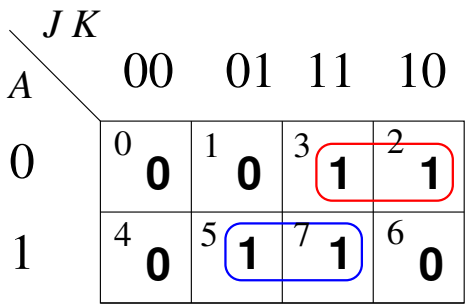
EE 231 – Homework 7
Due October 15, 2010

1. Show how to build a J-K flip-flop using a T flip-flop and some combinational logic.

   A J-K flipflop is a synchronous sequential circuit with two inputs (J and K) and one state flip-flop (A). We design this from a state transition table. We use a table like Table 5.5 (p. 205 of the text).

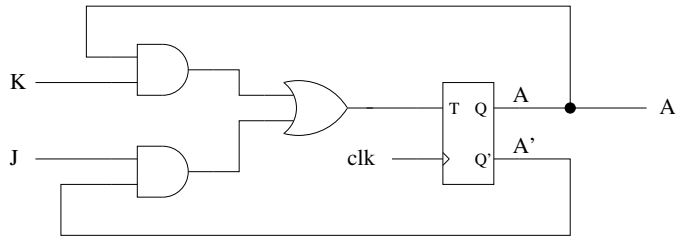
Present State	Input		Next State	Flip-Flop Inputs
A	J	K	A	T_A
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

   To find T_A look at the present state of A and the next state of A. If they are the same, the flip-flop should not toggle (T_A should be 0); if they are different, the flip-flop should toggle (T_A should be 1). We draw a three-input (A,J, K), one-output (T_A) Karnaugh map:

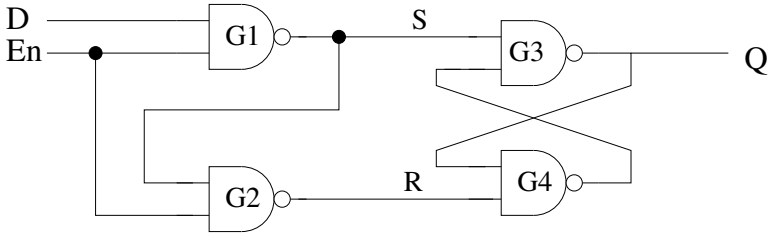


   $T_A = A'J + AK$

   The circuit looks like this:



2. Figure 5.6 of the text shows one way to build a D latch. The figure below shows another way. Show that the below functions identically to the D latch of Figure 5.6.



(a) If En is low, the outputs of $G1$ and $G2$ are high, and the RS latch made up of $G3$ and $G4$ holds the last value.

(b) If En is high and D is high, then the output of $G1$ (S) goes low, forcing the output of $G2$ (R) high. With S low and R high, the output of $G3$ (D) goes high, setting the D latch.

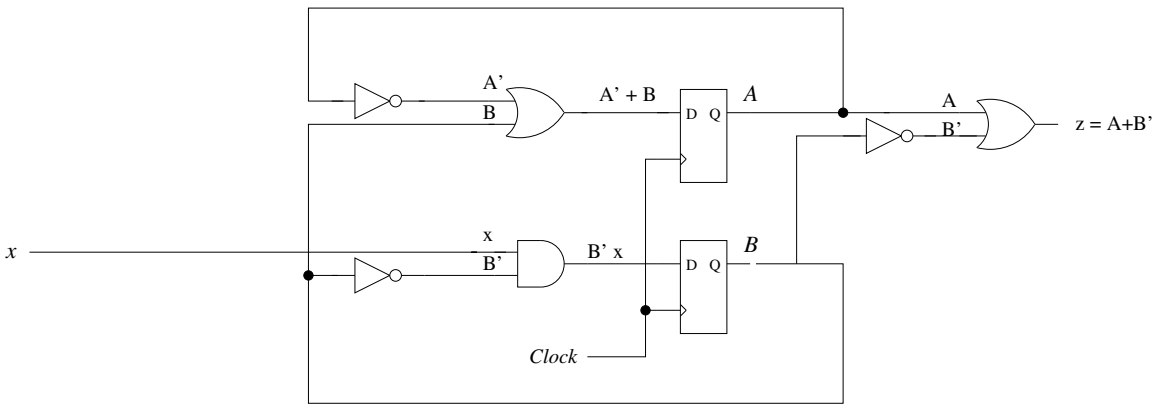
(c) If En is high and D is low, then the output of $G1$ (S) goes high, forcing the output of $G2$ (R) low. With S high and R low, the output of $G3$ (D) goes low, resetting the D latch.

This is the same behavior as the D latch of Figure 5.6. This circuit is easier to build because it requires one fewer gate.

3. A sequential circuit with two D flip-flops A and B, one input x, and one output z is specified by the following next-state and output equations:

$$\begin{aligned} A(t+1) &= A' + B \\ B(t+1) &= B'x \\ z &= A + B' \end{aligned}$$

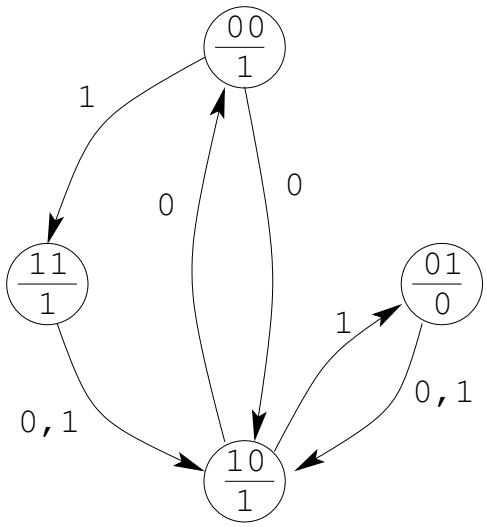
(a) Draw the logic diagram of the circuit.



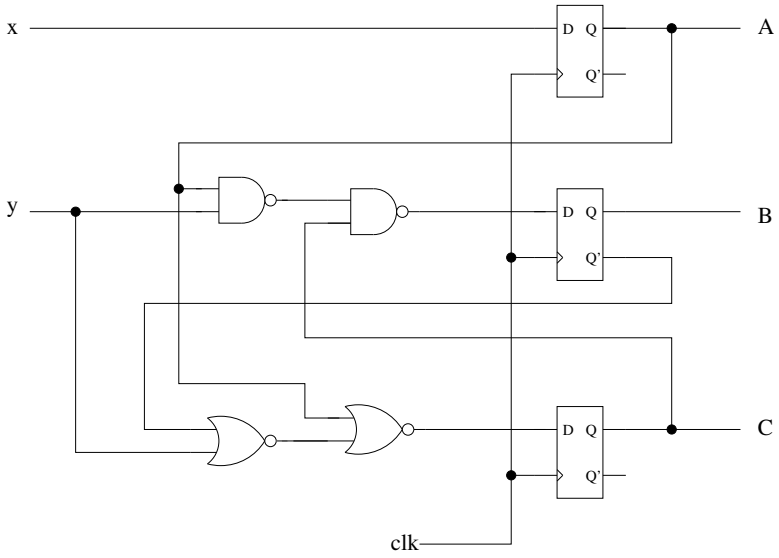
(b) List the state table for the circuit.

Present State		Input	Next State		Output
A	B	x	A	B	z
0	0	0	1	0	1
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	0	1

(c) Draw the corresponding state diagram.



4. A sequential circuit has three flip-flops A, B and C, and two inputs x and y, as shown below.



(a) Derive the state table of the sequential circuit.

The equations are:

$$A(t+1) = x$$

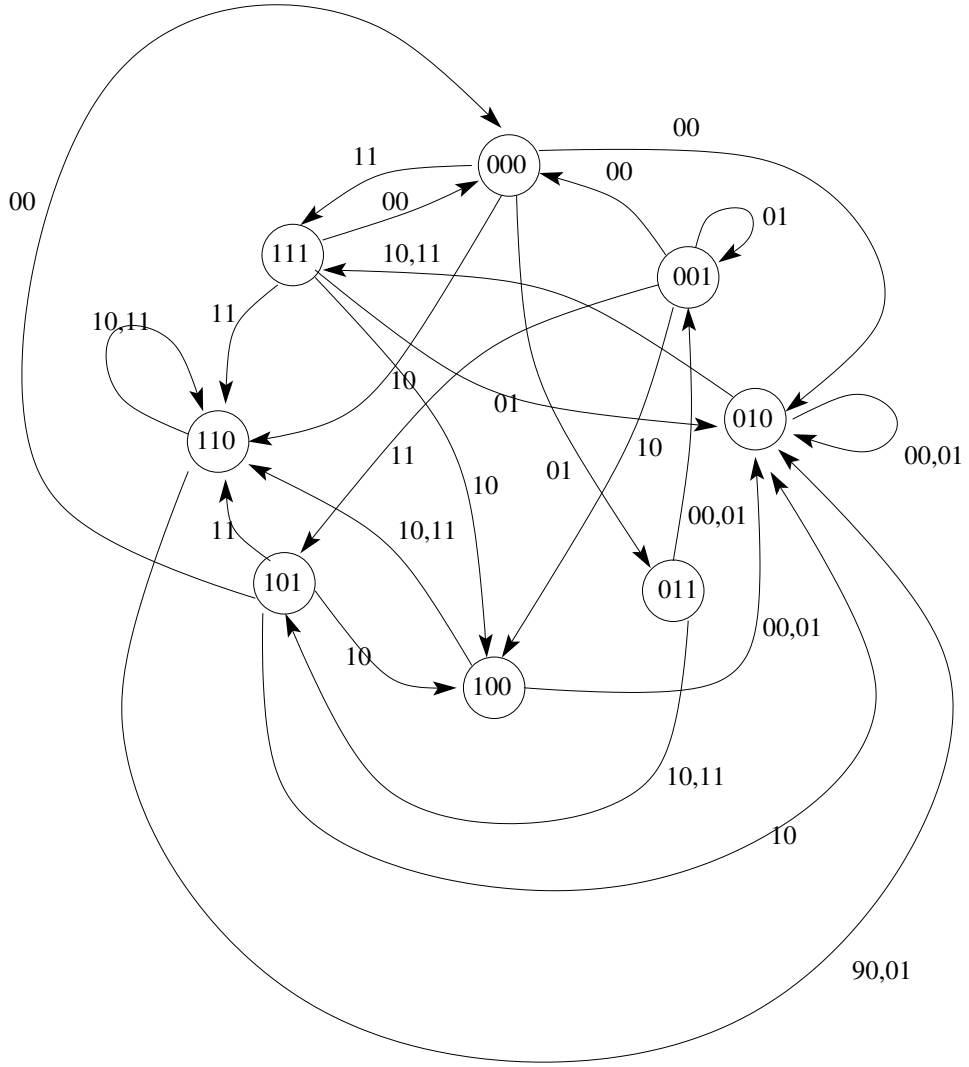
$$B(t+1) = [(Ay)'C]' = (Ay) + C'$$

$$C(t+1) = [(y + B')' + A]' = (y + B')A' = A'y + A'B'$$

Here is the state transition table:

Present State			Inputs		Next State		
A	B	C	x	y	A	B	C
0	0	0	0	0	0	1	1
0	0	0	0	1	0	1	1
0	0	0	1	0	1	1	1
0	0	0	1	1	1	1	1
0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	0	1	0	0
0	0	1	1	1	1	0	1
0	1	0	0	0	0	1	1
0	1	0	0	1	0	1	1
0	1	0	1	0	1	1	1
0	1	0	1	1	1	1	1
0	1	1	0	0	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	0	1	0	0
0	1	1	1	1	1	0	1
1	0	0	0	0	0	1	0
1	0	0	0	1	0	1	0
1	0	0	1	0	1	1	0
1	0	0	1	1	1	1	0
1	0	1	0	0	0	0	0
1	0	1	0	1	0	1	0
1	0	1	1	0	1	0	0
1	0	1	1	1	1	1	0
1	1	0	0	0	0	1	0
1	1	0	0	1	0	1	0
1	1	0	1	0	1	1	0
1	1	0	1	1	1	1	0
1	1	1	0	0	0	0	0
1	1	1	0	1	0	1	0
1	1	1	1	0	1	0	0
1	1	1	1	1	1	1	0

(b) Derive the state diagram of the sequential circuit.



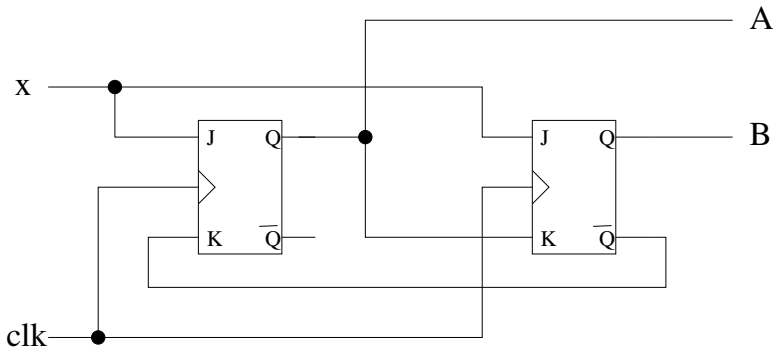
(c) Write a Verilog module to implement the circuit.

```
module hw7_p4(input clk, x, y, ouput reg A, B, C);

always @(posedge clk) begin
    A <= x;
    B <= (A&y) + ~C;
    C <= (~A & y) + (~A & B);
end

endmodule
```

5. Derive the state table and state diagram of the sequential circuit shown below. Draw a timing diagram for clk, x, A and B for 10 clock ticks, assuming that the machine starts in state 00 and x is always 1. Explain the function that the circuit performs.

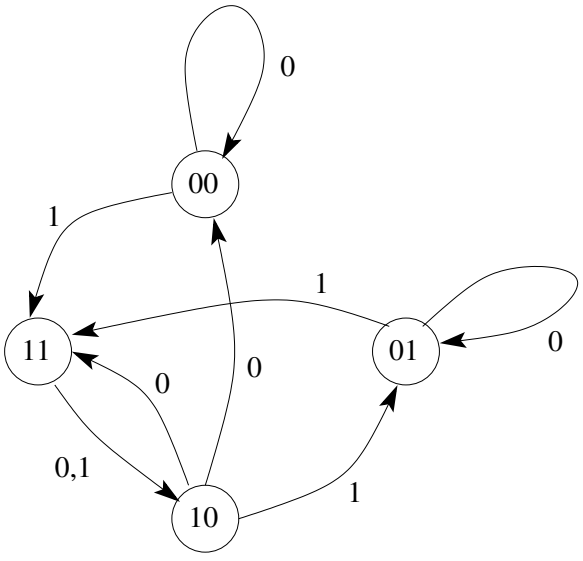


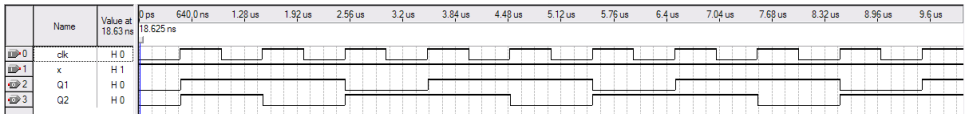
The equations are:

$J_A = x, K_A = B', J_B = x, K_B = A$

State table:

A	B	x	J_A	K_A	J_B	K_B	A	B
0	0	0	0	1	0	0	0	0
0	0	1	1	1	1	0	1	1
0	1	0	0	0	0	0	0	1
0	1	1	1	0	1	0	1	1
1	0	0	0	1	0	1	0	0
1	0	1	1	1	1	1	0	1
1	1	0	0	0	0	1	1	0
1	1	1	1	0	1	1	1	0





If the input x is 1, the system counts 3, 2, 1, 3, 2, 1, ...

If the input x is 0 and the state is 00, it stays in 00.

If the input x is 0 and the state is 01, it stays in 01.

If the input x is 0 and the state is 10, it goes to 00 and stays there.

If the input x is 0 and the state is 11, it goes to 10, then 00, and stays there.