Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

COURSE: CEG2136/CEG2536
Computer Architecture I

SEMESTER: Fall 2016

Assignment 1

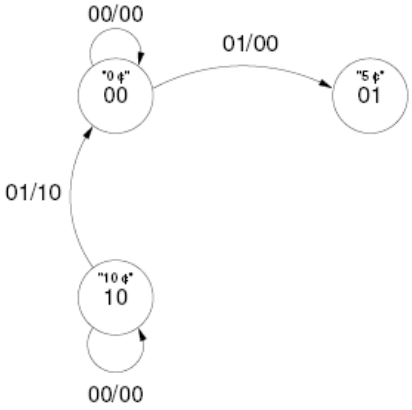
Q1. Design of a vending machine

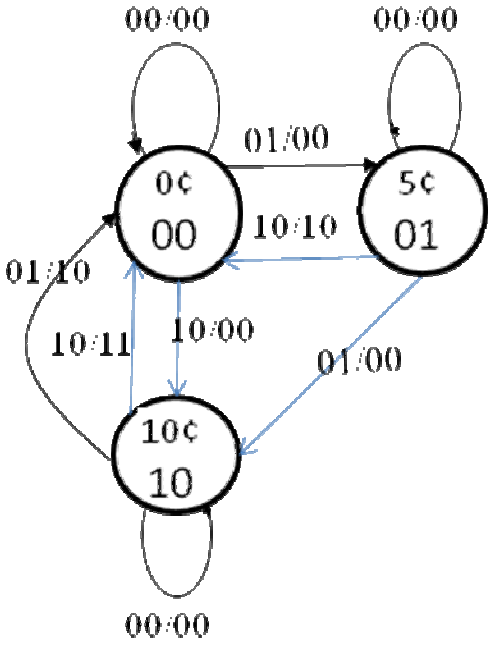
Design and draw the logical diagram of a vending machine for pop drinks. Each can cost 15 cents. Only coins of 5 cents and of 10 cents are accepted. The circuit has two inputs X_1X_0, and two outputs Y_1Y_0. The behaviour of the circuit is described below:

X_1X_0	Description
0 0	No money is deposited in the machine
0 1	1 nickel is deposited in the machine
1 0	1 dime is deposited in the machine

$Y1 = 1$ => Dispense a can
$Y0 = 1$ => Give change a nickel

1. Assuming that the machine starts from state 00, complete the state diagram of the sequential circuit given in the following figure:





2. Derive the state table and then the excitation table, given that JK flip-flops are used for the state register of this sequential circuit.

Present State S^n		In		S^{n+1}		Out		Q_1 in		Q_0 in	
Q_1	Q_0	X_1	X_0	Q_1^+	Q_0^+	Y_1	Y_0	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	0	0	x	0	x
0	0	0	1	0	1	0	0	0	x	1	x
0	0	1	0	1	0	0	0	1	x	0	x
0	0	1	1	x	x	x	x	x	x	x	x
0	1	0	0	0	1	0	0	0	x	x	0
0	1	0	1	1	0	0	0	1	x	x	1
0	1	1	0	0	0	1	0	0	x	x	1
0	1	1	1	x	x	x	x	x	x	x	x
1	0	0	0	1	0	0	0	x	0	0	x
1	0	0	1	0	0	1	0	x	1	0	x
1	0	1	0	0	0	1	1	x	1	0	x
1	0	1	1	x	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x

3. Derive the simplified excitation equations of the JK flip-flops and the output equations.

One can find J_1, K_1, J_0, K_0,

a) **directly from the above excitation table, or**

b) do the transition table

Next state S^{n+1} / Output = $Q_1^+ Q_0^+$ / $Y_1 Y_0$

Q_1Q_0 \ X_1X_0	0 0	0 1	1 1	1 0
00	0 0 / 0 0	0 1 / 0 0	x x / x x	1 0 / 0 0
01	0 1 / 0 0	1 0 / 0 0	x x / x x	0 0 / 1 0
11	x x / x x	x x / x x	x x / x x	x x / x x
10	1 0 / 0 0	0 0 / 1 0	x x / x x	0 0 / 1 1

and then find the transition and output functions as shown next.

Find first the transition function for Q^+_1 = $\delta(Q_1,Q_0,X_1,X_0)$ to obtain J & K by mapping it to the JKFF characteristic function $Q^{n+1} = Q^n \cdot J^n + Q^n \cdot K^n$

Q_1^+

Q_1Q_0 \ X_1X_0	00	01	11	10
00	0	0	x	1
01	0	1	x	0
11	x	x	x	x
10	1	0	x	0

$Q_1^+ = Q_0 X_0 + Q'_1 Q'_0 X_1 = Q_0(Q_1 + Q'_1) X_0 + Q'_1 Q'_0 X_1 =$

$= Q_0 (Q_1) X_0 + Q_0 (Q'_1) X_0 + Q'_1 Q'_0 X_1 =$

$Q_1^+ = Q'_1 (Q_0 X_0 + Q'_0 X_1) + Q_1 X_0 Q_0 =$

$Q_1^+ = Q'_1$ **J_1** $+ Q_1$ **K_1** (JKFF char. equation)

➔ **J_1** = $Q_0 X_0 + Q'_0 X_1$; **K_1** = $X_0 Q_0$

Q_0^+

Q_1Q_0 \ X_1X_0	00	01	11	10
00	0	1	x	0
01	1	0	x	0
11	x	x	x	x
10	0	0	x	0

$Q_0^+ = Q'_1 Q'_0 X_0$

➔ **J_0** = ... **K_0** = ...

Y_1

Q_1Q_0 \ X_1X_0	00	01	11	10
00	0	0	x	0
01	0	0	x	1
11	x	x	x	x
10	0	1	x	1

$Y_1 = Q_0 X_0 + Q_0 X_1 + Q_1 X_0 + Q_1 X_1$

Y_0

Q_1Q_0 \ X_1X_0	00	01	11	10
00	0	0	x	0
01	0	0	x	0
11	x	x	x	x
10	0	0	x	1

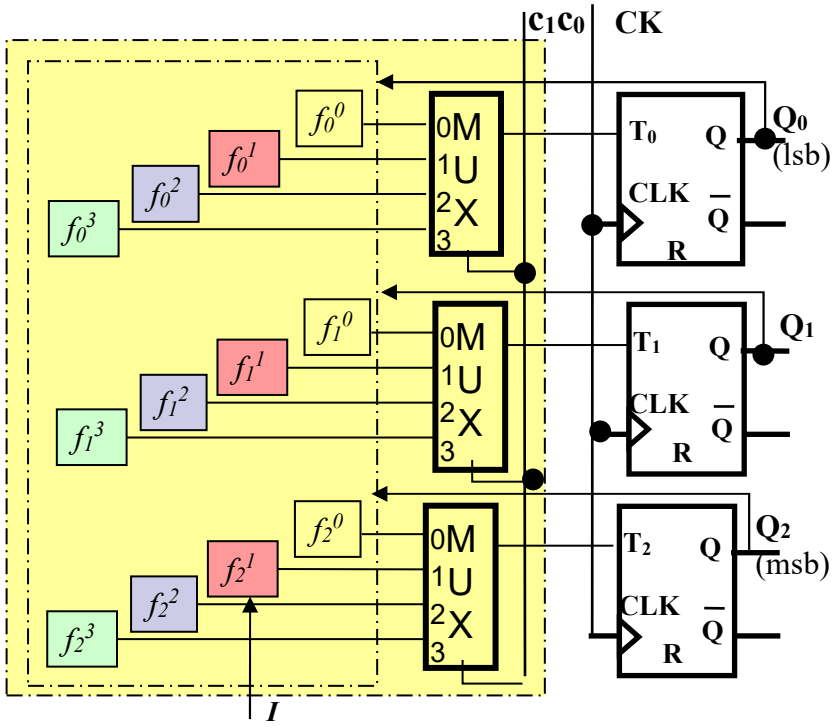
$Y_0 = Q_1 X_1$

4. Draw the logic diagram of the circuit, using only NAND gates and JK flip flops…

Q2. Conceive a 3-bit multifunction register that can perform the following operations:
Use T-type flip flops and any logic gates or other combinational digital circuits.

Top-down design:

c1	c0	Function
0	0	f0: Store (preserve) register's content
0	1	f1: Right shift (register's *serial input* is connected to an external input *I*)
1	0	f2: Increment by 1
1	1	f3: Swap the most significant bit (msb) with the least significant bit (lsb)



c_1	c_0	Function	Transition Equations (preliminary steps)	Excitation Equations T_i (final results - derived for each case separately, as shown after this table)
0	0	f^0: Store (preserve) register's content	$Q_i(n+1) = Q_i(n)$; $i = \{0,1,2\}$	$T_i = 0$; $i = \{0,1,2\}$
0	1	f^1: Right shift (register's *serial input* is connected to an external input *I*)	$Q_i(n+1) = Q_{i+1}(n)$; $i=0,1$ $Q_2(n+1) = I$ (n)	if $Q_i(n) = Q_{i+1}(n) \Rightarrow T_i = 0$; if $Q_i(n) \neq Q_{i+1}(n) \Rightarrow T_i = 1$; *i.e.,* $T_i = Q_i \oplus Q_{i+1}$, $i = 0,1$ $T_2 = Q_2 \oplus I$
1	0	f^2: Increment by 1	$[Q_2\ Q_1\ Q_0](n+1) = [Q_2\ Q_1\ Q_0](n) + 1$	$T_2 = Q_1 \cdot Q_0$ □□ $T_1 = Q_0$ □ $T_0 = 1$
1	1	f^3: Swap the most significant bit (msb) with the least significant bit (lsb)	$Q_0(n+1) = Q_2(n)$ $Q_1(n+1) = Q_1(n)$ $Q_2(n+1) = Q_0(n)$	$T_2 = T_0 = \overline{Q_2} \cdot Q_0 + Q_2 \cdot \overline{Q_0} = Q_2 \oplus Q_0$ $T_1 = 0$

$c_1c_0 = 01$: Transition equation to implement: $Q_i\,(n+1) = Q_{i+1}\,(n)$; i=1,2

Input of Q_i	*Present state of Q_i*	*Next state of Q_i*	
$Q_{i+1}(n)$	$Q_i(n)$	$Q_i(n+1)$	T_i
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

$$T_i = Q_i \oplus Q_{i+1};\ i = 1,2$$
$$T_2 = Q_2 \oplus I$$

$c_1c_0 = 10$: $[Q_2\ Q_1\ Q_0](n+1) = [Q_2\ Q_1\ Q_0](n) + 1$

Present State			Next State					
$Q_2\,(n)$	$Q_1\,(n)$	$Q_0\,(n)$	$Q_2(n+1)$	$Q_1(n+1)$	$Q_0(n+1)$	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

$$T_2 = Q_1 \cdot Q_0$$
$$T_1 = Q_0$$
$$T_0 = 1$$

$c_1c_0 = 11$:

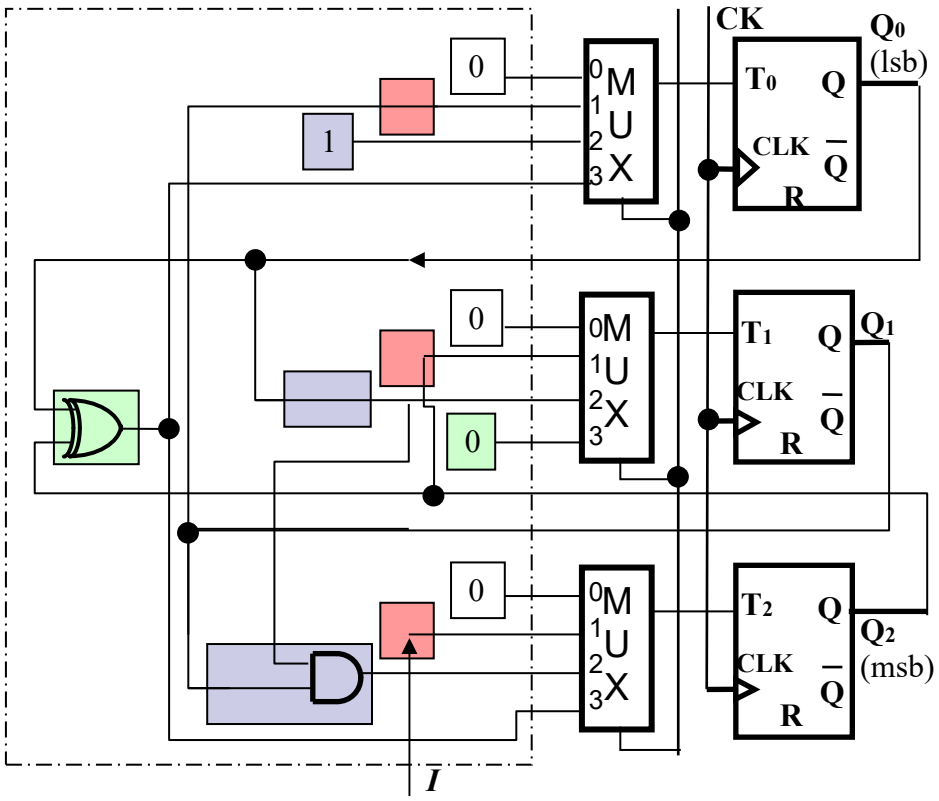
$$Q_0\,(n+1) = Q_2\,(n)$$
$$Q_1\,(n+1) = Q_1\,(n)$$
$$Q_2\,(n+1) = Q_0\,(n)$$

Present State			Next State					
$Q_2\,(n)$	$Q_1\,(n)$	$Q_0\,(n)$	$Q_2(n+1)$	$Q_1(n+1)$	$Q_0(n+1)$	T_2	T_1	T_0
0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	1
0	1	0	0	1	0	0	0	0
0	1	1	1	1	0	1	0	1
1	0	0	0	0	1	1	0	1
1	0	1	1	0	1	0	0	0
1	1	0	0	1	1	1	0	1
1	1	1	1	1	1	0	0	0

$T_2 = T_0$

Q_2 \ $Q_1\,Q_0$	00	01	11	10
0	0	1	1	0
1	1	0	0	1

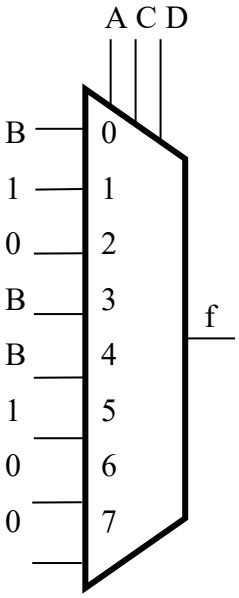
$$T_2 = T_0 =$$
$$= \overline{Q_2} \cdot Q_0 + Q_2 \cdot \overline{Q_0}$$
$$= Q_2 \oplus Q_0$$
$$T_1 = 0$$



Q3. Implement ***f*** *(A,B,C,D)*= Σ(1,4,5,7,9,12,13) using a MUX(8x1) where *A*, *C*, and *D* are connected to the multiplexor's select inputs.

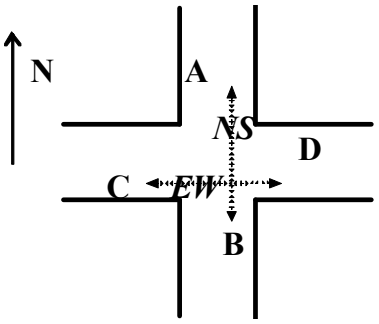
	A	B	C	D	f
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

	A	C	D	B	**f**	
0	0	0	0	0	**0**	B
	0	0	0	1	**1**	
1	0	0	1	0	**1**	1
	0	0	1	1	**1**	
2	0	1	0	0	**0**	0
	0	1	0	1	**0**	
3	0	1	1	0	**0**	B
	0	1	1	1	**1**	
4	1	0	0	0	**0**	B
	1	0	0	1	**1**	
5	1	0	1	0	**1**	1
	1	0	1	1	**1**	
6	1	1	0	0	**0**	0
	1	1	0	1	**0**	
7	1	1	1	0	**0**	0
	1	1	1	1	**0**	



Q4. The figure below represents an intersection of two streets. Sensors have been placed in the proximity of the intersection, along A, B, C and D lanes to detect vehicles' presence. A sensor's output is **1** when there is no vehicle on its monitored lane (indicated as *free* in the following) and **0** when a vehicle is present (*busy* lane). The two sets of intersection lights (East-West or *EW*, and North-South or *NS*) observe the following logic:

1) The *EW* lights will turn green when vehicles will be simultaneously detected on both C and D lanes.
2) The *EW* lights will also be green when either C is *busy* or D is *busy*, and, if A and B are not both *busy* at the same time.
3) The *NS* lights will be green when vehicles will be sensed on both A and B lanes, and, if C and D are not simultaneously *busy*.
4) The *NS* lights will also turn green when either A is *busy* or B is *busy*, but C and D are both *free*.
5) The *EW* lights will also be green when no vehicle is present in the intersection.



a) Using sensors A, B, C, D as input variables, implement a logical combinational circuit that controls the lights. There will be 2 output functions *NS(A,B,C,D)* and *EW(A,B,C,D)* whose values will be 1 when the light is green (and **0** for the red!). Simplify the 2 functions and derive their Sum-of-Products minimal form (no logic diagram asked).

NOTE: The Oxford English Dictionary explains "either … or" as follows:
*The primary function of **either**, etc., is to emphasize the indifference of the two (or more) things or courses … but a secondary function is to emphasize the **mutual exclusiveness**, = either of the two, but not both.*

1)	The *EW* lights will turn green when vehicles will be simultaneously detected on both C and D lanes.	$EW_1 = C'D'$
2)	The *EW* lights will also be green when either C is *busy* or D is *busy*, and, if A and B are not both *busy* at the same time.	**$EW_2 = (C'$ xor $D')\ (A'B')'$**
3)	The *NS* lights will be green when vehicles will be sensed on both A and B lanes, and, if C and D are not simultaneously *busy*.	**$NS_1 = A'B' . (C'D')'$**
4)	The *NS* lights will also turn green when either A is *busy* or B is *busy*, but C and D are both *free*.	**$NS_2 = (A'$ xor $B') . (CD)$**
5)	The *EW* lights will also be green when no vehicle is present in the intersection.	**$EW_3 = ABCD$**

EW

AB \ CD	00	01	11	10
00	**1)**	~~2)~~		~~2)~~
01	**1)**	**2)**		**2)**
11	**1)**	**2)**	**5)**	**2)**
10	**1)**	**2)**		**2)**

NS

AB \ CD	00	01	11	10
00	~~3)~~	**3)**	**3)**	**3)**
01	~~4)~~	~~4)~~	**4)**	~~4)~~
11				
10	~~4)~~	~~4)~~	**4)**	~~4)~~

EW

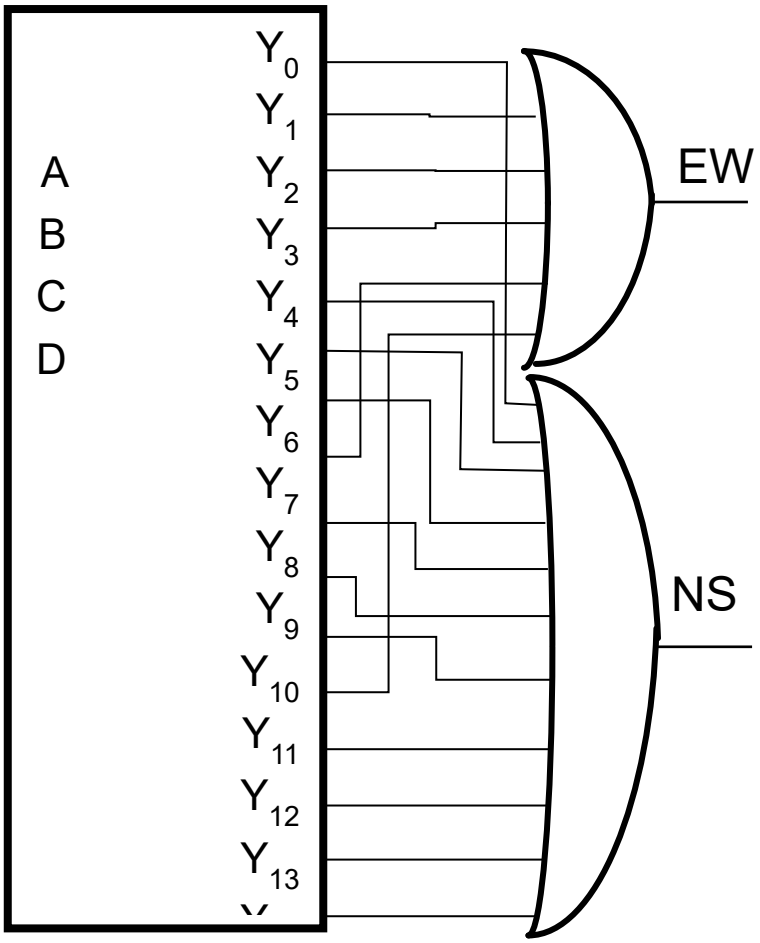
AB \ CD	00	01	11	10
00	**1**			
01	**1**	**1**		**1**
11	**1**	**1**	**1**	**1**
10	**1**	**1**		**1**

NS

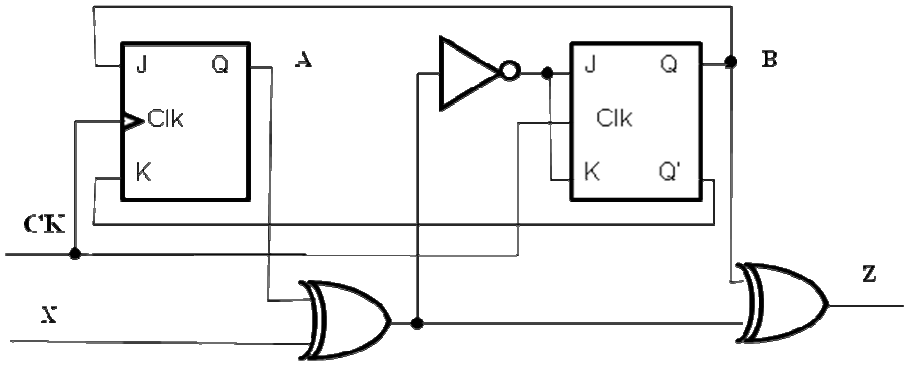
AB \ CD	00	01	11	10
00		**1**	**1**	**1**
01			**1**	
11				
10			**1**	

Condition:					**1)**			**2)**	**3)**	**4)**	**5)**		
	A	B	C	D	**EW_1**	C' xor D'	(A'B')'	**EW_2**	**NS_1**	**NS_2**	**EW_3**	**EW**	**NS**
0	0	0	0	0	**1**							**1**	
1	0	0	0	1		1			**1**				**1**
2	0	0	1	0		1			**1**				**1**
3	0	0	1	1					**1**				**1**
4	0	1	**0**	**0**	**1**		1					**1**	
5	0	1	0	1		1	1	**1**				**1**	
6	0	1	1	0		1	1	**1**				**1**	
7	0	1	1	1			1			**1**			**1**
8	1	0	**0**	**0**	**1**		1					**1**	
9	1	0	0	1		1	1	**1**				**1**	
10	1	0	1	0		1	1	**1**				**1**	
11	1	0	1	1			1			**1**			**1**
12	1	1	0	0	**1**		1					**1**	
13	1	1	0	1		1	1	**1**				**1**	
14	1	1	1	0		1	1	**1**				**1**	
15	1	1	1	1			1				1	1	

b) Realize NS(A,B,C,D) and EW(A,B,C,D) using a 4-to-16 decoder and 2 OR gates; draw the circuit diagram.



Q5. A sequential circuit has two *JK* flip-flops *A* and *B*, one input *X* and one output *Z*. The circuit **logic diagram is shown below**. Derive the circuit **State Table** as well as the **State diagram**.



$JA = B,\ KA = B',$
$JB = KB = (A \oplus X)' = AX + A'X',$
$Y = (A \oplus X) \oplus B$

A_{t+1} and B_{t+1} can be derived using the JK characteristic equation ($Q^{t+1} = \overline{Q^t} \cdot J^t + Q^t \cdot \overline{K^t}$).
Ex., $A_{t+1} = A_tK_A' + A_t'J_A = A_t(B')' + A_t'(B) = A_t(B) + A_t'(B) = (A_t + A_t')B = B$

You can also use the characteristic table instead of the characteristic equation:

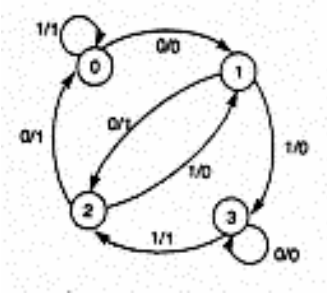
J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q'_t

A	B	X	JA	KA	**A_{t+1}**	JB		KB	**B_{t+1}**
0	0	0	0	1	**0**	1		1	**1**
0	0	1	0	1	**0**	0		0	**0**
0	1	0	1	0	**1**	1		1	**0**
0	1	1	1	0	**1**	0		0	**1**
1	0	0	0	1	**0**	0		0	**0**
1	0	1	0	1	**0**	1		1	**1**
1	1	0	1	0	**1**	0		0	**1**
1	1	1	1	0	**1**	1		1	**0**

State Table:

Present State		In	**Next State**		**Out**
A	B	X	**A_{t+1}**	**B_{t+1}**	**Y**
0	0	0	**0**	**1**	**0**
0	0	1	**0**	**0**	**1**
0	1	0	**1**	**0**	**1**
0	1	1	**1**	**1**	**0**
1	0	0	**0**	**0**	**1**
1	0	1	**0**	**1**	**0**
1	1	0	**1**	**1**	**0**
1	1	1	**1**	**0**	**1**

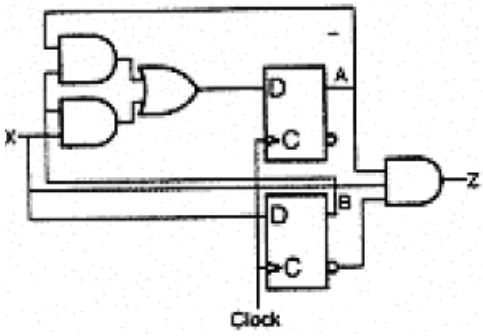
State Diagram



Q6. Using *D flip-flops*, design and implement (devise the logic diagram) the sequential circuit specified by the following state table:

Present State		In	**Next State**		**Out**
A	B	X	**A_{t+1}**	**B_{t+1}**	**Z**
0	0	0	**0**	**0**	**0**
0	0	1	**0**	**1**	**0**
0	1	0	**0**	**0**	**0**
0	1	1	**1**	**1**	**0**
1	0	0	**0**	**0**	**0**
1	0	1	**0**	**1**	**1**
1	1	0	**1**	**0**	**0**
1	1	1	**1**	**1**	**0**

$A^{n+1} = AB + BX$ $B^{n+1} = X$



Q7. Using *JK flip-flops*, design and implement (devise the logic diagram) the sequential circuit specified by the following state table:

Present State		Input	Next State		FF's Inputs (Excitations)			
A	B	X	A	B	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

JA

A \ BX	00	01	11	10
0	0	0	0	1
1	X	X	X	X

JA = BX'

KA

A \ BX	00	01	11	10
0	X	X	X	X
1	0	0	1	0

KA = BX

JB

A \ BX	00	01	11	10
0	0	1	X	X
1	0	1	X	X

JB = X

KB

A \ BX	00	01	11	10
0	X	X	0	1
1	X	X	1	0

KB = A'B + AX