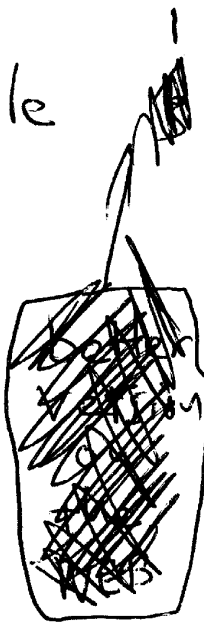
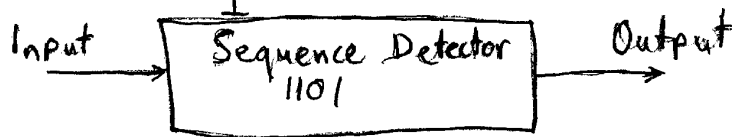


4-5 Sequential ckt design

1. Obtain either state diagram or state table from the statement of the problem.
2. If we don't have one already, obtain state table from state diagram.
3. Assign binary codes to the states.
4. Derive ~~the~~ the FF input equations from the next-state entries in the table.
5. Derive output equations from the output entries in state table.

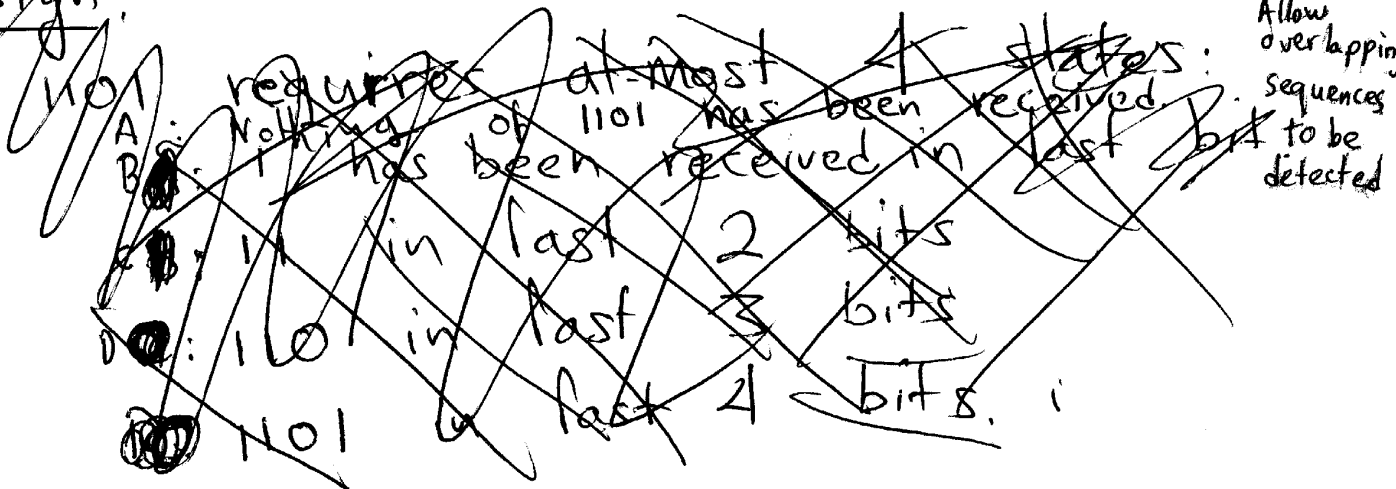


Ex 4-1 Detect 1101 by outputting 1, when the input bit-sequence matches 1101.



Input 11011100101101101
Output 000100000001001

Design:



Design

- Use different states:

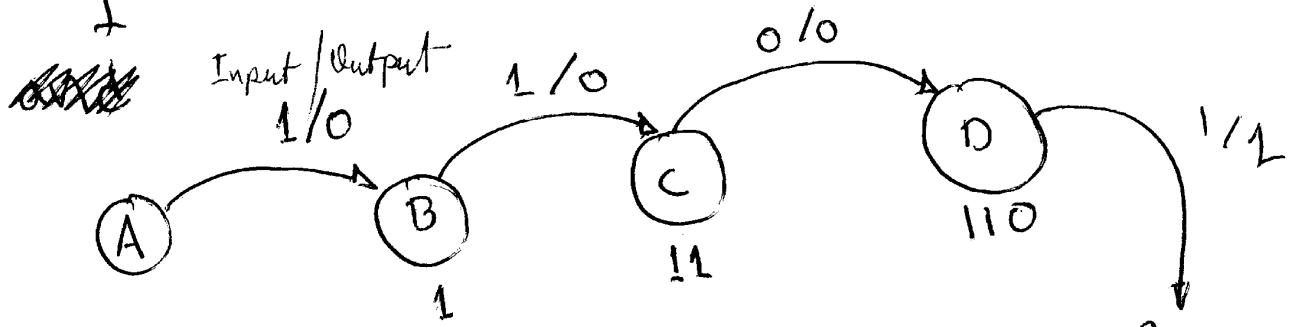
A: nothing of 1101 received yet.
Meaning all 0s are inputted.

B: got first 1 of 1101.

C: got first 11 of 1101

D: got first 110 of 1101

- At D, a 1 input means the entire sequence has been received, output 1.



Where does this arrow lead to?

- 1101 contains the first 1 of a possible another 1101 to be detected. (Suppose that overlapping sequences could be detected, i.e. Arrow goes to B. 1101101)

We must consider all possible inputs to each state:

For A: 0 keeps us in A. (1 not received).

For B: 0 means 10 which is not the beginning of 1101. Back to A.

For C: 1 means 111 with last 2 digits the ones in 1101, stay in C.

For D: 0 means 1100 which is not part of 1101. Back to A.

In other words:

(i) $1100 \neq 1101 \Rightarrow \text{output} = 0$

(ii) $\begin{array}{c} \widetilde{1100} \\ \underline{1101} \end{array}$ does not match. (Do not stay in D)

(iii) $\begin{array}{c} \widetilde{1100} \\ \underline{1101} \end{array}$ does not match. (Do not go to C)

(iv) $\begin{array}{c} \widetilde{1100} \\ \underline{1101} \end{array}$ does not match (Do not go to B)

\Rightarrow No match, go back to A.

Useful hints for drawing state diagrams

(i) move to the next state.

(ii) stay in the current one.

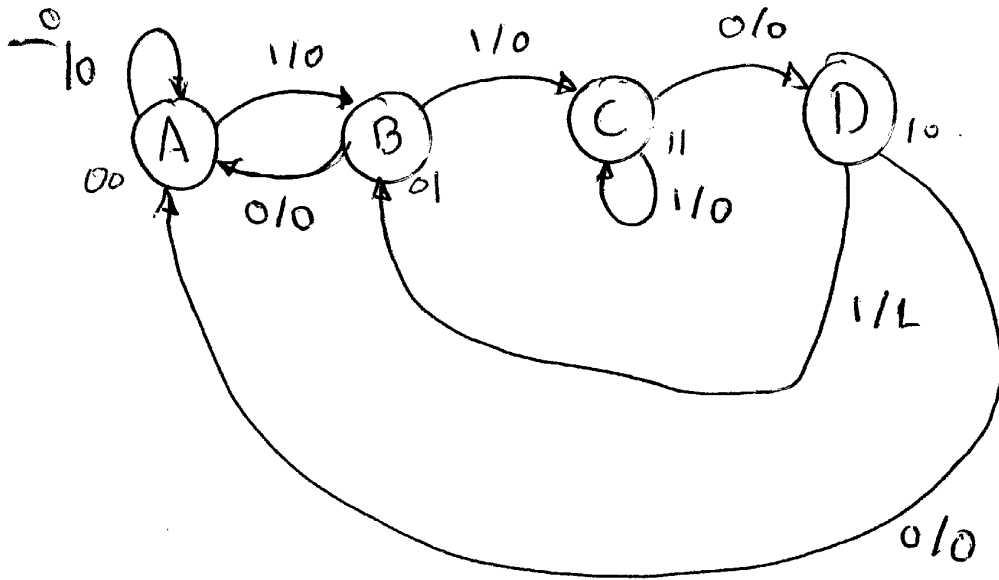
(iii) go back 1,

(iv) go back 2, ...

NB: Try to

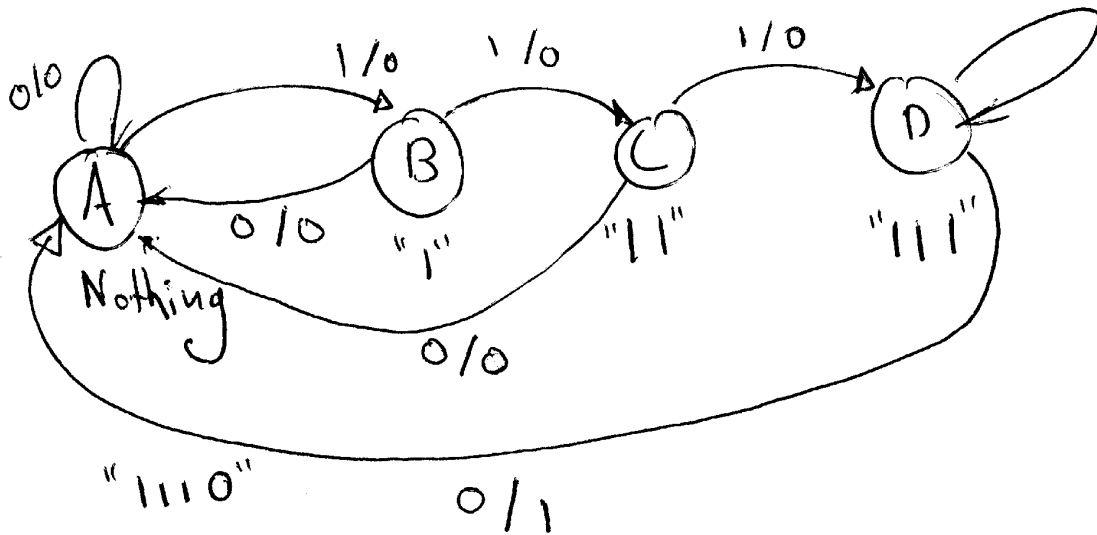
CONSIDER IN THIS ORDER!

Final Diagram:



Another Example: Detect 1110

1/0



In original eg, from state diagram to transition table, using D FFs & JK FFs

Assign the states: (not necessarily optimal):

- A is 00
- B is 01
- C is 11
- D is 10

} note assignment in an attempt to minimize logic.

Make table from: Present states & Inputs

to Next states & outputs

Recall:

Q(t) → Q(t+1)	J K
0 → 0	0 X
0 → 1	1 X
1 → 0	X 1
1 → 1	X 0

Mealy design

Present State	Input I	Next State	Output	D FF inputs			
				J	K	J	K
A 00	0	00	0	0	X	0	X
	1	01	0	0	X	1	X
	0	00	0	0	0	X	1
	1	01	0	0	1	X	0
B 01	0	11	0	X	0	X	1
	1	10	0	X	0	X	0
C 11	0	11	0	X	1	0	X
	1	00	0	X	1	1	X
D 10	0	00	0				
	1	01	1				

To finish the problem, use k-maps to find minimal CKTs realizing the D FFs inputs.