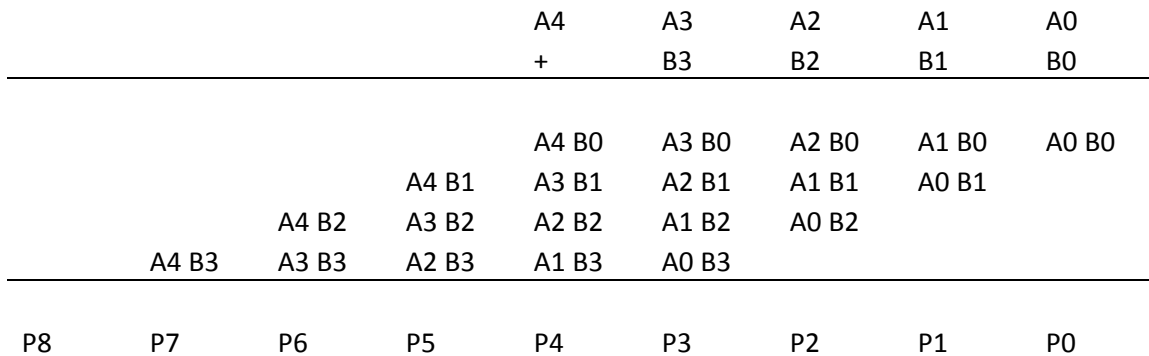


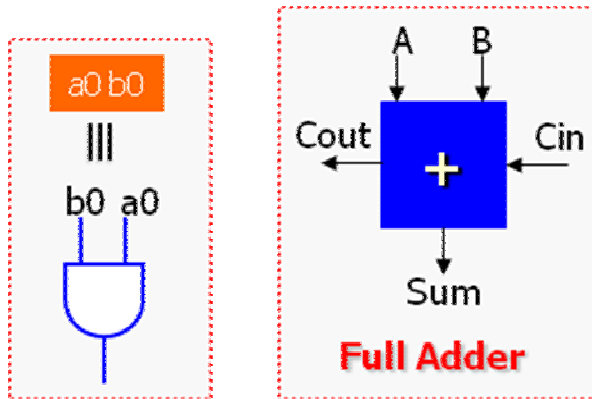
**ECE2030A Introduction to Computer Engineering**  
**Fall 2008**  
**Homework Assignment #3**  
Assigned 10/29/08                      Due in the **first 5 min** in class 11/07  
**SOLUTIONS**                                      No late turn-in accepted

1. (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.)

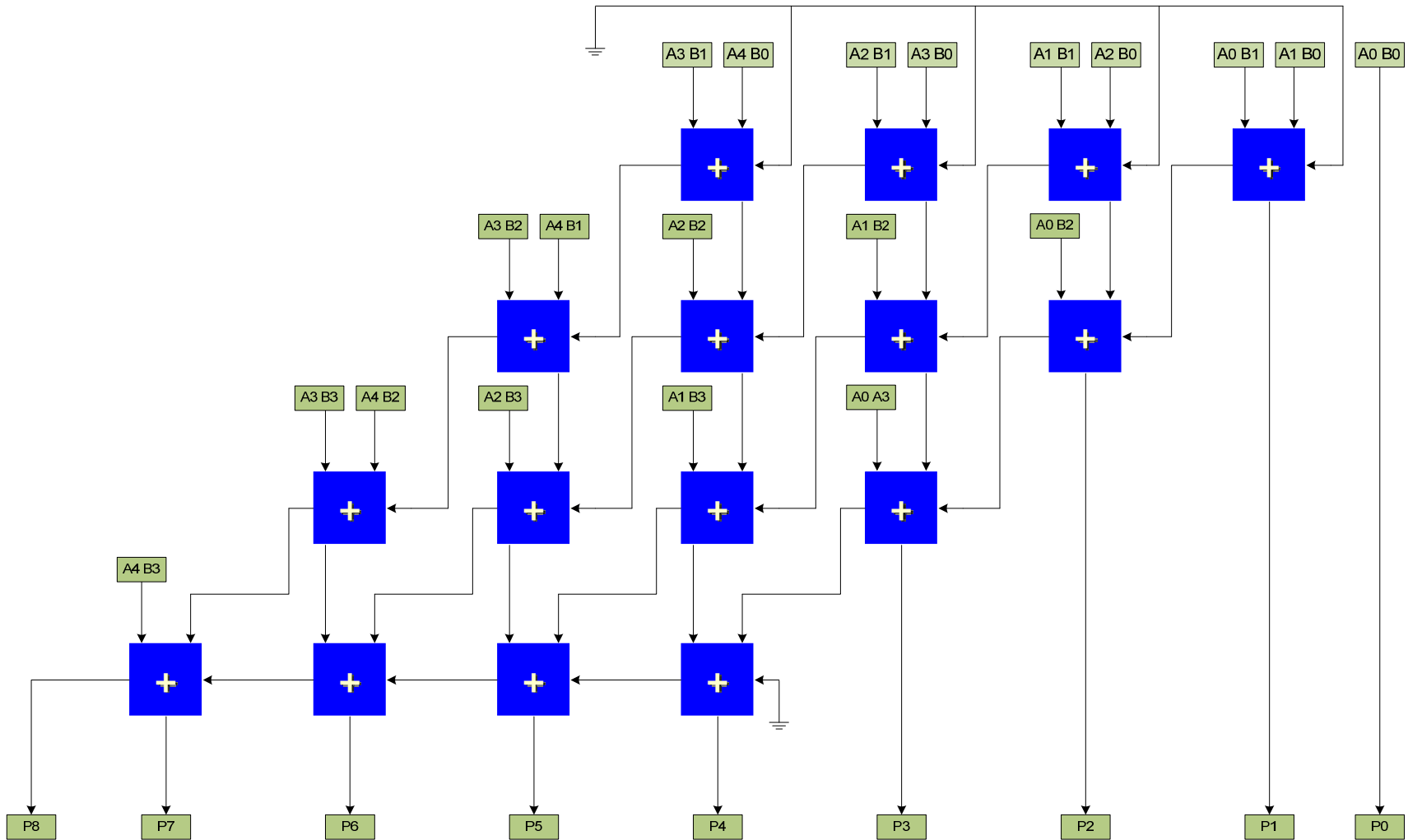
Let the top input be A[4:0], and bottom input be B[3:0].



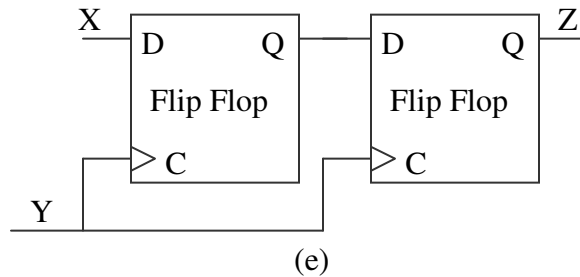
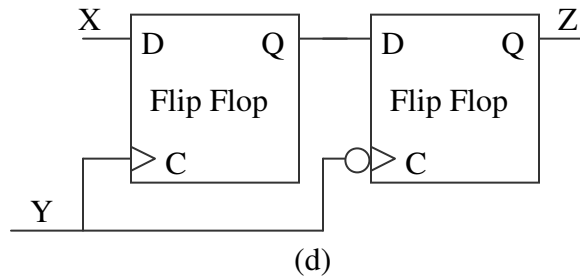
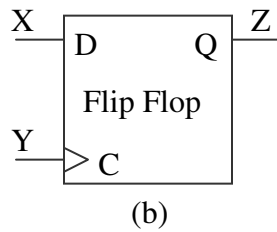
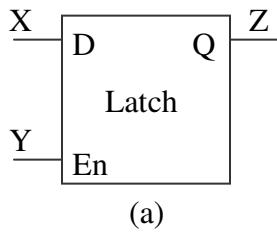
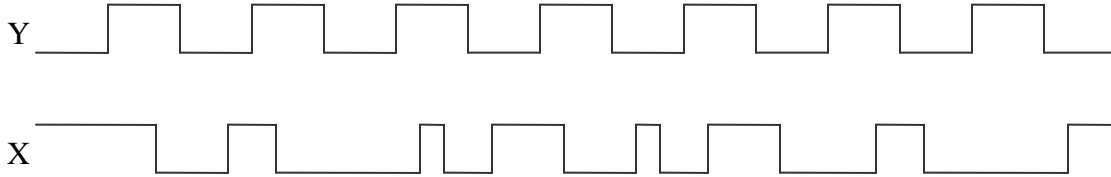
For Simplicity let:

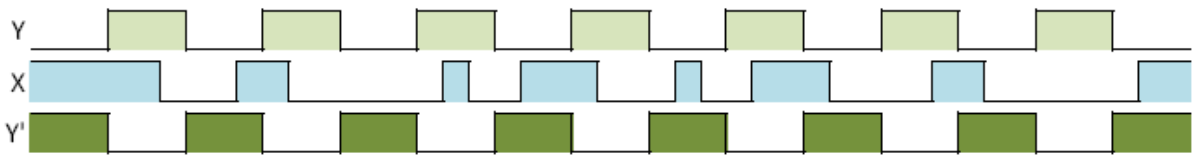


The Carry-Save Multiplier design:

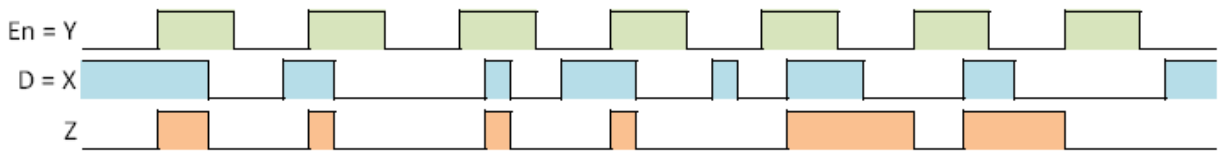


2. (20%) Given the following wave forms (timing diagram), draw the output Z for each of the five circuits below. Assume the initial values are zero. (Hint for (e): similar case to our synchronous counter using Toggle cell. For transition, consider there is slight delay for the arrival of the input to the second D flip-flop.)

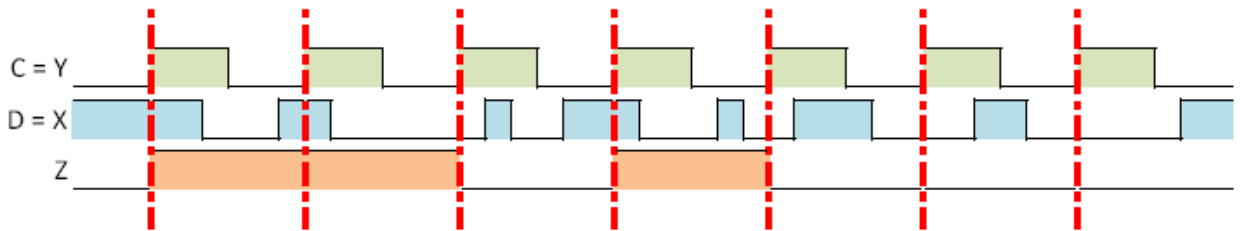




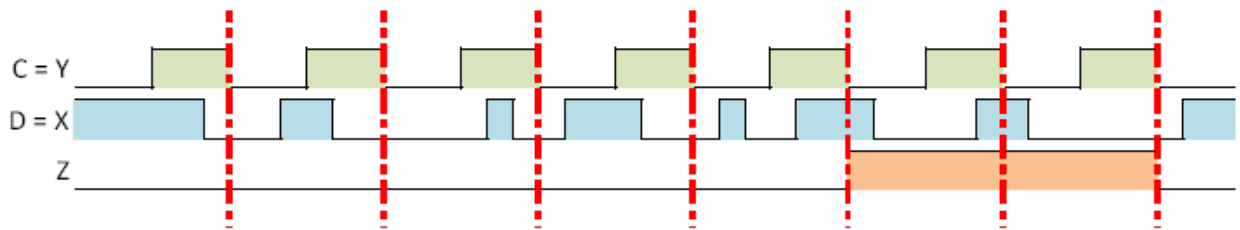
a) Latch



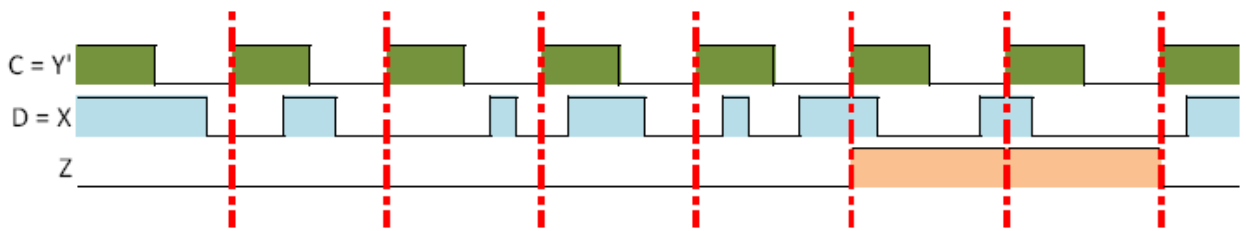
b) Positive Edge Triggered FF



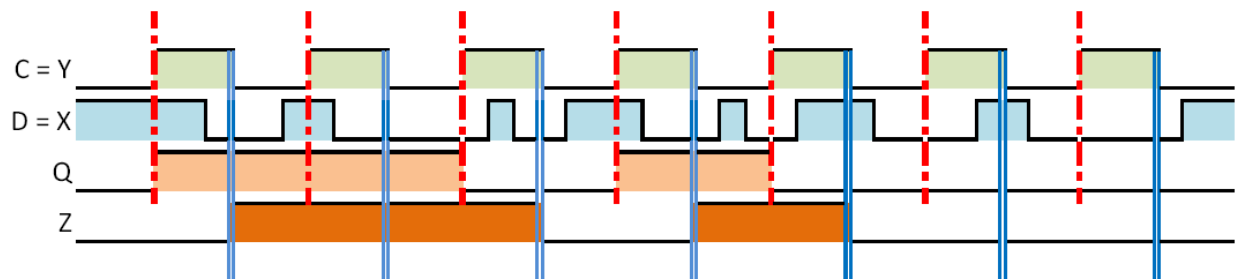
c) Negative Edge Triggered FF



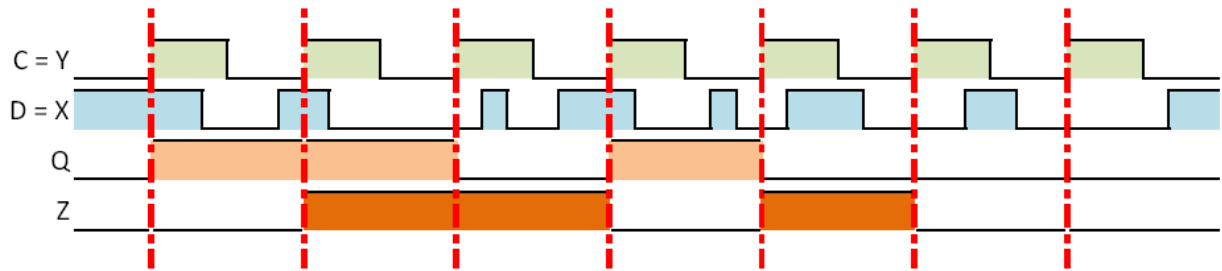
Note this is the same result if you treated it as a PET FF with the Clock input inverted.



d)



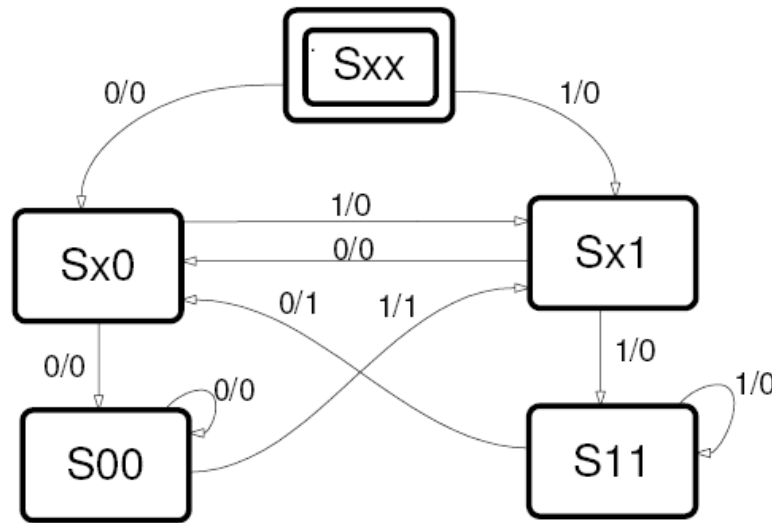
e)



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.)

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

One possible state diagram is:



There are 5 states, so we need 3 bits to encode the states.

Encode the states as:

- Sxx = 100 (Initialization state. Note that in this state no valid inputs have arrived yet.)
- Sx0 = 010 (We have one bit working towards the 001 sequence)
- Sx1 = 001 (We have one bit working towards the 110 sequence)
- S00 = 000 (We have two bits working towards the 001 sequence)
- S11 = 011 (We have two bits working towards the 110 sequence)

The first valid can occur only after 3 samples have arrived.

The state table for the state diagram is:

| Present State |    |    | input | Next State |    |    | Output |
|---------------|----|----|-------|------------|----|----|--------|
| P2            | P1 | P0 | x(t)  | N2         | N1 | N0 |        |
| 0             | 0  | 0  | 0     | 0          | 0  | 0  | 0      |
| 0             | 0  | 0  | 1     | 0          | 0  | 1  | 1      |
| 0             | 0  | 1  | 0     | 0          | 1  | 0  | 0      |
| 0             | 0  | 1  | 1     | 0          | 1  | 1  | 0      |
| 0             | 1  | 0  | 0     | 0          | 0  | 0  | 0      |
| 0             | 1  | 0  | 1     | 0          | 0  | 1  | 0      |
| 0             | 1  | 1  | 0     | 0          | 1  | 0  | 1      |
| 0             | 1  | 1  | 1     | 0          | 1  | 1  | 0      |
| 1             | 0  | 0  | 0     | 0          | 1  | 0  | 0      |
| 1             | 0  | 0  | 1     | 0          | 0  | 1  | 0      |

Use K-maps to minimize the logic for the next state and output.

$N2 = 0.$

$N0 = x(t).$

N1:

| P2 P1 |    | P0 x(t) |    |    |    |
|-------|----|---------|----|----|----|
|       |    | 00      | 01 | 11 | 10 |
| 00    | 00 | 0       | 0  | 1  | 1  |
| 01    | 01 | 0       | 0  | 1  | 1  |
| 11    | 11 | x       | x  | x  | x  |
| 10    | 10 | 1       | 0  | x  | x  |

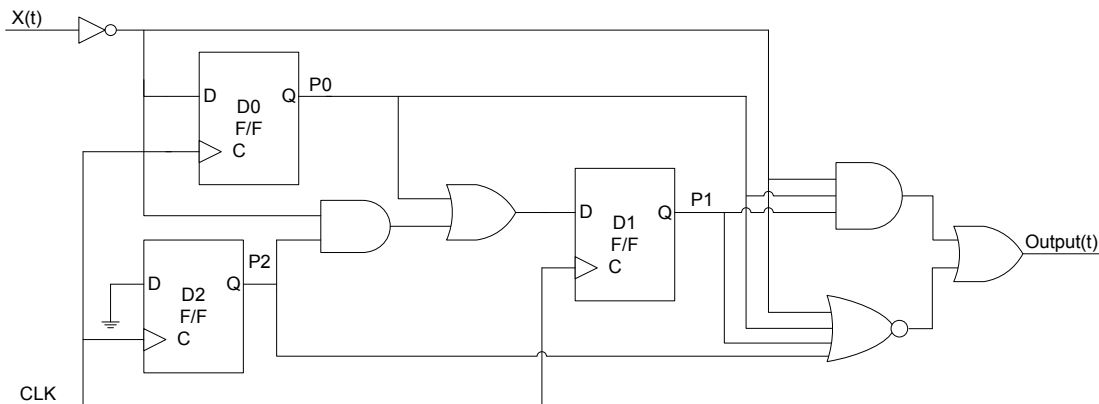
Output:

| P2 P1 |    | P0 x(t) |    |    |    |
|-------|----|---------|----|----|----|
|       |    | 00      | 01 | 11 | 10 |
| 00    | 00 | 0       | 1  | 0  | 0  |
| 01    | 01 | 0       | 0  | 0  | 1  |
| 11    | 11 | x       | x  | x  | x  |
| 10    | 10 | 0       | 0  | x  | x  |

$N1 = P0 + P2 \cdot \overline{x(t)}$

$Output(t) = P1 \cdot P0 \cdot \overline{x(t)} + \overline{P2} \cdot \overline{P1} \cdot \overline{P0} \cdot x(t)$

The circuit:



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 |

The state table:

| P3 | P2 | P1 | P0 | N3 | N2 | N1 | N0 |
|----|----|----|----|----|----|----|----|
| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  |
| 0  | 0  | 0  | 1  | 0  | 0  | 1  | 1  |
| 0  | 0  | 1  | 0  | 0  | 1  | 1  | 0  |
| 0  | 0  | 1  | 1  | 0  | 0  | 1  | 0  |
| 0  | 1  | 0  | 0  | 1  | 1  | 0  | 0  |
| 0  | 1  | 0  | 1  | 0  | 1  | 0  | 0  |
| 0  | 1  | 1  | 0  | 0  | 1  | 1  | 1  |
| 0  | 1  | 1  | 1  | 0  | 1  | 0  | 1  |
| 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 1  | 0  | 0  | 1  | 1  | 0  | 0  | 0  |
| 1  | 0  | 1  | 0  | 1  | 0  | 1  | 1  |
| 1  | 0  | 1  | 1  | 1  | 0  | 0  | 1  |
| 1  | 1  | 0  | 0  | 1  | 1  | 0  | 1  |
| 1  | 1  | 0  | 1  | 1  | 1  | 1  | 1  |
| 1  | 1  | 1  | 0  | 1  | 0  | 1  | 0  |
| 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  |

Use K-Map to minimize the next state logic.

For N3:

| P3 P2 \ P1 P0 |  | P1 P0 |    |    |    |
|---------------|--|-------|----|----|----|
|               |  | 00    | 01 | 11 | 10 |
| 00            |  | 0     | 0  | 0  | 0  |
| 01            |  | 1     | 0  | 0  | 0  |
| 11            |  | 1     | 1  | 1  | 1  |
| 10            |  | 0     | 1  | 1  | 1  |

$$N3 = P3 \cdot P1 + P3 \cdot P0 + P2 \cdot \overline{P1} \cdot \overline{P0}$$

$$= P3 \cdot (P1 + P0) + P2 \cdot \overline{P1} \cdot \overline{P0}$$

For N2:

| P3 P2 \ P1 P0 |  | P1 P0 |    |    |    |
|---------------|--|-------|----|----|----|
|               |  | 00    | 01 | 11 | 10 |
| 00            |  | 0     | 0  | 0  | 1  |
| 01            |  | 1     | 1  | 1  | 1  |
| 11            |  | 1     | 1  | 1  | 0  |
| 10            |  | 0     | 0  | 0  | 0  |

$$N2 = P2 \cdot P1 + P2 \cdot P0 + P3 \cdot P1 \cdot P0$$

$$= P2 \cdot (P1 + P0) + P3 \cdot P1 \cdot P0$$

For N1:

| P3 P2 \ P1 P0 |  | P1 P0 |    |    |    |
|---------------|--|-------|----|----|----|
|               |  | 00    | 01 | 11 | 10 |
| 00            |  | 0     | 1  | 1  | 1  |
| 01            |  | 0     | 0  | 0  | 1  |
| 11            |  | 0     | 1  | 1  | 1  |
| 10            |  | 0     | 0  | 0  | 1  |

$$N1 = \overline{P3} \cdot \overline{P2} \cdot P0 + P3 \cdot \overline{P2} \cdot P0 + P1 \cdot \overline{P0}$$

$$= (\overline{P3} \cdot \overline{P2} + P3 \cdot \overline{P2}) \cdot P0 + P1 \cdot \overline{P0}$$

$$= (\overline{P3} \oplus P3) \cdot \overline{P2} \cdot P0 + P1 \cdot \overline{P0}$$

For N0:

| P3 P2 \ P1 P0 |  | P1 P0 |    |    |    |
|---------------|--|-------|----|----|----|
|               |  | 00    | 01 | 11 | 10 |
| 00            |  | 1     | 1  | 0  | 0  |
| 01            |  | 0     | 0  | 1  | 1  |
| 11            |  | 1     | 1  | 0  | 0  |
| 10            |  | 0     | 0  | 1  | 1  |

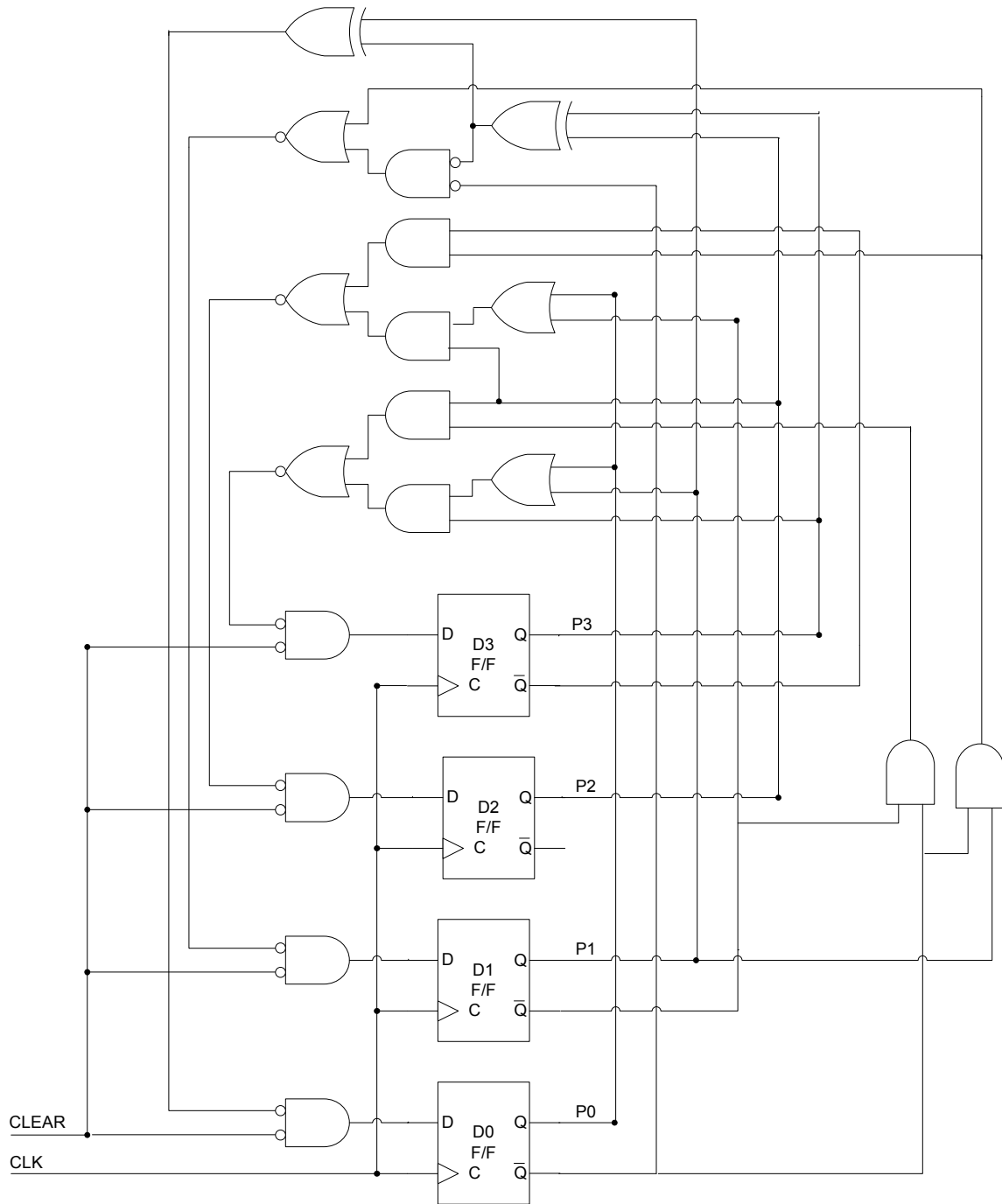
$$N0 = \overline{P3} \cdot \overline{P2} \cdot \overline{P1} + \overline{P3} \cdot \overline{P2} \cdot P1 + P3 \cdot \overline{P2} \cdot \overline{P1} + P3 \cdot \overline{P2} \cdot P1$$

$$= (\overline{P3} \cdot \overline{P2} + P3 \cdot \overline{P2}) \cdot \overline{P1} + (\overline{P3} \cdot \overline{P2} + P3 \cdot \overline{P2}) \cdot P1$$

$$= (\overline{P3} \oplus P3) \cdot \overline{P2} + (P3 \oplus \overline{P3}) \cdot \overline{P2} \cdot P1$$

$$= (\overline{P3} \oplus P3) \oplus P1$$

The circuit:

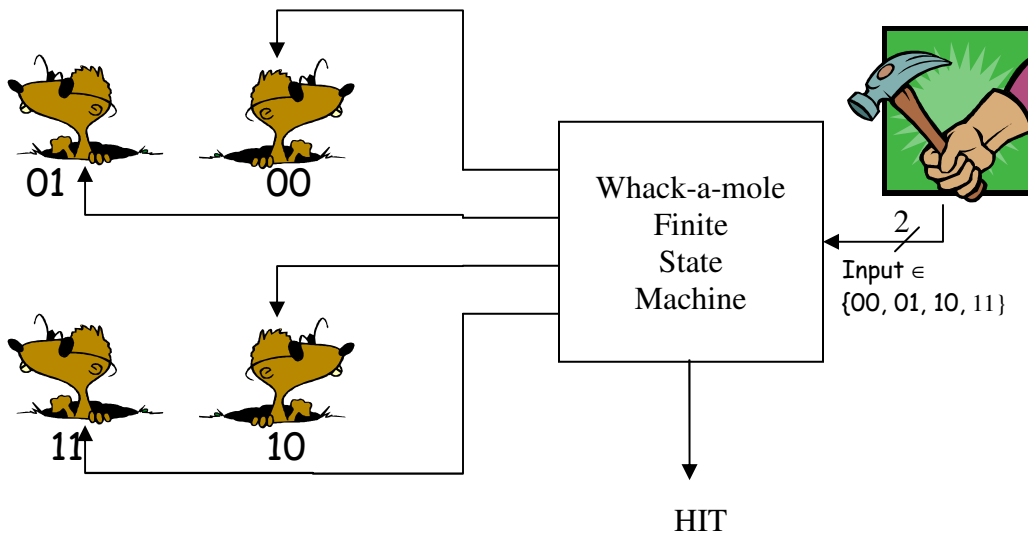


5. (30%) Whack-a-mole. Design a finite-state machine that generates the specified mole's pop-up behavior with an input hammer. Assume that one hammer smash will be given per clock cycle. There are only 4 moles (encoded as 00, 01, 10, and 11) in this console as shown below. We use  $M(t)$  to represent 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.)

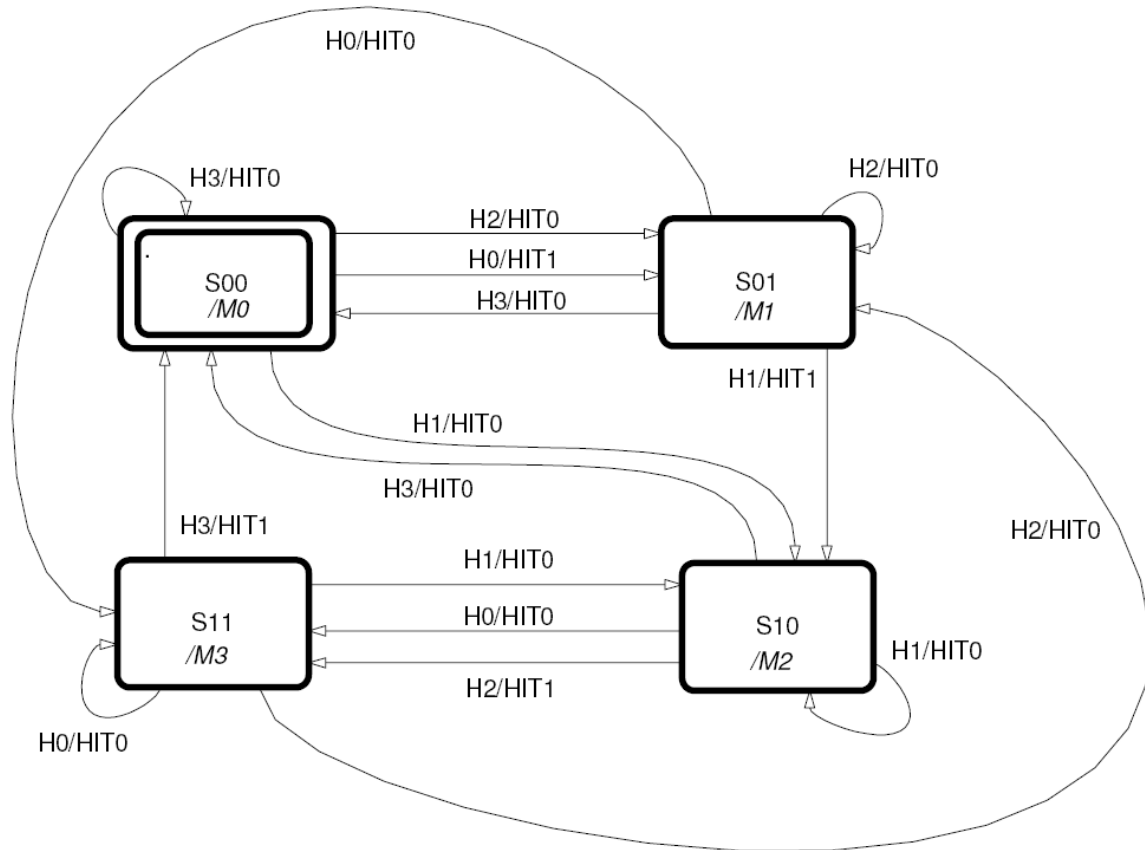
```

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

```



State Diagram:



S00 = State with Mole 00 popped up  
 S01 = State with Mole 01 popped up  
 S10 = State with Mole 10 popped up  
 S11 = State with Mole 11 popped up  
 HIT1 = HIT equals 1  
 HIT0 = HIT equals 0  
 H3 = 11 (tired to hit Mole (11))  
 H2 = 10 (tired to hit Mole (10))  
 H1 = 01 (tired to hit Mole (01))  
 H0 = 00 (tired to hit Mole (00))

Let the each Mole actuator correspond to a bit. The 4 bit signals sent to the Moles are order as M(11), M(10), M(01), M(00).  
 M0 means M(11) = 0, M(10) = 0, M(01) = 0, M(00) = 1 and is encoded as 0001.

M0 = 0001 (makes Mole (00) pop)  
 M1 = 0010 (makes Mole (01) pop)  
 M2 = 0100 (makes Mole (10) pop)  
 M3 = 1000 (makes Mole (11) pop)

State Table

| DECODER | Present |    | INPUT |    | Next |    | OUTPUT |
|---------|---------|----|-------|----|------|----|--------|
| Value   | P1      | P0 | H1    | H0 | N1   | N0 | HIT    |
| D0      | 0       | 0  | 0     | 0  | 0    | 1  | 1      |
| D1      | 0       | 0  | 0     | 1  | 1    | 0  | 0      |
| D2      | 0       | 0  | 1     | 0  | 0    | 1  | 0      |
| D3      | 0       | 0  | 1     | 1  | 0    | 0  | 0      |
| D4      | 0       | 1  | 0     | 0  | 1    | 1  | 0      |
| D5      | 0       | 1  | 0     | 1  | 1    | 0  | 1      |
| D6      | 0       | 1  | 1     | 0  | 0    | 1  | 0      |
| D7      | 0       | 1  | 1     | 1  | 0    | 0  | 0      |
| D8      | 1       | 0  | 0     | 0  | 1    | 1  | 0      |
| D9      | 1       | 0  | 0     | 1  | 1    | 0  | 0      |
| D10     | 1       | 0  | 1     | 0  | 1    | 1  | 1      |
| D11     | 1       | 0  | 1     | 1  | 0    | 0  | 0      |
| D12     | 1       | 1  | 0     | 0  | 1    | 1  | 0      |
| D13     | 1       | 1  | 0     | 1  | 1    | 0  | 0      |
| D14     | 1       | 1  | 1     | 0  | 0    | 1  | 0      |
| D15     | 1       | 1  | 1     | 1  | 0    | 0  | 1      |

| DECODER | Present |    | Mole Actuators |       |       |       |
|---------|---------|----|----------------|-------|-------|-------|
| Value   | P1      | P0 | M(11)          | M(10) | M(01) | M(00) |
| D0      | 0       | 0  | 0              | 0     | 0     | 1     |
| D1      | 0       | 1  | 0              | 0     | 1     | 0     |
| D2      | 1       | 0  | 0              | 1     | 0     | 0     |
| D3      | 1       | 1  | 1              | 0     | 0     | 0     |

Let the Present State and Input bits be inputted into a 4-to-16 decoder.  
The next state and HIT equations depend on the input (Mealy Machine).

The next states equations are:  
 $N1 = D1 + D4 + D5 + D8 + D9 + D10 + D12 + D13$   
 $N0 = H0'$

The HIT output equation is:  
 $HIT = D0 + D5 + D10 + D15$

Let the Present State also be the input into a 2-to-4 decoder.  
The Mole actuators equations depend only on the state (Moore Machine).

The Mole Actuators equations are simply the output of the 2-to-4 decoder.

If you did the K-map method:

For N1

| P1 P0 \ H1 H0 |    | H1 H0 |    |    |    |
|---------------|----|-------|----|----|----|
|               |    | 00    | 01 | 11 | 10 |
| P1 P0         | 00 | 0     | 1  | 0  | 0  |
|               | 01 | 1     | 1  | 0  | 0  |
|               | 11 | 1     | 1  | 0  | 0  |
|               | 10 | 1     | 1  | 0  | 1  |

$$N1 = \overline{H1} \cdot P0 + \overline{H1} \cdot H0 + \overline{H0} \cdot P1 \cdot \overline{P0}$$

$$= \overline{H1} \cdot (P0 + H0) + \overline{H0} \cdot P1 \cdot \overline{P0}$$

For Output

| P1 P0 \ H1 H0 |    | H1 H0 |    |    |    |
|---------------|----|-------|----|----|----|
|               |    | 00    | 01 | 11 | 10 |
| P1 P0         | 00 | 1     | 0  | 0  | 0  |
|               | 01 | 0     | 1  | 0  | 0  |
|               | 11 | 0     | 0  | 1  | 0  |
|               | 10 | 0     | 0  | 0  | 1  |

Output = 1 only if P1P0 = H1H0. You could either implement this with a 2 bit comparator or simply as XNORs:

$$\text{Output} = \overline{(P1 \oplus H1)} \cdot \overline{(P0 \oplus H0)}$$

Remember XNOR is 1 when both inputs are the same.

The Circuit

