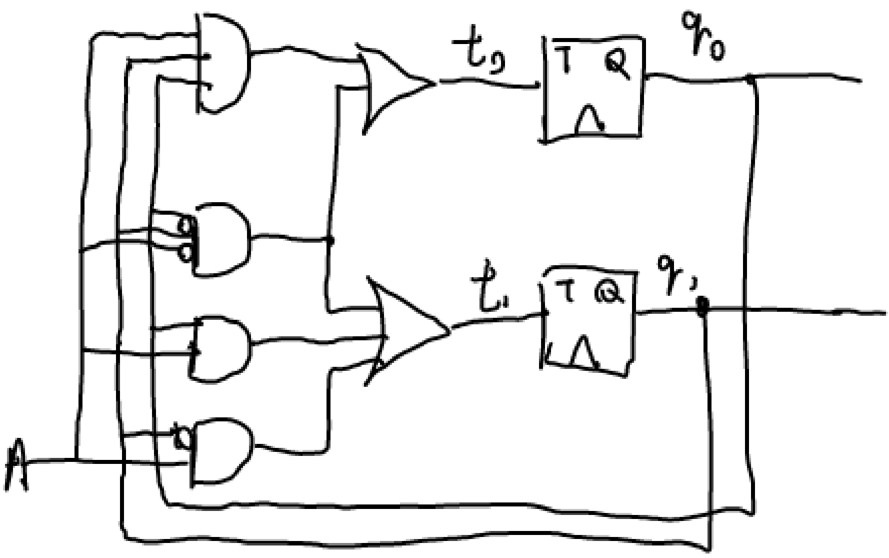
q_1q_0 \ a	0	1
00	0	1
01	0	1
11	0	1
10	1	0

q_1q_0 \ a	0	1
00	0	0
01	0	0
11	0	1
10	1	0

$$t_0 = a q_1 q_0 + \bar{a} q_1 \bar{q}_0$$

$$t_1 = a \bar{q}_1 + a q_0 + \bar{a} q_1 \bar{q}_0$$



JK:

q_1q_0 \ a	0	1
00	0	1
01	0	1
11	X	X
10	X	X

q_1q_0 \ a	0	1
00	X	X
01	X	X
11	0	1
10	1	0

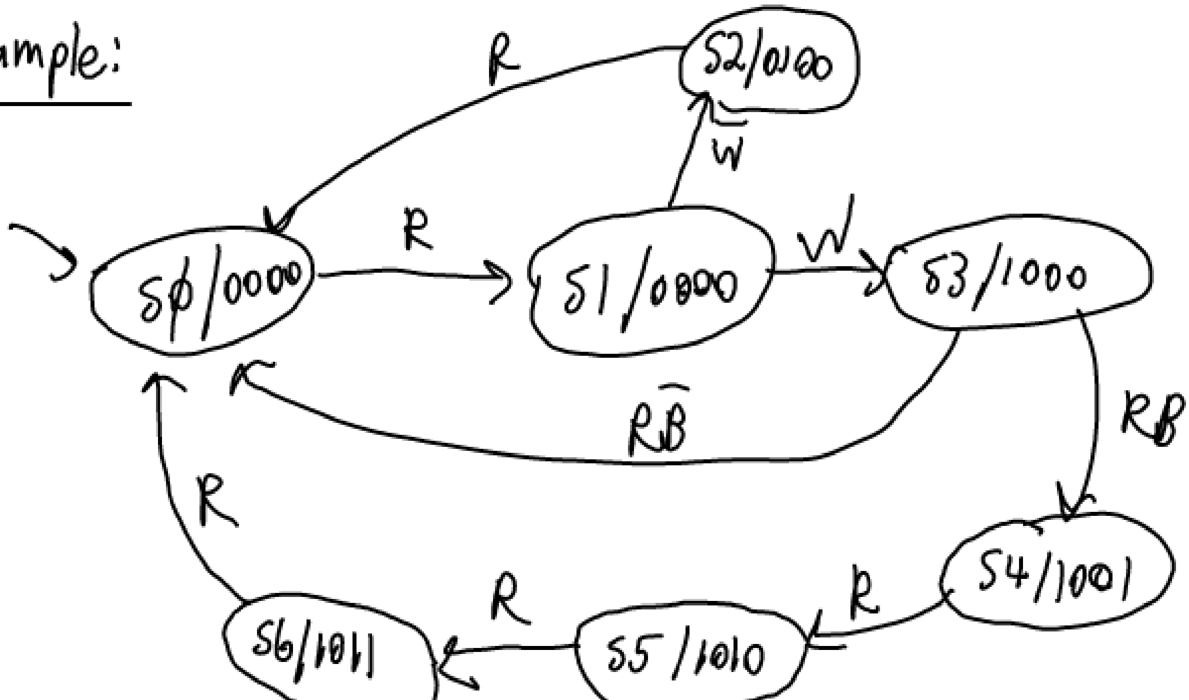
q_1q_0 \ a	0	1
00	0	0
01	X	X
11	X	X
10	1	0

q_1q_0 \ a	0	1
00	X	X
01	0	0
11	0	1
10	X	X

State Assignment

* Typically, a state diagram has symbolic names for states → We need to assign binary patterns to each state → called state assignment.
* Different ways to do it.

Example:



3 inputs / 4 outputs.

Method 1: Minimum number of flipflops.

* Use binary patterns s.t. we use as few FF_s as possible.

n states $\rightarrow \lceil \log_2 n \rceil$ ff_s

In this example, we need 3 ff_s.

One pattern to one state. If all binary patterns are not used it's okay.
Here are unused states

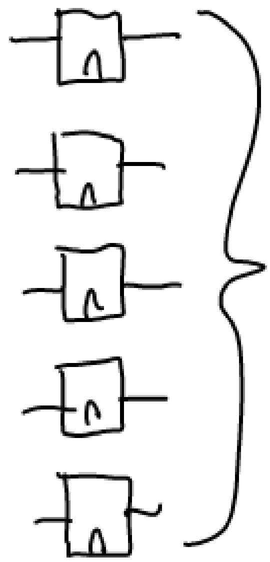
S0 ← 000
S1 ← 011
S2
S3 ← 001
S4 etc 010
S5 100
S6 101
110
111

Method 2: Output Encoding

* Sometimes we can take circuit outputs directly from the state. (i.e., no output logic). (maybe appropriate for a moore machine).

State	Output	extra bit
S0	0000	0
S1	0000	1
S2	0100	0
S3	1000	0
S4	1001	0
S5	1010	0
S6	1011	0

In this example:



circuit outputs

Method 3: One-hot encoding

* use one ff for each state

n states $\rightarrow$ n ffs.

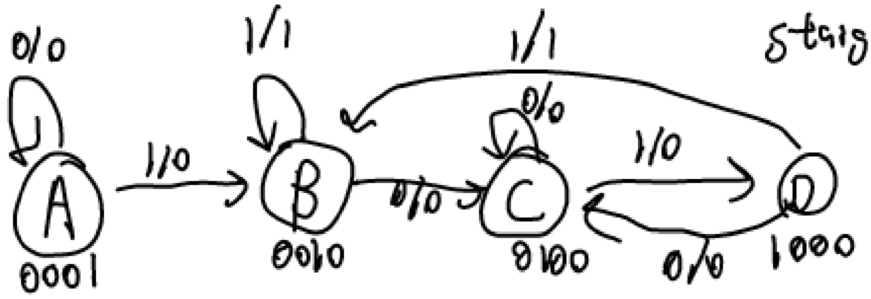
binary pattern for state j is

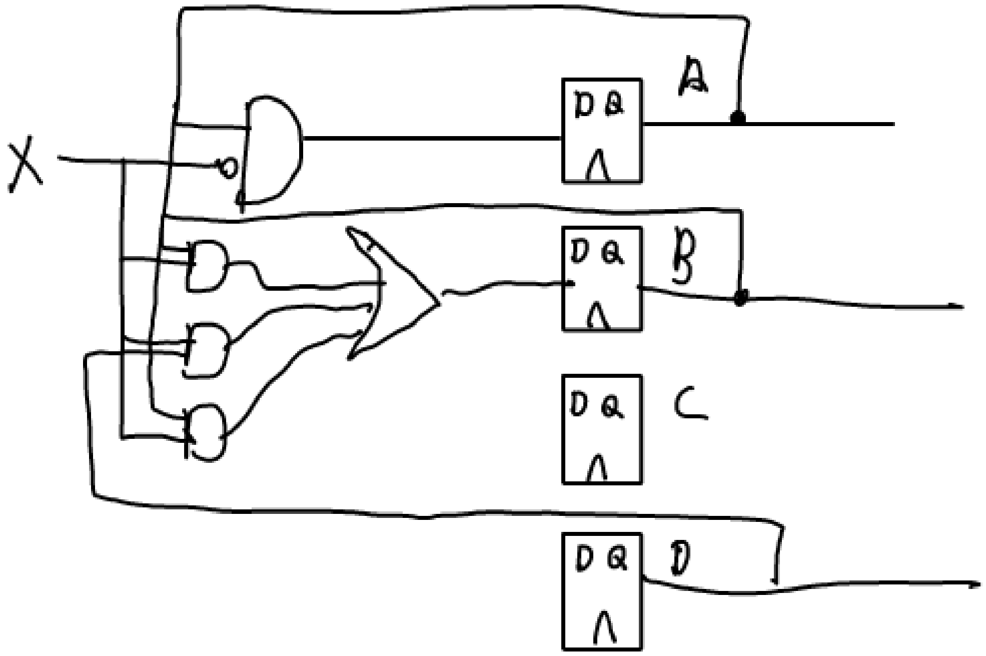
$00\ldots010\ldots0$
$\uparrow$
j

S0 ← 0000001
S1 ← 0000010
S2 ← 0000100
S3 ← 0001000
S4 ← 0010000
S5 ← 0100000
S6 ← 1000000

disadvantage: lots of FFs
advantage: - logic usually simple (for input AND output)
- ff DFFs can go straight to circuit.

Example:




X
D Q
A
D Q
B
D Q
C
D Q
D