ECE 370: Digital Systems-Logic Design

Sample Test: Chapter 8 & Final Exam
Spring 2005

Sequential and Combinational Circuit Design

1. What is the difference between a Mealy and Moore Finite State Machine (FSM)? For each machine, give an expression for the *next state(s)* and *output(s)* as a function of the *input(s)* and the *present state(s)*.

2. For *n* state variables, provide an expression that relates the number of state variables, *n*, to the number of possible states *f(n)*. Given *f(n)*, what is the size of the state register needed to implement the FSM.

3. In general, why is a Mealy FSM termed to be more "asynchronous" than a Moore FSM.

4. Describe three different heuristics for deciding how to make a state assignment,

5. Suppose a Mealy FSM is implemented with three flip-flops, two inputs, and six asynchronous outputs. Consider the *complete* state diagram for this machine (that is, there are no don't cares). Answer the following questions:

   (a) What are the minimum and maximum numbers of states in the state diagram?
   (b) What are the minimum and maximum numbers of transition arrows starting at a particular state?
   (c) What are the minimum and maximum numbers of transition arrows that can end in a particular state?
   (d) What are the minimum and maximum numbers of different binary patterns that can be displayed on the outputs?

6. Suppose a Moore FSM has five flip-flops, three inputs, and nine outputs. Answer the following questions:

   (a) What are the minimum and maximum numbers of states in the state diagram?
   (b) What are the minimum and maximum numbers of transition arrows starting at a particular state?
   (c) What are the minimum and maximum numbers of transition arrows that can end in a particular state?
   (d) What are the minimum and maximum numbers of different binary patterns that can be displayed on the outputs?

7. A reset-dominant S-R Flip Flop is a type of SR Flip-Flop without the "forbidden" state. In the forbidden state, the reset-dominant SR Flip Flop resets the next state output.

   (a) What is the truth table for a reset-dominant S-R Flip-Flop
   (b) Derive the characteristic equation of this Flip-Flop.
   (c) Derive the excitation table for this Flip-Flop.
   (d) Design a 2-bit synchronous counter using this Flip-Flop

8. Design a system that has one input, *x*, and one output, *z*. When the system detects three 0s in a row, it asserts the output. Overlapping sequences are allowed.

   (a) Provide a State Graph implementing the above design specification. *Give a brief description of the meaning of each state*.
   (b) Provide a State Table using any kind of state of assignment you wish.
   (c) Implement this design with an even number of D Flip-Flops and an odd number of T Flip-Flops using Shannnon's expansion algebraically.
   (d) Provide a detailed schematic of your final design.

9. Design a system that has one input, *x*, and one output, *z*. When the system detects the bit-string 101 **or** bit-string 110, the output is asserted. Overlapping sequences are allowed.

   (a) Provide a State Graph implementing the above design specification. *Give a brief description of the meaning of each state*.
   (b) Provide a State Table using any kind of state of assignment you wish.
   (c) Implement this design with an even number of D Flip-Flops and an odd number of T Flip-Flops using Shannon's expansion with K-Maps.
   (d) Provide a detailed schematic of your final design.

10. A digital system recognizes the 11011 bit-string **and** detects an even number of 1s at its input. One input, *x,* exists and two outputs, *y,* and *EVEN* exits in the system. When detecting the bit-string, overlapping sequences are allowed.

    (a) Provide a State Graph implementing the above design specification. *Give a brief description of the meaning of each state*.

(b) Provide a State Table using any kind of state of assignment you wish.
(c) Implement this design with an even number of J-K Flip-Flops and an odd number of T Flip-Flops. Implement this design using a POS implementation and two binary decoders of appropriate size.
(d) Provide a detailed schematic of your final design.

10. Design a digital system representing a 4-bit parity checker with 2 inputs, $EVEN / \overline{ODD}$ and IN. The parity checker is an even parity checker if $EVEN / \overline{ODD}$ is logic HIGH and an ODD parity checker if $EVEN / \overline{ODD}$ is logic LOW. The system contains one output, ERROR, that is asserted when a parity error occurs, depending on the $EVEN / \overline{ODD}$ signal. (*An ODD parity error occurs if the number of 1's at the IN input is even, and an EVEN parity error occurs if the number of 1's at the IN input is odd).*

(a) Provide a State Graph implementing the above design specification. *Give a brief description of the meaning of each state*.
(b) Provide a State Table using any kind of state of assignment you wish.
(c) Implement this design with at least every kind of Flip-Flop and only POS implementations of logic.
(d) Provide a detailed schematic of your final design.

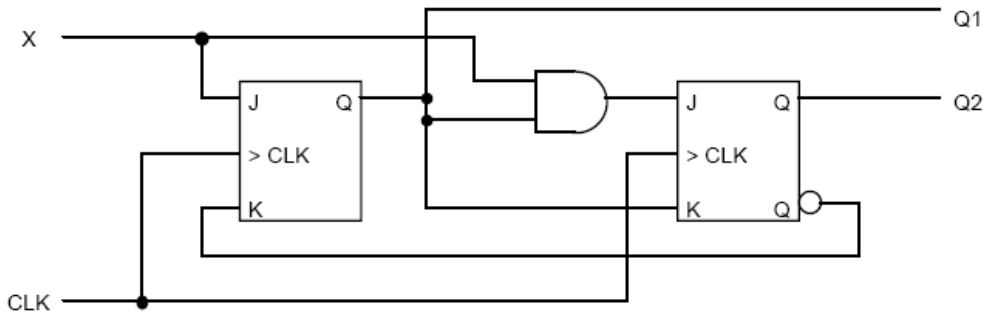
11. A system that controls the flow of cars at a typical toll-booth is described here. The cost of the toll is 30 cents and the booth only accepts *nickels, dimes,* and *quarters*. There are four inputs to the system, *NICKLE, DIME,* and *QUARTER*, and *CHEAT*. CHEAT is an input sensor that detects whether or not a car is attempting to "run away" from the booth and not paying the full 30 cents. The system contains three outputs, *RED, GREEN,* and *YELLOW.* Below are the specifications. Assume that exact change needs to be given!

- *RED* is HIGH until 30 cents is received.
- *YELLOW* is HIGH only when 30 cents have not been received and a car attempts to "run away" from the booth without giving 30 cents.
- *CHEAT* is HIGH only when a car attempts to "run away" without paying the 30 cents.
- *GREEN* is HIGH only when the driver has finished paying 30 cents.

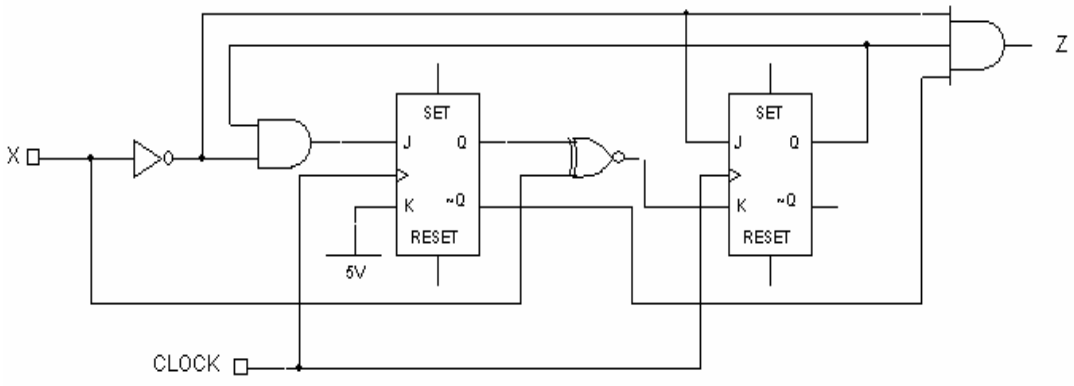
Provide only a state graph for the above design. *Give a brief description of the meaning of each state*.

12. Reverse engineer the following FSMs. Provide the next state logic, generate the state/input/output table, and generate the State Graph for each schematic below. If possible, describe what function each FSM performs.

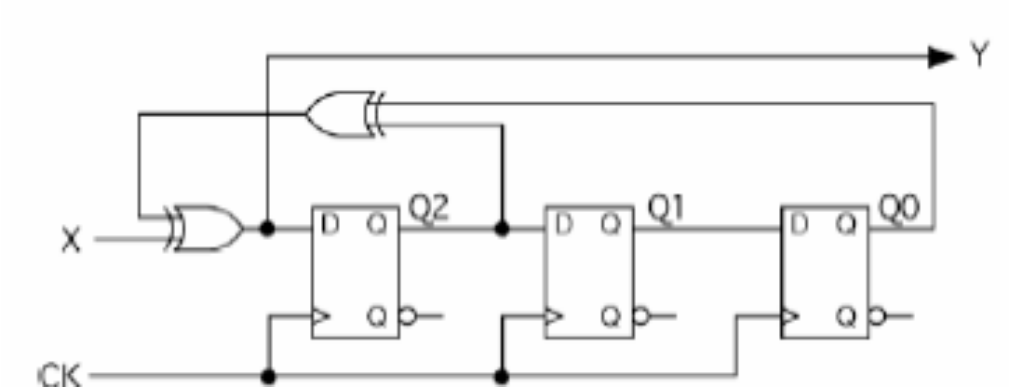
(a)



(b)



(c)



13. Using a 7-segment display, design a "direction run indicator" (similar to that used on VCRs) that sequences the perimeter segments in a *clockwise* fashion if mode control input M=1 (A→B→C→D→E→F→A→B…), or in a *counter-clockwise* fashion if mode control input M=0. Input S should control whether the display is running (S=0) or stopped (S=1); if stopped, the middle segment (G) should be on and the rest should be off. The figure below shows how this "direction run indicator" works. Fully implement this design.