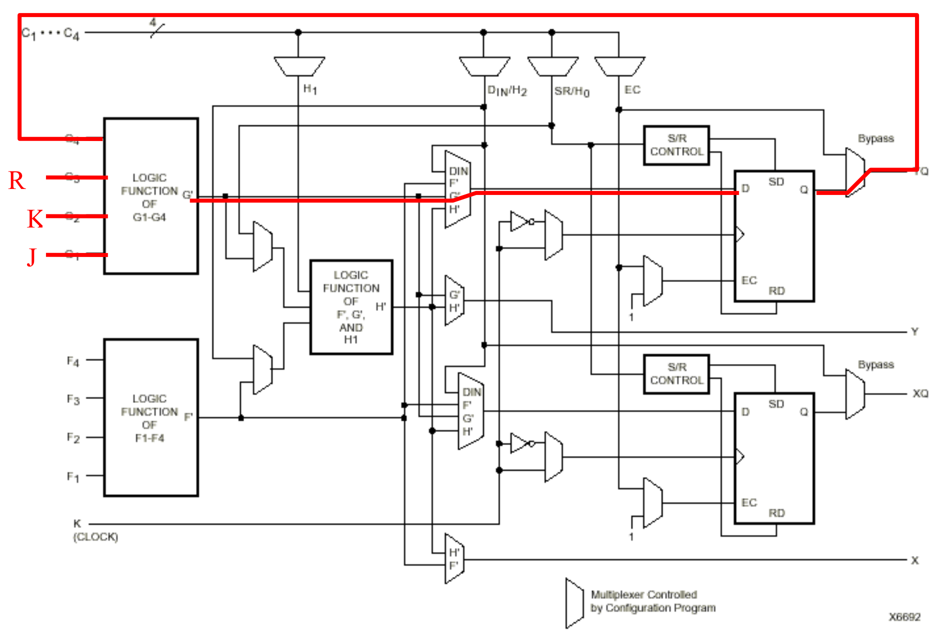
*The delay from the clock edge to Q,Q' is $t_P$=20ns; the delay from Q,Q' to J,K is equal to 2 gate delays. Then $T-t_{su} > t_P+2t_d$,*

*$\Rightarrow t_d < (T- t_{su} - t_P)/2 = 30$ ns*

d) Recalculate the sum-of-products equation describing the combinational logic block assuming that you must now also implement a synchronous reset signal.

*D = R'(J'K'Q + JK'Q' + JK'Q + JKQ')*

e) Suppose you are to implement the J-K flip-flop with synchronous reset from above in a Xilinx CLB. Mark the active blocks and multiplexer paths on the diagram below.




f) How many J-K FF's can be implemented per CLB?

*Two J-K FF's can be implemented per CLB.*

g) Suppose you wanted to implement a J-K FF with synchronous Set and Reset using a full sum-of-products form. Now, how many J-K FF's can be implemented per CLB?

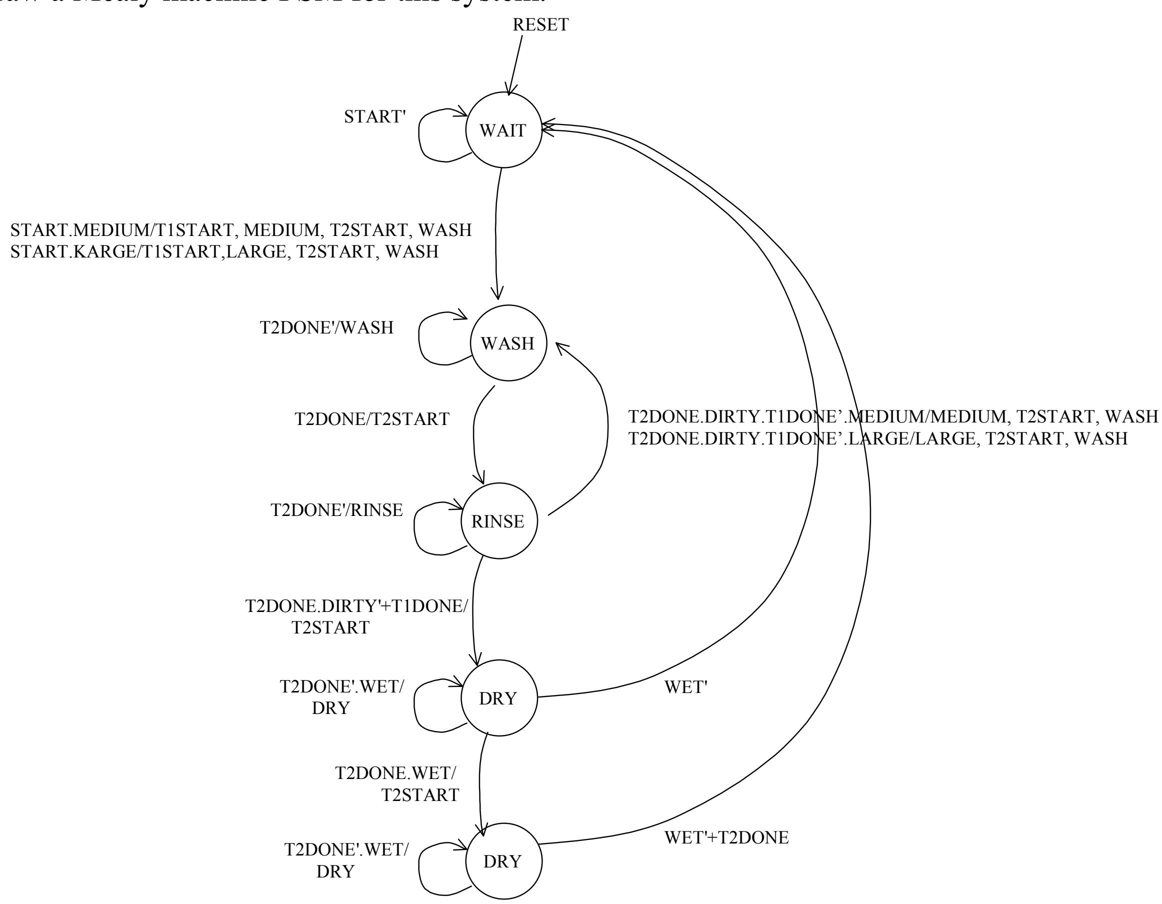
*Now the combinational function has 5 variables, then only one J-K FF can be implemented per CLB.*

Name: Email: ID#

2) You are implementing the brains for a “smart” washing machine. The washing machine works in the following manner:
   - When you press “Start” after loading in the clothes, the washing machine determines the load size (Medium / Large) and then dispenses the appropriate amount of water and detergent.
   - The machine then washes the clothes for 10 minute.
   - The machine then rinses the clothes for 10 minutes. If the effluent is dirty at the end of the rinse, the machine repeats the wash + rinse cycle, but for no more than a total of 3 cycles.
   - The machine then spin-dries the clothes until it detects no water discharge, but for not more than 20 minutes.

| Inputs: | Output: |
|---|---|
| START Button | MEDIUM water + soap dispense |
| MEDIUM load sensor | LARGE water + soap dispense |
| LARGE load sensor | WASH cycle actuator |
| DIRTY effluent sensor | RINSE cycle actuator |
| WET water discharge sensor | DRY cycle actuator |
| T1DONE = 60 min timer | T1START 60 min timer start |
| T2DONE = 10 min timer | T2START 10 min timer start |

Draw a Mealy machine FSM for this system.



Name: Email: ID#

3) ASCII, as you know, is a common data representation format. Write the state transition table for an ASCII hexadecimal counter; i.e., a counter than counts from 0-15 and outputs the results as 0-9, A-F in ASCII (NOTE: 0-9 is $48_{10}$-$57_{10}$, A-F is $65_{10}$-$70_{10}$). Assume the data is stored in an 8-bit wide D-FF register.

| Current State | | | | | | | | | | Next State | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base10 | Hex | Register Contents | | | | | | | | Base10 | Hex | Register Contents | | | | | | | |
| | | Q8 | Q7 | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | | | N8 | N7 | N6 | N5 | N4 | N3 | N2 | N1 |
| 0 | 0 | *0* | *0* | *1* | *1* | *0* | *0* | *0* | *0* | *1* | *1* | *0* | *0* | *1* | *1* | *0* | *0* | *0* | *1* |
| 1 | 1 | *0* | *0* | *1* | *1* | *0* | *0* | *0* | *1* | *2* | *2* | *0* | *0* | *1* | *1* | *0* | *0* | *1* | *0* |
| 2 | 2 | *0* | *0* | *1* | *1* | *0* | *0* | *1* | *0* | *3* | *3* | *0* | *0* | *1* | *1* | *0* | *0* | *1* | *1* |
| 3 | 3 | *0* | *0* | *1* | *1* | *0* | *0* | *1* | *1* | *4* | *4* | *0* | *0* | *1* | *1* | *0* | *1* | *0* | *0* |
| 4 | 4 | *0* | *0* | *1* | *1* | *0* | *1* | *0* | *0* | *5* | *5* | *0* | *0* | *1* | *1* | *0* | *1* | *0* | *1* |
| 5 | 5 | *0* | *0* | *1* | *1* | *0* | *1* | *0* | *1* | *6* | *6* | *0* | *0* | *1* | *1* | *0* | *1* | *1* | *0* |
| 6 | 6 | *0* | *0* | *1* | *1* | *0* | *1* | *1* | *0* | *7* | *7* | *0* | *0* | *1* | *1* | *0* | *1* | *1* | *1* |
| 7 | 7 | *0* | *0* | *1* | *1* | *0* | *1* | *1* | *1* | *8* | *8* | *0* | *0* | *1* | *1* | *1* | *0* | *0* | *0* |
| 8 | 8 | *0* | *0* | *1* | *1* | *1* | *0* | *0* | *0* | *9* | *9* | *0* | *0* | *1* | *1* | *1* | *0* | *0* | *1* |
| 9 | 9 | *0* | *0* | *1* | *1* | *1* | *0* | *0* | *1* | *10* | *A* | *0* | *1* | *0* | *0* | *0* | *0* | *0* | *1* |
| 10 | A | *0* | *1* | *0* | *0* | *0* | *0* | *0* | *1* | *11* | *B* | *0* | *1* | *0* | *0* | *0* | *0* | *1* | *0* |
| 11 | B | *0* | *1* | *0* | *0* | *0* | *0* | *1* | *0* | *12* | *C* | *0* | *1* | *0* | *0* | *0* | *0* | *1* | *1* |
| 12 | C | *0* | *1* | *0* | *0* | *0* | *0* | *1* | *1* | *13* | *D* | *0* | *1* | *0* | *0* | *0* | *1* | *0* | *0* |
| 13 | D | *0* | *1* | *0* | *0* | *0* | *1* | *0* | *0* | *14* | *E* | *0* | *1* | *0* | *0* | *0* | *1* | *0* | *1* |
| 14 | E | *0* | *1* | *0* | *0* | *0* | *1* | *0* | *1* | *15* | *F* | *0* | *1* | *0* | *0* | *0* | *1* | *1* | *0* |
| 15 | F | *0* | *1* | *0* | *0* | *0* | *1* | *1* | *0* | *16* | *0* | *0* | *0* | *1* | *1* | *0* | *0* | *0* | *0* |

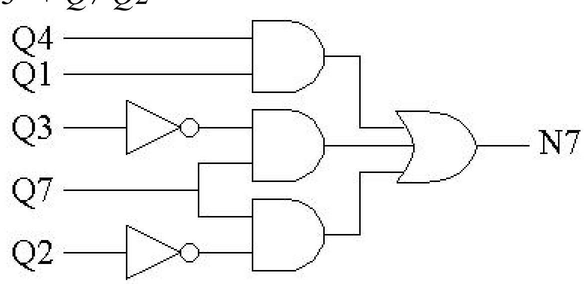
a) Determine the next state functions for the following registers in abbreviated SoP form:

i) N8 = *0*

ii) N7*(Q8, Q7, Q6, Q5, Q4, Q3, Q2, Q1) = Σ m(57, 65, 66, 67, 68, 69)*

b) Simplify N7 to its simplest form, and draw the combinational logic block as it would connect to the D input of FF7.

*N7 = Q4 Q1 + Q7 Q3' + Q7 Q2'*



Name: Email: ID#

c) What is the minimum # of FFs you would need to implement the counter in the following schemes. Give reasons for your answers:

i) One-hot

*16 FFs. In the one-hot state assignment scheme, each state corresponds to a flip-flop, such that a flip-flop is high whenever the FSM is in the corresponding state.*

ii) Sequential

*4 FFs. Each of the sixteen states can be represented as a four-bit binary number. We could implement a sequential assignment scheme by assigning each state to its 4-bit binary representation. For example, state 4xH would be assigned as 0100, state BxH as 1011, etc.*

4) You are to implement a 2 bit shift register using D-FFs with some special features. The register has 3 modes of operation determined by two input bit (C1) – 00, 01, 10, 11.
1) 00: The shift register right shifts, taking input from LIN->Q2->Q1
2) 01: The shift register left shifts, taking input from RIN->Q1->Q2
3) 10: The shift register loops back, i.e., Q2->Q1, Q1->Q2
4) 11: The shift register toggles all its bits, i.e., all ones become zeroes and vice versa

a) Draw the FSM diagram for this shift register