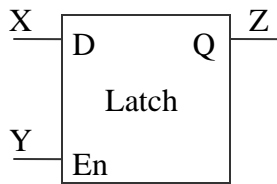
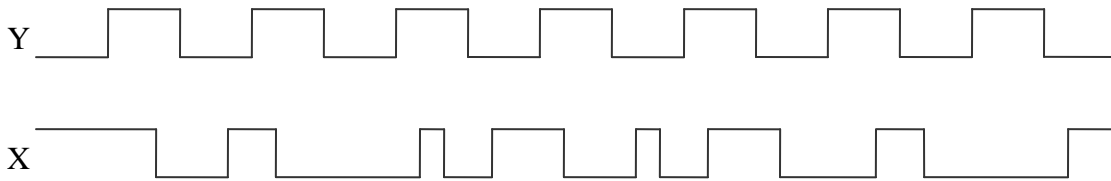
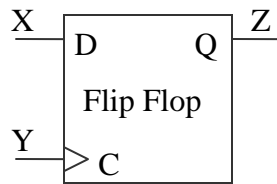


ECE2030A Introduction to Computer Engineering
Fall 2008
Homework Assignment #3
 Assigned 10/29/08 Due in the **first 5 min** in class 11/07
No late turn-in accepted

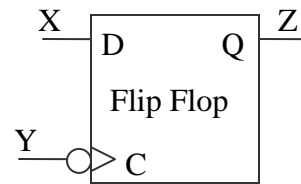
- (10%) Draw a 5-bit by 4-bit unsigned integer multiplier using AND gates and Full Adders, i.e., the multiplicand (top) is 5 bit while the multiplier (bottom) is 4 bit. (You don't need to draw the internal details of the Full Adder.)
- (20%) Given the following wave forms (timing diagram), draw the output Z for each of the five circuits below. Assume the initial values are zero. (Hint for (e): similar case to our synchronous counter using Toggle cell. For transition, consider there is slight delay for the arrival of the input to the second D flip-flop.)



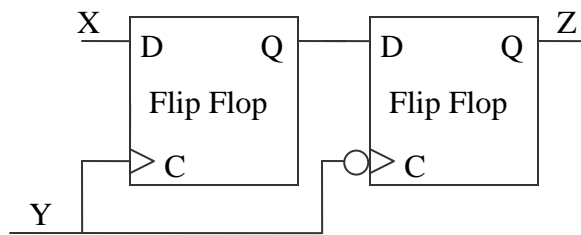
(a)



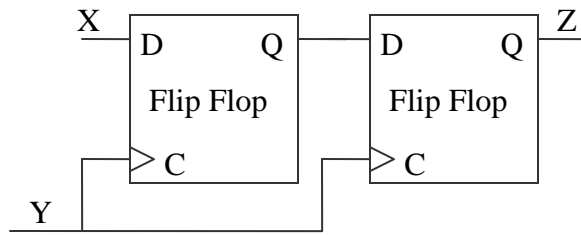
(b)



(c)



(d)



(e)

3. (20%) Using finite-state machine (FSM) method, design a sequence detector which will output 1 when the following input strings are encountered: 001 and 110. In other words, your design has to fulfill the following conditions: (Note that, you must implement this in one single circuit.)

$$\text{Output}(t) = \begin{cases} 1, & \text{if Input}(t-2, t) = 001 \\ 1, & \text{if Input}(t-2, t) = 110 \\ 0, & \text{Otherwise} \end{cases}$$

4. (20%) Using finite-state machine method to design a 4-bit Gray Counter. The Gray Counter counts numbers using Gray Code in which two successive numbers will have their Hamming distance to be one. The counting sequence is shown below. Design your sequential circuits, so that the count advances (and wraps around) upon each positive clock edge. Assume your D flip-flops do not have a clear bit input, so design the “clear” function as part of your implementation so that you can reset the counter back to 0000. (You can use D Flip-flop symbol directly without drawing its internal detail circuits.)

Gray Counter Sequence

0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0

5. (30%) Whack-a-mole. Design a finite-state machine that generates the specified mole's pop-up behavior with an input hammer. Assume that one hammer smash will be given per clock cycle. There are only 4 moles (encoded as 00, 01, 10, and 11) in this console as shown below. We use $M(t)$ to represent the encoded ID of the mole popping up at clock t . Only one mole will pop up at a time. Assume that the initial state $M(0) = 00$, i.e., Mole 00 will pop up at clock 0. The following conditional construct specifies the desired behavior. For example, if Mole 00 pops up, and you also hammer Mole 00 (by giving an input 00), then the next popping-up Mole will be Mole 01 and the HIT signal is asserted. Another hit example, if Mole 11 pops up, and you hammer the right mole ($M(t)=11$), the next Mole to pop up will be Mole 00 (i.e. the increment will wrap.) However, say, if Mole 10 pops up, but you hammer 01 (a miss), then the next Mole to pop up will be $01 \text{ XOR } 11 = \text{Mole } 10$, and the HIT signal will be 0. Note that, there are 2 input signals (from the hammer) and 5 output signals including 4 signals to actuate the Moles and 1 HIT signal. To design this FSM, you need to first construct the state diagram, and then the state table, and finally derive the logic circuits. (You can use D Flip-flop symbol directly without drawing its internal detail circuits.)

```

if (M(t) == Input(t))
    M(t+1) = M(t) + 1;
    HIT = 1;
else
    M(t+1) = Input(t) XOR 11;
    HIT = 0;

```

