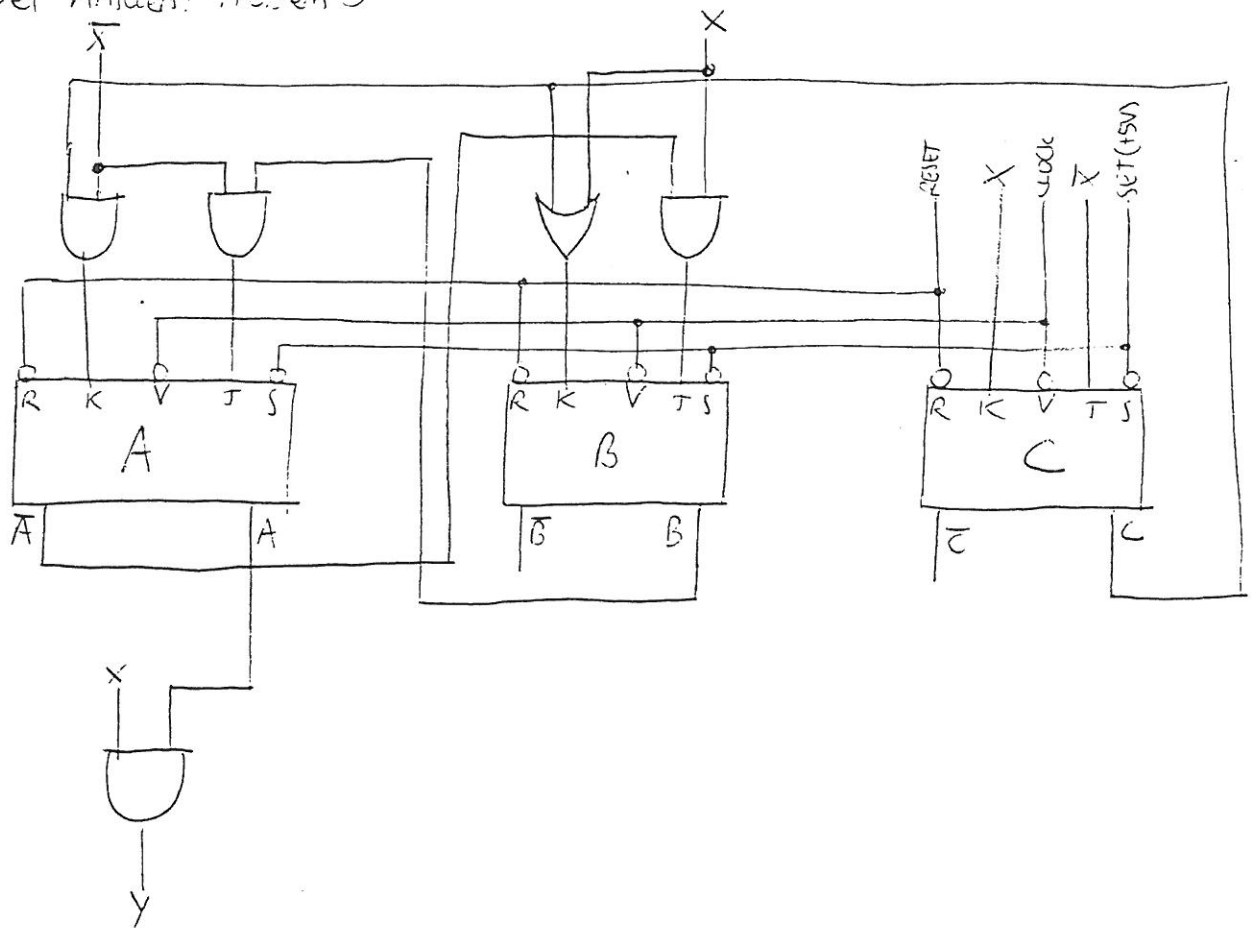


Model Answer: Problem 5



$$K_A = C \cdot \bar{X}$$

$$J_A = \bar{A} \cdot \bar{X}$$

$$K_B = C + X$$

$$J_B = \bar{A} \cdot X$$

$$Y = X \cdot A$$

$$K_C = X$$

$$J_C = \bar{X}$$

AB \ CX	00	01	11	10
00				
01				
11				
10	1	1	1	1

K_A

AB \ CX	00	01	11	10
00				
01				
11				
10	1	1	1	1

J_A

AB \ CX	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

K_B

AB \ CX	00	01	11	10
00				
01	1	1	1	1
11	1	1	1	1
10				

J_B

AB \ CX	00	01	11	10
00				
01	1	1	1	1
11	1	1	1	1
10				

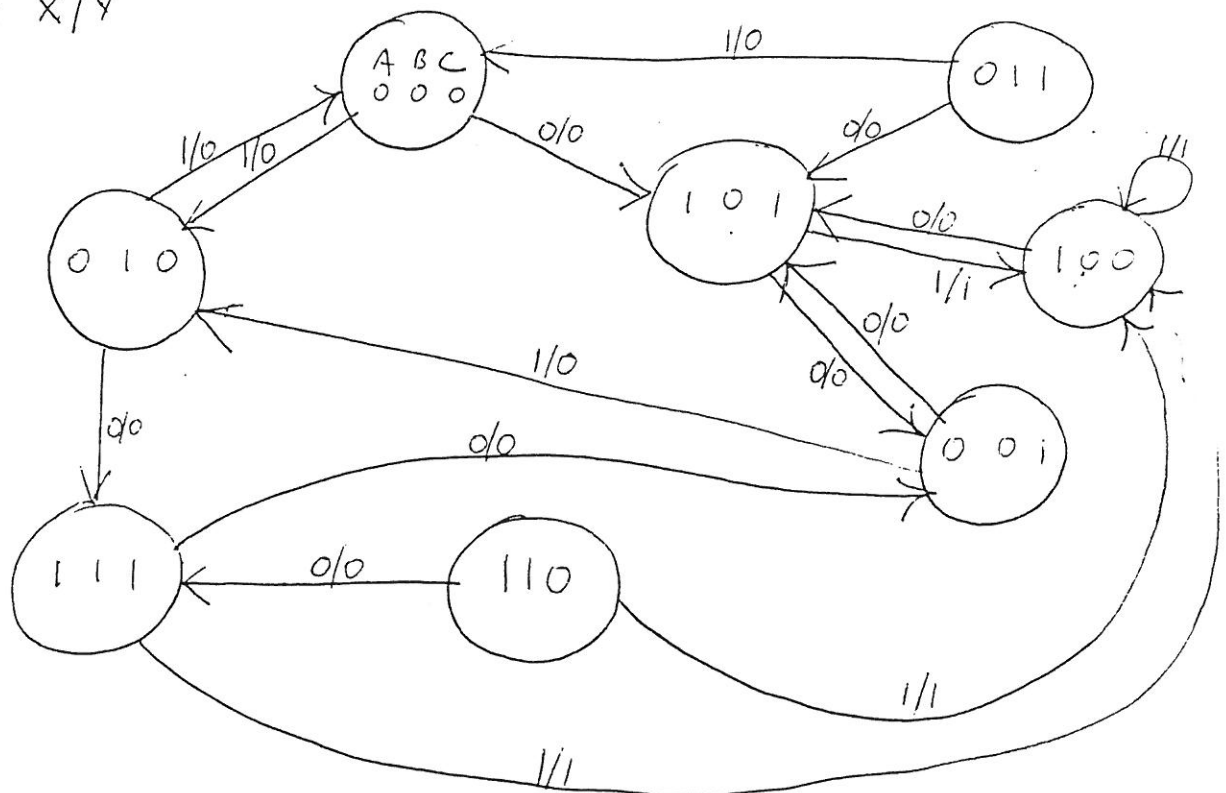
K_C

AB \ CX	00	01	11	10
00				
01				
11				
10	1	1	1	1

J_C

Present State			Input	Next State Codes						Output	True Next State		
A	B	C	X	J _A	K _A	J _B	K _B	J _C	K _C	Y	A	B	C
0	0	0	0	1	0	0	0	1	0	0	1	0	1
0	0	0	1	0	0	1	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	1	0	0	1	0	1
0	0	1	1	0	0	1	1	0	1	0	0	1	0
0	1	0	0	1	0	0	0	1	0	0	1	1	1
0	1	0	1	0	0	1	1	0	1	0	0	0	0
0	1	1	0	1	1	0	1	1	0	0	1	0	1
0	1	1	1	0	0	1	1	0	1	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	1	0	1
1	0	0	1	0	0	0	1	0	1	1	1	0	0
1	0	1	0	0	1	0	1	1	0	0	0	0	1
1	0	1	1	0	0	0	1	0	1	1	1	0	0
1	1	0	0	0	0	0	0	1	0	0	1	1	1
1	1	0	1	0	0	0	1	0	1	1	1	0	0
1	1	1	0	0	1	0	1	1	0	0	0	0	1
1	1	1	1	0	0	0	1	0	1	1	1	0	0

FORMAT: X/Y



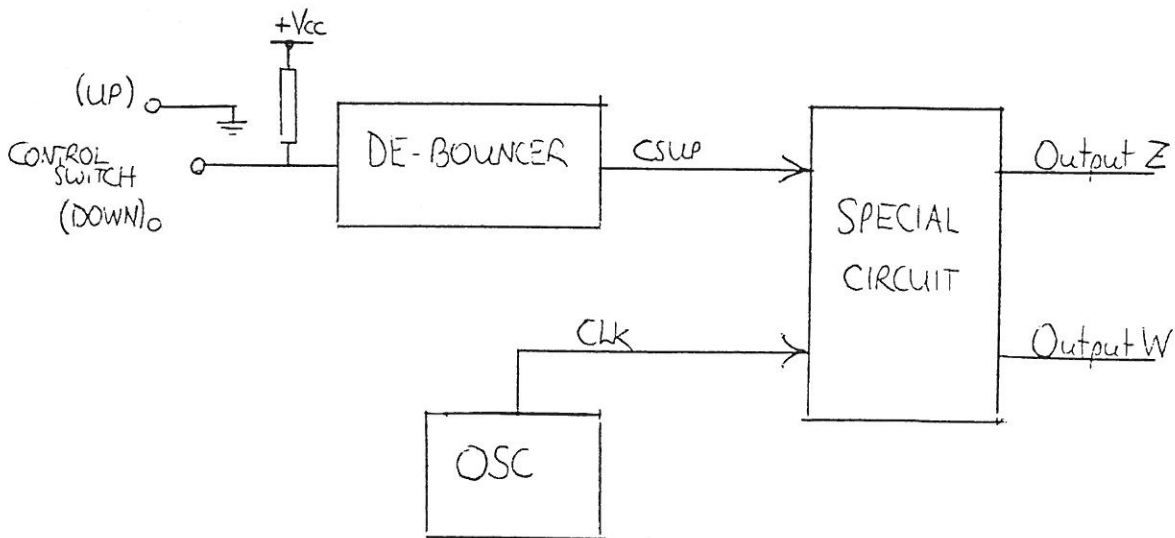
Design of Synchronous Sequential Circuits

The following design method is applicable for circuits having less than or equal to two inputs and a possible number of states ≤ 16 .

- Step 1: Receive design specifications
- Step 2: Spend time studying specifications sufficient to gain an insight into the "real" operational behavior of the circuit.
- Step 3: Make a block diagram model of your design. Identify all inputs and outputs. Note: Make certain that you completely understand when and under what conditions outputs are to be generated.
- Step 4: Design a primitive state diagram based on the information obtained from Steps 2 and 3.
- Step 5: Develop a primitive state table from the primitive state diagram and check this table systematically for possible redundant states in the state diagram.
- Step 6: If necessary, develop a simplified state diagram from the simplified primitive state table.
- Step 7: Make a state assignment using rules assigned to the states in the simplified diagram. Document your state assignment in a state map. This step may be optional if a state assignment is specified in the design.
- Step 8: Develop a PRESENT/NEXT STATE TABLE using assignments from the simplified state diagram with state assignments.
- Step 9: Develop the NEXT STATE maps using the PRESENT/NEXT STATE TABLE with state assignments. From these maps derive the NEXT STATE DECODER logic for D, T and JK Flip Flops.
- Step 10: Make a selection for your memory elements (Flip Flops).
- Step 11: From Step 3 and state diagram with state assignments, develop the output decoder logic by plotting the output maps.
- Step 12: Draw the schematic diagram.

Worked Example 1

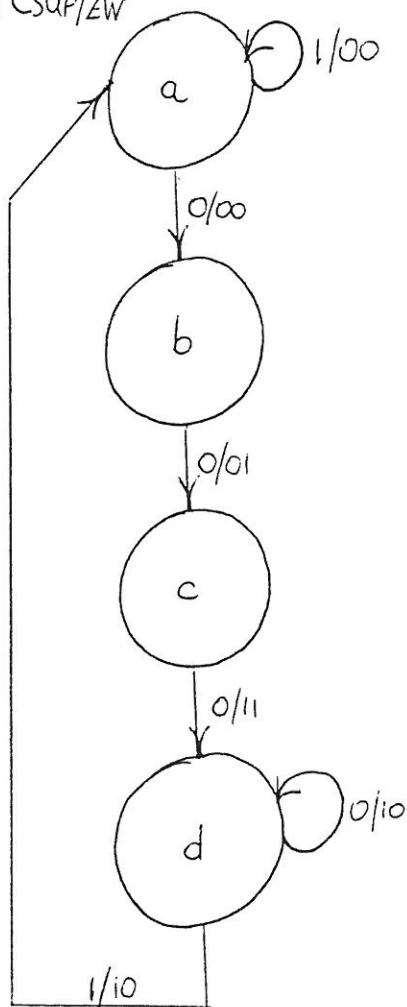
The circuit below is to control the code on the outputs Z and W based on the position of the control switch (CS). This control is prescribed as follows: When the switch is in the UP position ($CS_{UP} = 0$) the ZW code equals 00. When the circuit senses that the switch has been moved to the down position ($CS_{UP} = 1$), the output is to sequence, with the rising edge of the CLK, from $ZW = 00$ to $ZW = 01$ to $ZW = 11$ to $ZW = 10$ and then stop and hold until the switch is moved back to the up position once more, at which time the ZW code reverts to $ZW = 00$.



Design Steps

A state diagram and state table are generated to define the sequential behaviour of the circuit. Next the state diagram and state table are checked for possible state redundancy, that is, checked to determine if any unnecessary states have been introduced during the state diagram design. By inspection you find that each state is necessary.

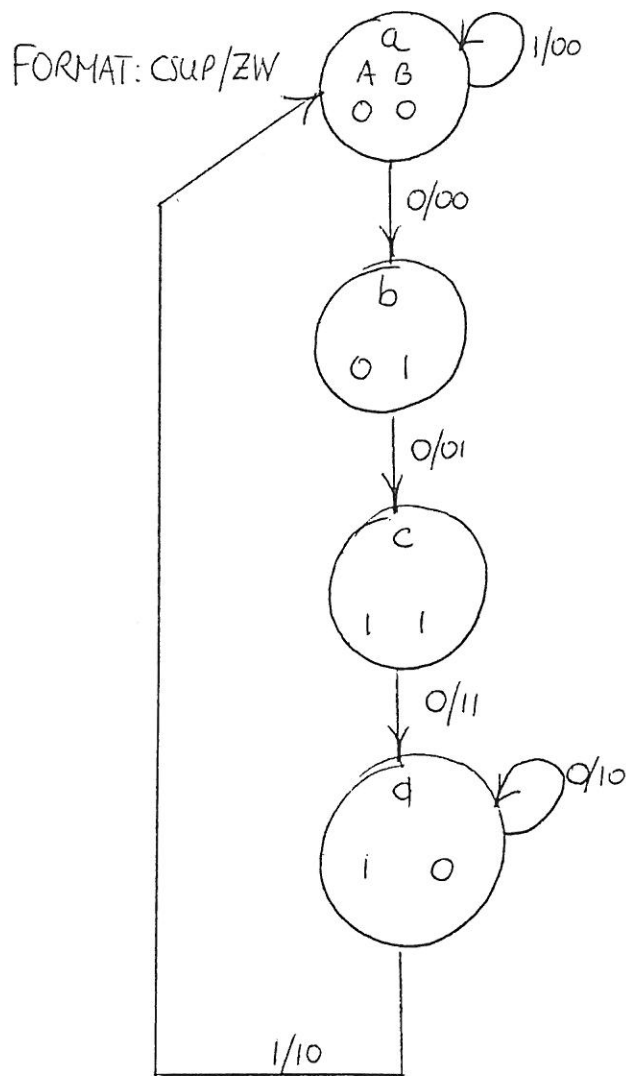
Format: CSUP/ZW



PRESENT STATE	INPUT CSUP	NEXT STATE	OUTPUTS Z W	
a	0	b	0	0
	1	a	0	0
b	0	c	0	1
	1	c	0	1
c	0	d	1	1
	1	d	1	1
d	0	d	1	0
	1	a	1	0

This table is made directly from the state diagram

Next a STATE ASSIGNMENT is made on the state diagram such that each STATE is uniquely defined (Note: the state assignment code was made identical to the output code). From this state diagram the state table with state assignments is filled out. Note the truth table form.



PRESENT STATE			INPUT	NEXT STATES		OUTPUT		
	A	B	CSUP	A	B	Z	W	
(a) 0	0	0	0	0	1	0	0	
1	0	0	1	0	0	0	0	
(b) 2	0	1	0	1	1	0	1	
3	0	1	1	1	1	0	1	
(c) 6	1	1	0	1	0	1	1	
7	1	1	1	1	0	1	1	
(d) 4	1	0	0	1	0	1	0	
5	1	0	1	0	0	1	0	

Assuming two D type flipflops are to be used, the next state maps are filled out directly from the state table with the state assignment.

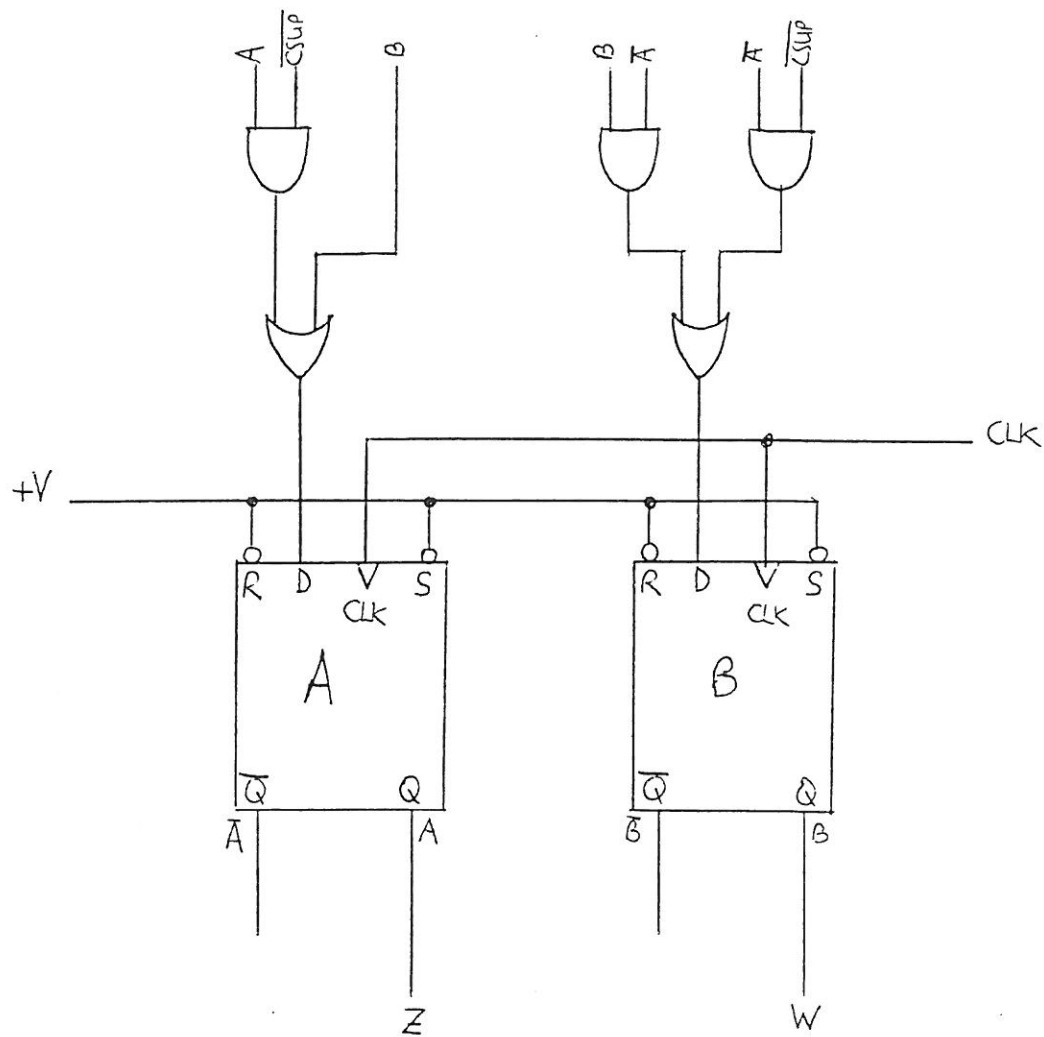
AB		00	01	11	10
CSUP	0	0	1	1	1
	1	0	1	1	0

$$D_A = B + A \cdot (\overline{CSUP})$$

AB		00	01	11	10
CSUP	0	1	1	0	0
	1	0	1	0	0

$$D_B = \overline{A} \cdot B + \overline{A} \cdot (\overline{CSUP})$$

Next the schematic diagram is drawn showing that the outputs Z and W are taken directly from the state variables A and B using a special notation for the re-definition of $A = Z$ and $B = W$

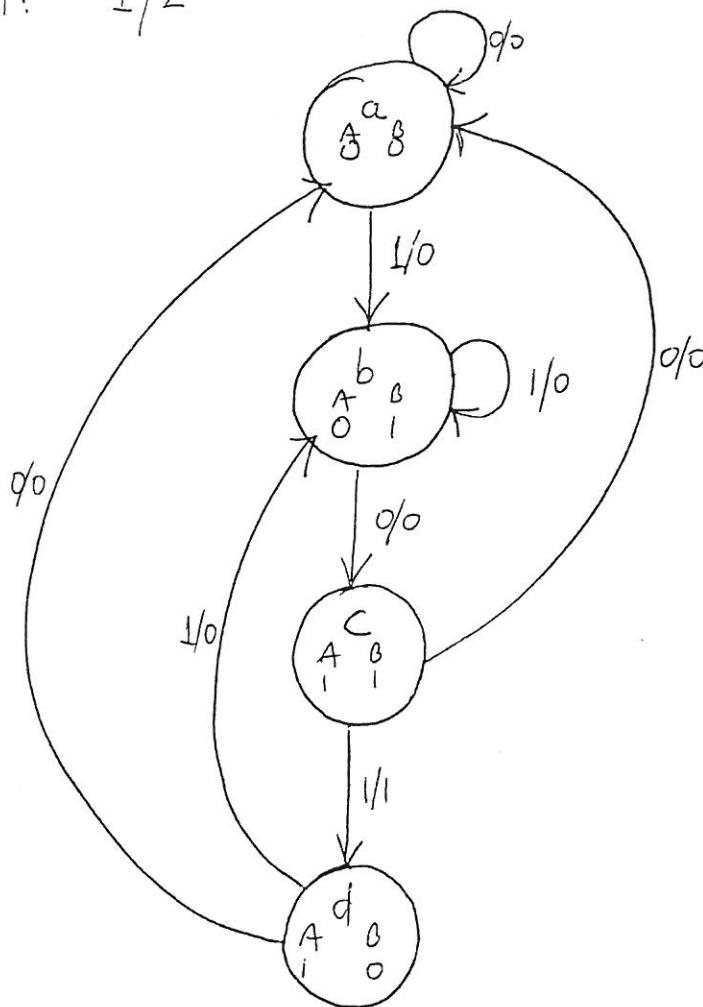


Worked Example 2

Problem: Design a sequential logic circuit that will detect the sequence 101 in a stream of serial data, input at the rate of 1 bit per clock pulse.

The circuit has a single input variable I which is either 0 or 1, and a single output Z that becomes 1 when the input sequence 101 has been received. However, it is not

Format: I/Z



State	Present State		Input	Next State		Output
	A	B		A	B	
a	0	0	0	0	0	0
	0	0	1	0	1	0
b	0	1	0	1	1	0
	0	1	1	0	1	0
c	1	0	0	0	0	0
	1	0	1	0	1	0
d	1	1	0	0	0	0
	1	1	1	1	0	1

Implement Using D-flipflops.

I \ AB	00	01	11	10
0	0	1	0	0
1	0	0	1	0

$$D_A = \bar{A}.B.\bar{I} + A.B.I$$

I \ AB	00	01	11	10
0		1		
1	1	1		1

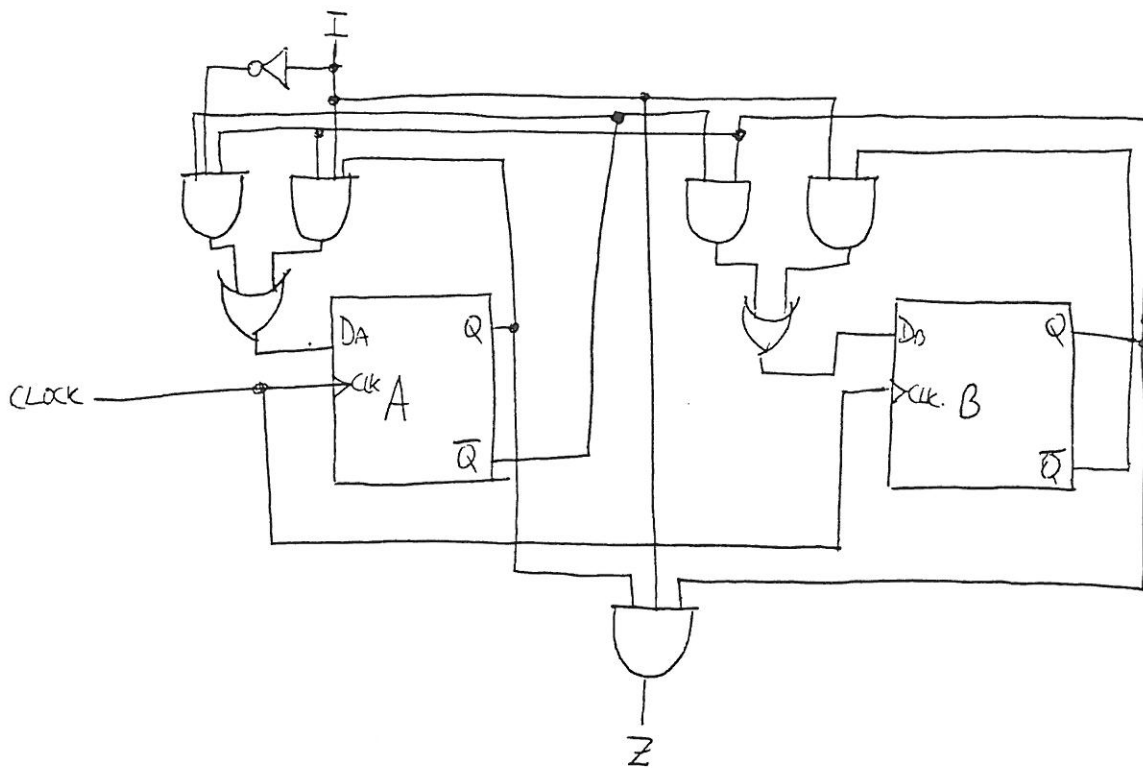
$$D_B = \bar{A}.B + \bar{B}.I$$

For Race free

$$D_B = \bar{A}.I + \bar{A}.B + \bar{B}.I$$

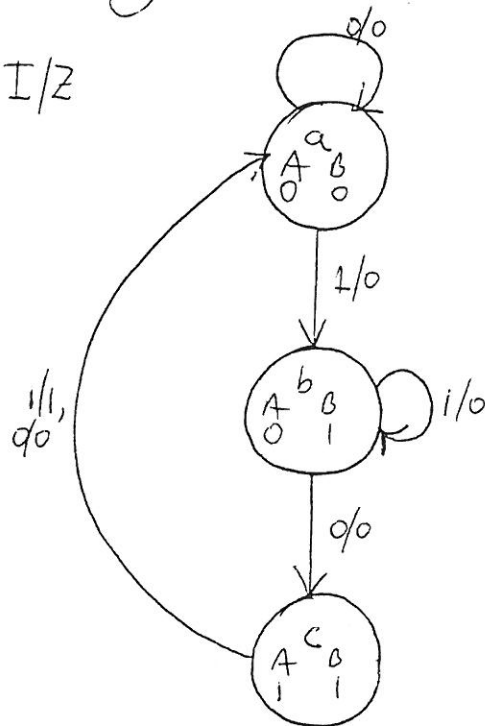
I \ AB	00	01	11	10
0				
1			1	

$$Z = A.B.I$$



This circuit could, however, have been simplified with a little care at the initial specification of the state diagram. The diagram below performs the same function.

Format: I/Z



Minimised state diagram:

Proceeding with the design (Note: state 10 is not used so can be assigned don't care states).

State.	Present State		Input	Next State		Output
	A	B		A	B	
a	0	0	0	0	0	0
	0	0	1	0	1	0
b	0	1	0	1	1	0
	0	1	1	0	1	0
c	1	1	0	0	0	0
	1	1	1	0	0	1
X	1	0	1	X	X	X
	1	0	0	X	X	X

Implementing Using D-flipflops.

AB					
I		00	01	11	10
	0	0	1	0	X
	1	0	0	0	X

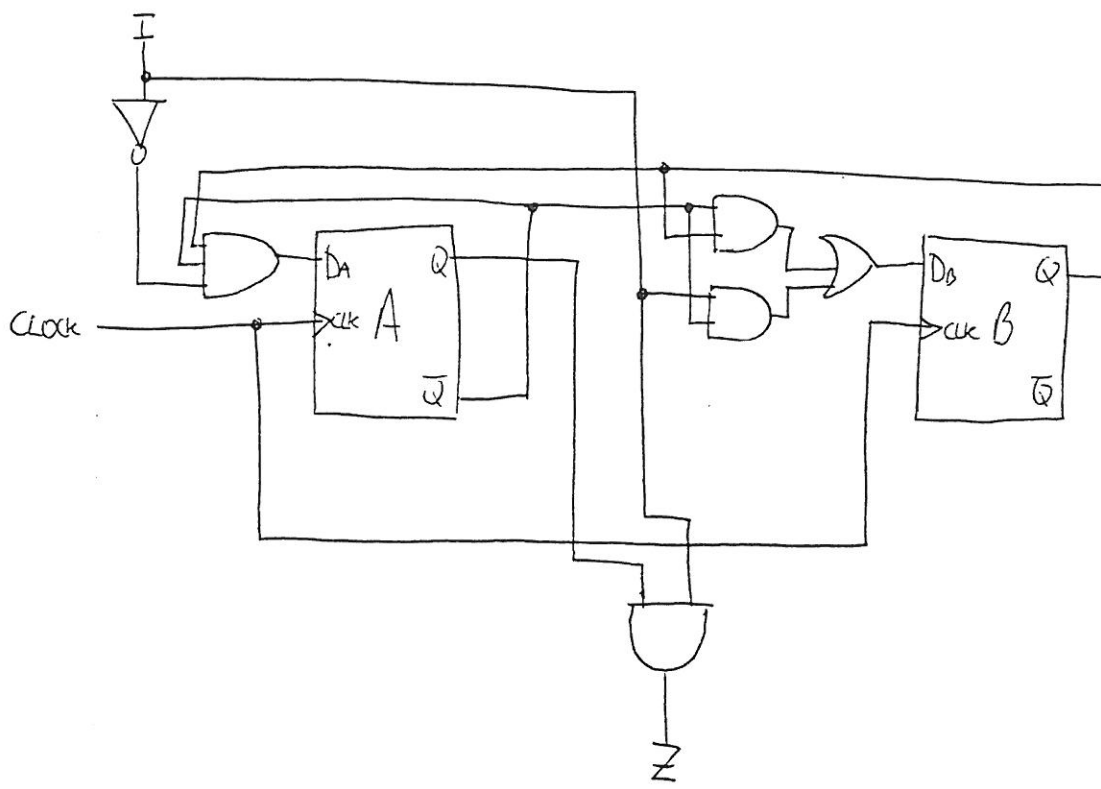
$$D_A = \bar{A} \cdot B \cdot \bar{I}$$

AB					
I		00	01	11	10
	0	0	1	0	X
	1	1	1	0	X

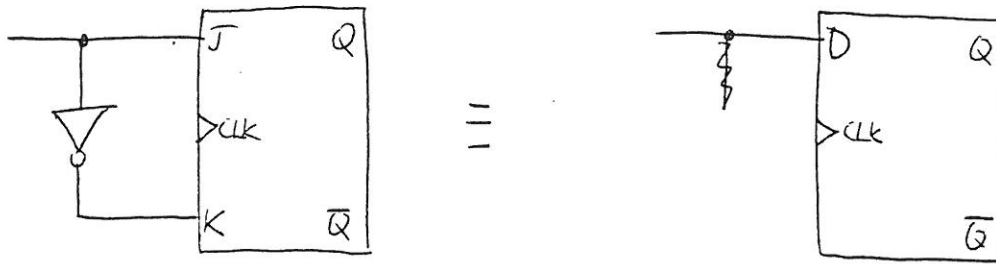
$$D_B = \bar{A} \cdot I + \bar{A} \cdot B$$

AB					
I		00	01	11	10
	0	0	0	0	X
	1	0	0	1	X

$$Z = A \cdot I$$



Converting a D-flipflop design to J-k flip flops.



$$J = D, K = \bar{D}$$

The truth tables for the J input are the same as for a D while K is its inverse.

For the second example:
Implementing using J-k flip flops.

I \ AB				
	00	01	11	10
0	0	1	~~X~~	X
1	0	0	~~X~~	X

$$J_A = A \cdot B \cdot \bar{I}$$

I \ AB				
	00	01	11	10
0	X	X	0	X
1	X	X	0	X

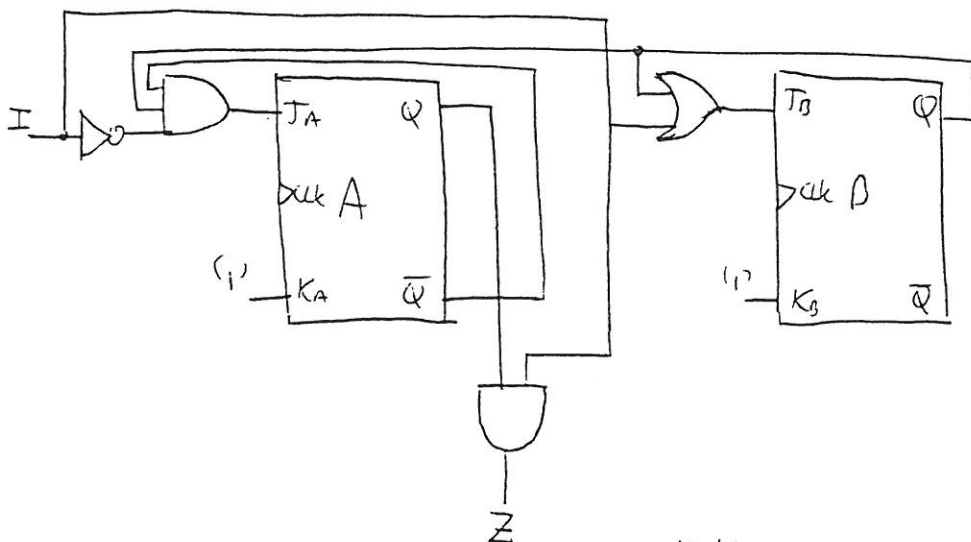
$$K_A = 1$$

I \ AB				
	00	01	11	10
0	0	1	X	X
1	1	1	X	X

$$J_B = I + B$$

I \ AB				
	00	01	11	10
0	X	X	1	X
1	X	X	1	X

$$K_B = 1$$



Worked Example 2

Design a synchronous finite state machine which will sequence from state $a=00$, to $b=11$, $c=10$ and $d=01$, then reverse itself at state a and state d if the input $COSQ$ is high. Any time the $COSQ$ input is low the circuit is to revert to state a and hold. Give the circuit using both D and J-K flip flops.

Block diagram:



X and Y are the state variables, DIR is an output needed to define the direction of the cycle. It is not used otherwise.

FORMAT: INPUT/DIR

