Exercise Q3.17

- Design an FSM to keep track of the mood of four students working in the digital design lab. Each Student is either:

  1. Happy (the circuit works)
  2. Sad (the circuit blew up)
  3. Busy (working on the circuit)
  4. Clueless (confused about the circuit)
  5. Asleep (face down on the circuit board)

- How many states does the FSM have ? What is the minimum number of bits necessary to represent these states ?

Exercise Q3.17 Solution

- Each Student can either be:

  1. Happy (the circuit works).
  2. Sad (the circuit blew up).
  3. Busy (working on the circuit).
  4. Clueless (confused about the circuit).
  5. Asleep (face down on the circuit board).

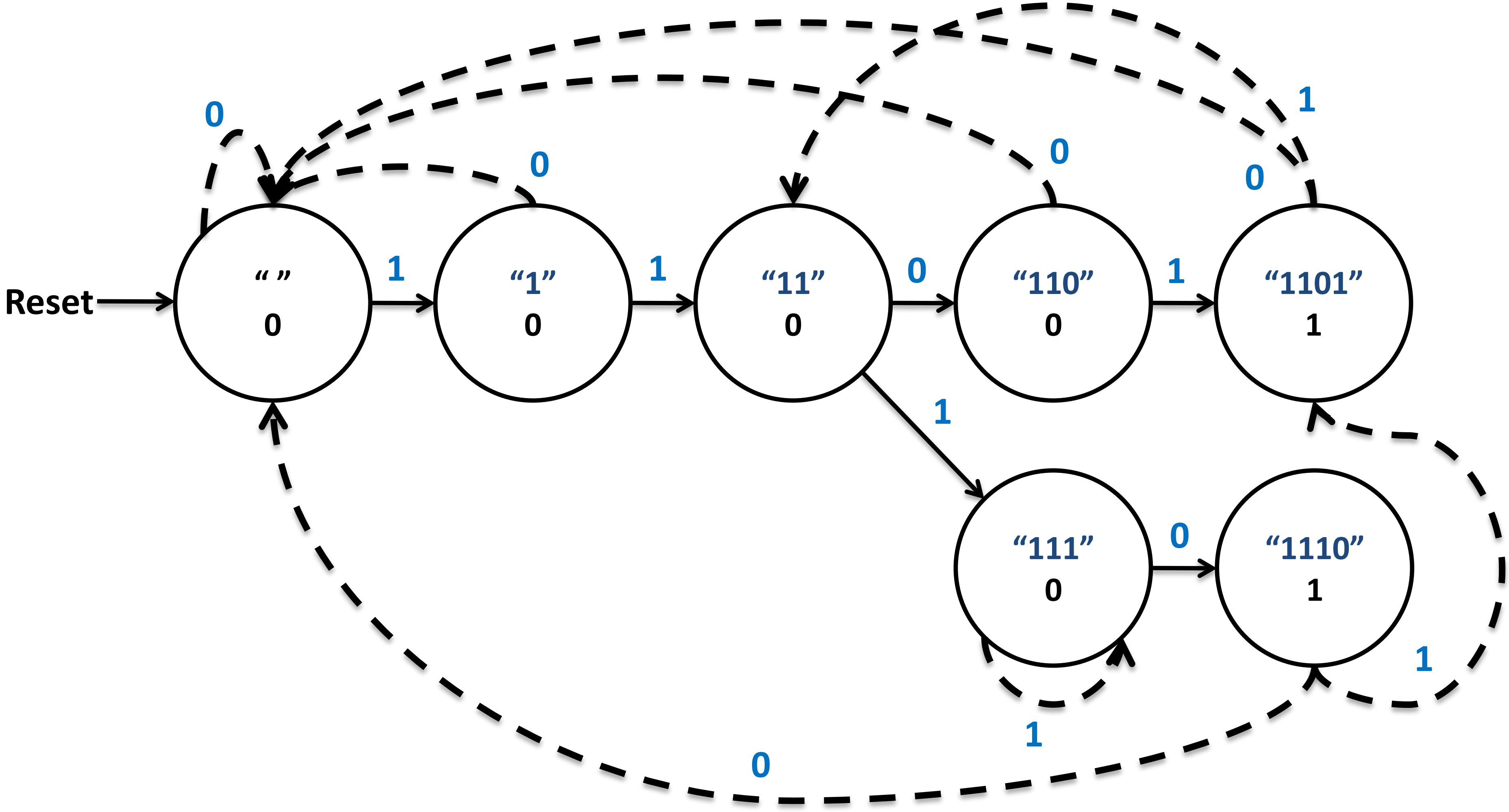
- So each student can be in five different states. Hence, we can say that overall we have 625 distinct states (5 possible states for 4 students, $5^4 = 625$).

- For 625 distinct states, we need a minimum of ceiling($\log_2 625$) = 10 bits.

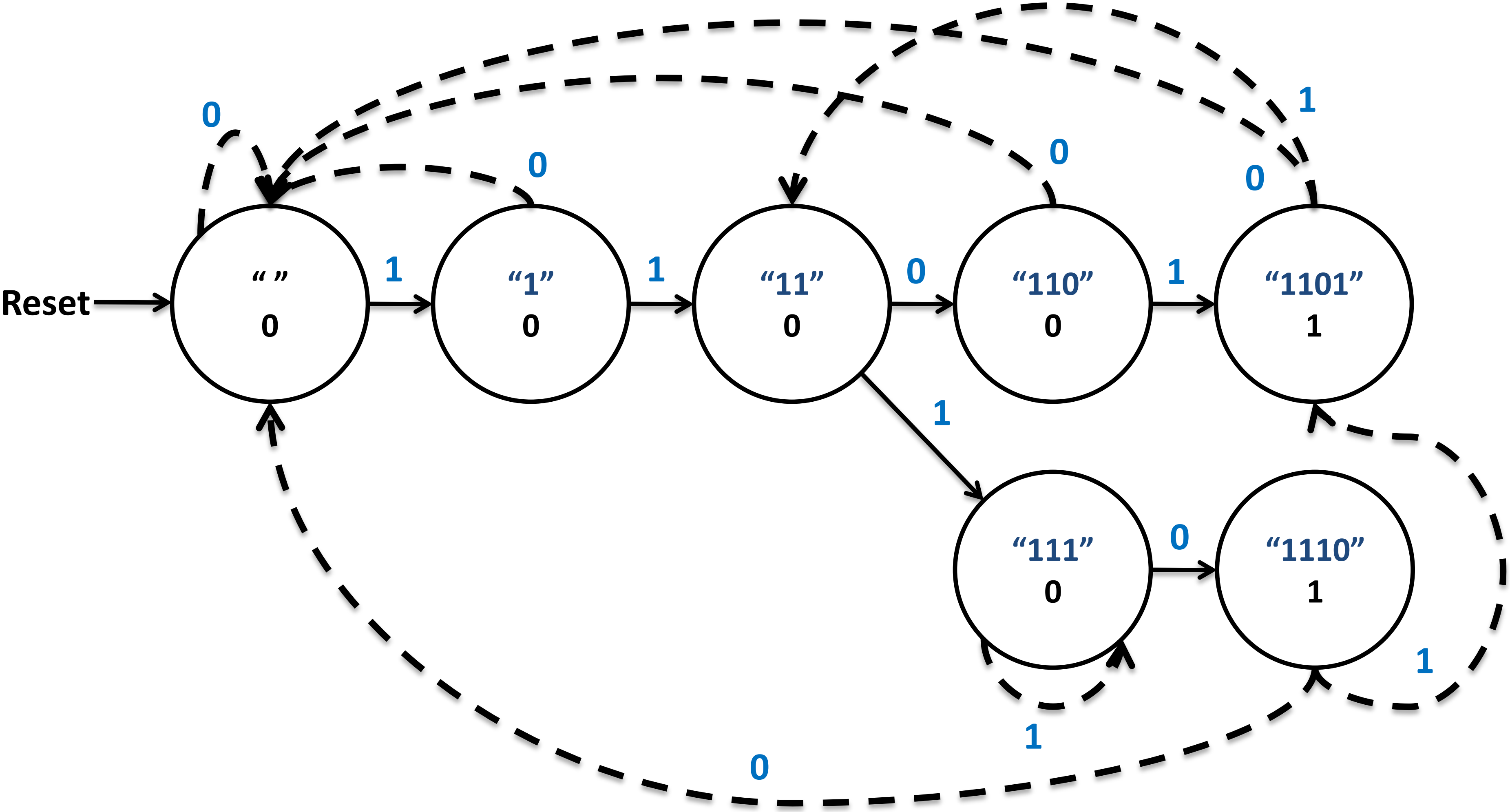
Exercise Q3.22

- Design an FSM that recognizes 1101 or 1110.
  - Draw state transition diagram (use as few states as possible).
  - Choose state encodings.
  - Write state transition and output table using the encodings.
  - Write next state equations and output equations.

Exercise Q3.22 Solutuon



State encoding:

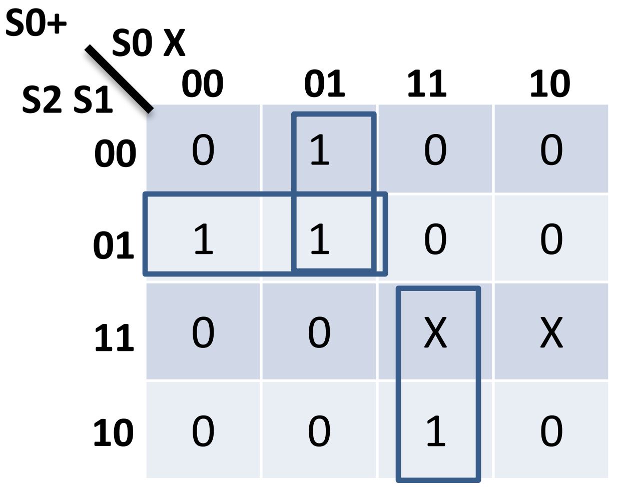


A = 000 ; B = 001 ; C = 010 ; D = 011; E = 100 ; F = 101 ; G = 110

State transition and output table:

Present State S2 S1 S0	X = 0	X = 1	F
	NS S2+ S1+ S0+	NS S2+ S1+ S0+	
0 0 0	0 0 0	0 0 1	0
0 0 1	0 0 0	0 1 0	0
0 1 0	0 1 1	1 0 1	0
0 1 1	0 0 0	1 0 0	0
1 0 0	0 0 0	0 1 0	1
1 0 1	1 1 0	1 0 1	0
1 1 0	0 0 0	1 0 0	1

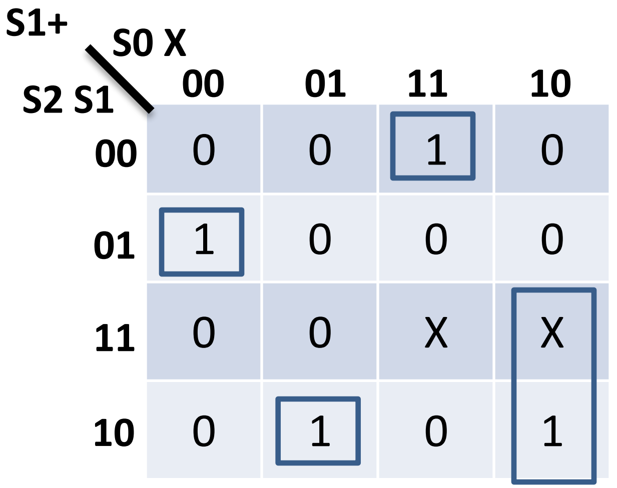
K-Maps:



S0+

S2 S1 \ S0 X	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	0	0	X	X
10	0	0	1	0

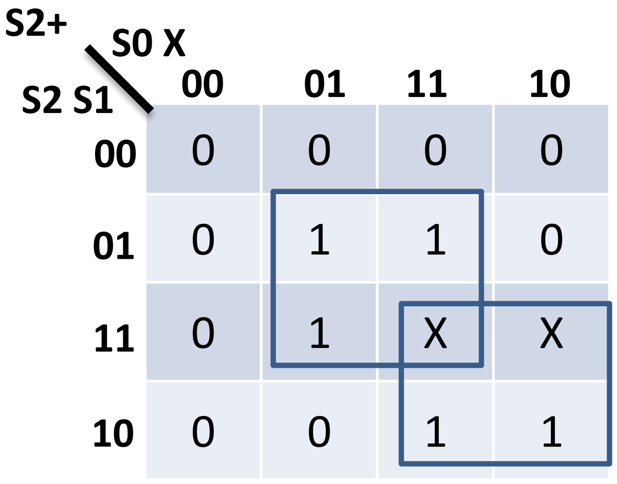
S0+ = S0'.S1.S2' + X.S0'.S2' + X.S0.S2



S1+

S2 S1 \ S0 X	00	01	11	10
00	0	0	1	0
01	1	0	0	0
11	0	0	X	X
10	0	1	0	1

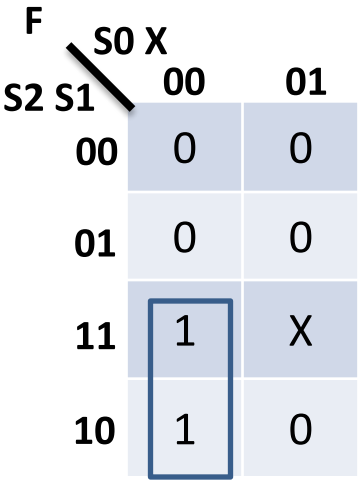
S1+ = X.S0.S1'.S2' + X'.S0'.S1.S2' + X.S0'.S1'.S2 + X'.S0.S2



S2+

S2 S1 \ S0 X	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	X	X
10	0	0	1	1

S2+ = S0.S2 + X.S1



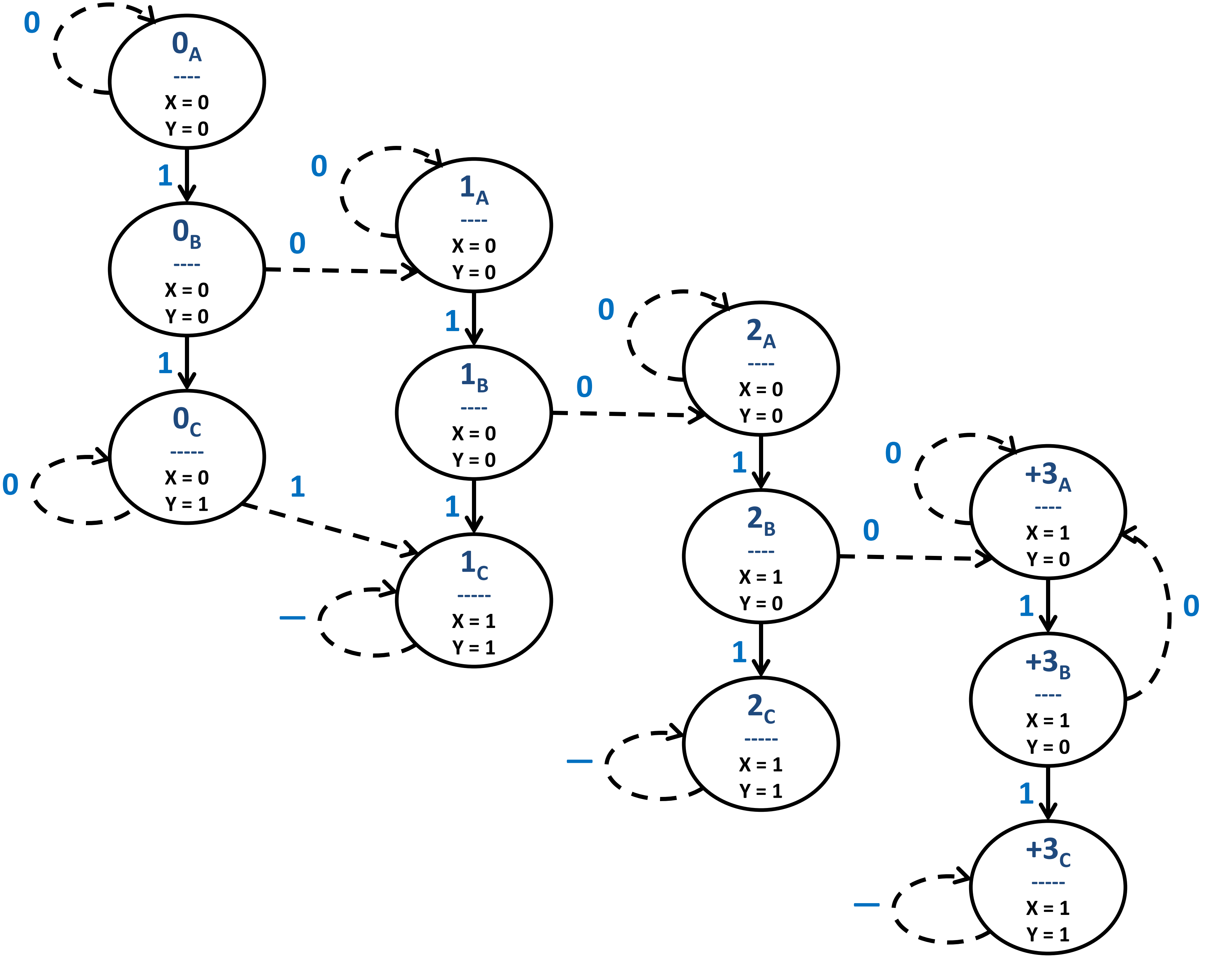
F

S2 S1 \ S0 X	00	01
00	0	0
01	0	0
11	1	X
10	1	0

F = S0'.S2

Exercise Q3.27

- Design an FSM with one input, *A,* and two outputs, *X* and *Y.*
- *X* should be 1 if *A* has been 1 for at least three cycles altogether (not necessarily consecutively).
- *Y* should be 1 if *A* has been 1 for at least two consecutive cycles.
- Show your state transition diagram, encoded state transition table, next state and output equations, and schematic.


0
0_A

X = 0
Y = 0
1
0_B

X = 0
Y = 0
0
1
0_C

X = 0
Y = 1
0
1
0
1_A

X = 0
Y = 0
1
1_B

X = 0
Y = 0
0
1
1_C

X = 1
Y = 1
—
0
2_A

X = 0
Y = 0
1
2_B

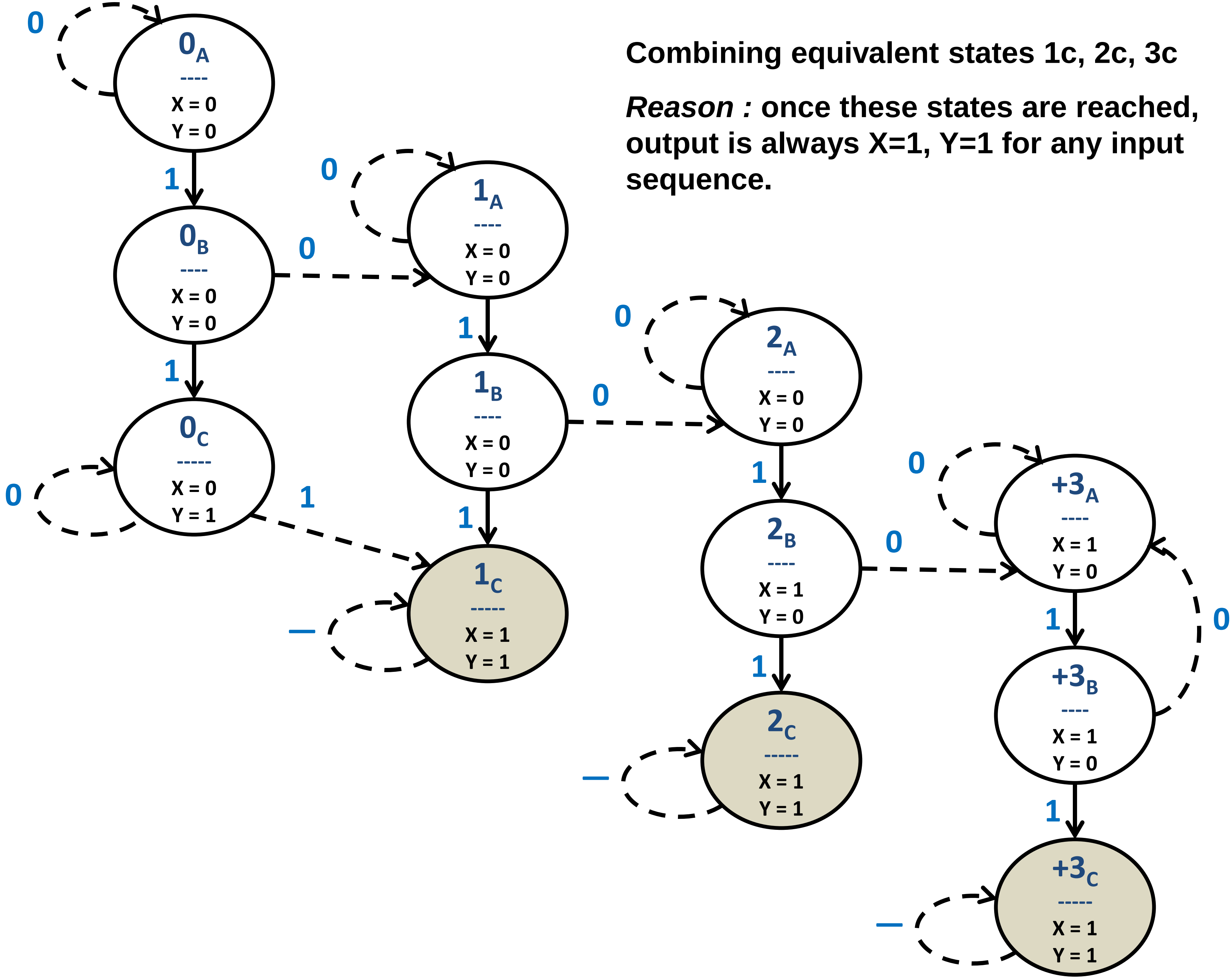
X = 1
Y = 0
0
1
2_C

X = 1
Y = 1
—
0
$+3_A$

X = 1
Y = 0
1
0
$+3_B$

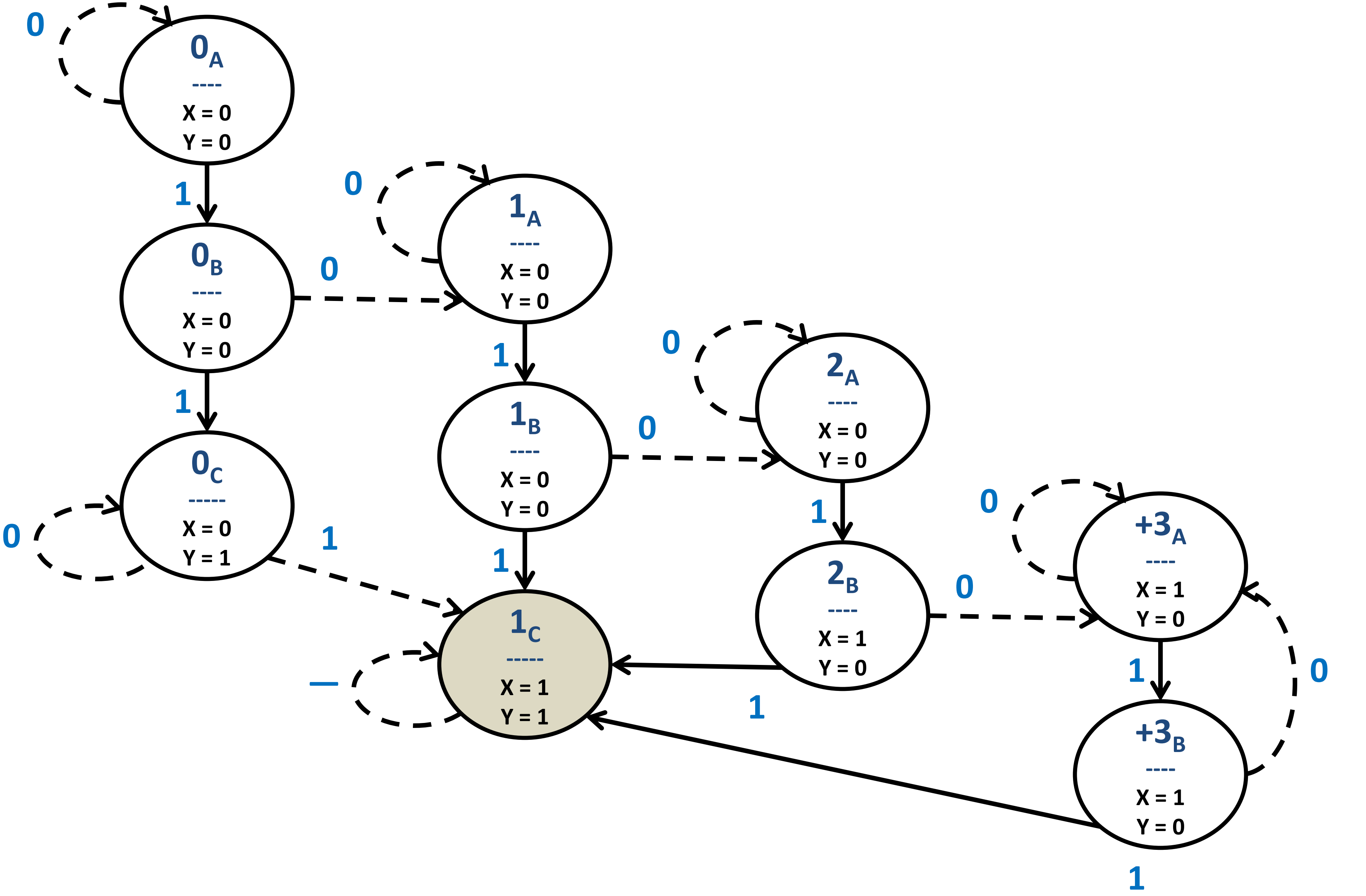
X = 1
Y = 0
1
$+3_C$

X = 1
Y = 1
—



Combining equivalent states 1c, 2c, 3c

Reason : once these states are reached, output is always X=1, Y=1 for any input sequence.


0_A

X = 0
Y = 0
0_B

X = 0
Y = 0
0_C

X = 0
Y = 1
1_A

X = 0
Y = 0
1_B

X = 0
Y = 0
1_C

X = 1
Y = 1
2_A

X = 0
Y = 0
2_B

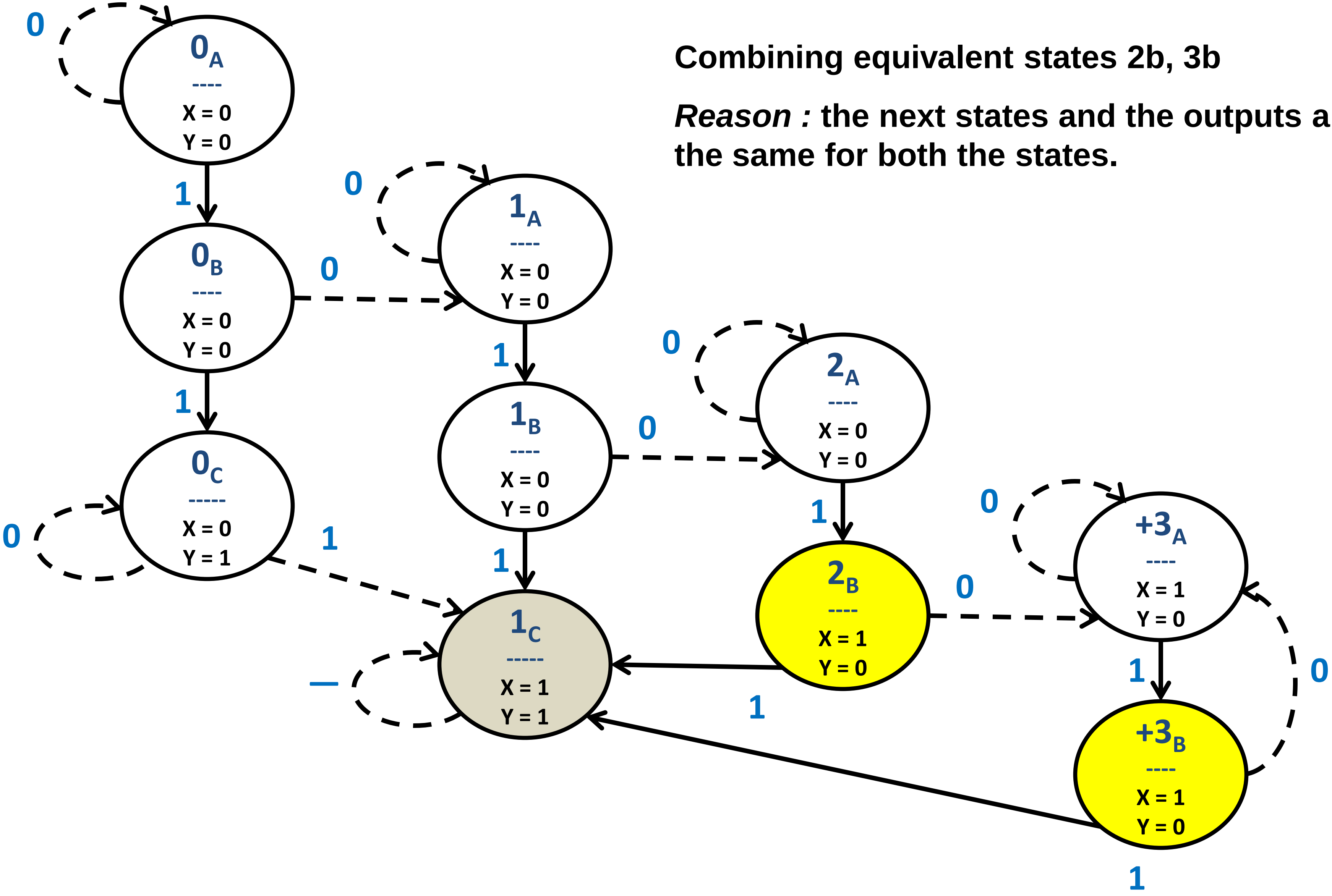
X = 1
Y = 0
$+3_A$

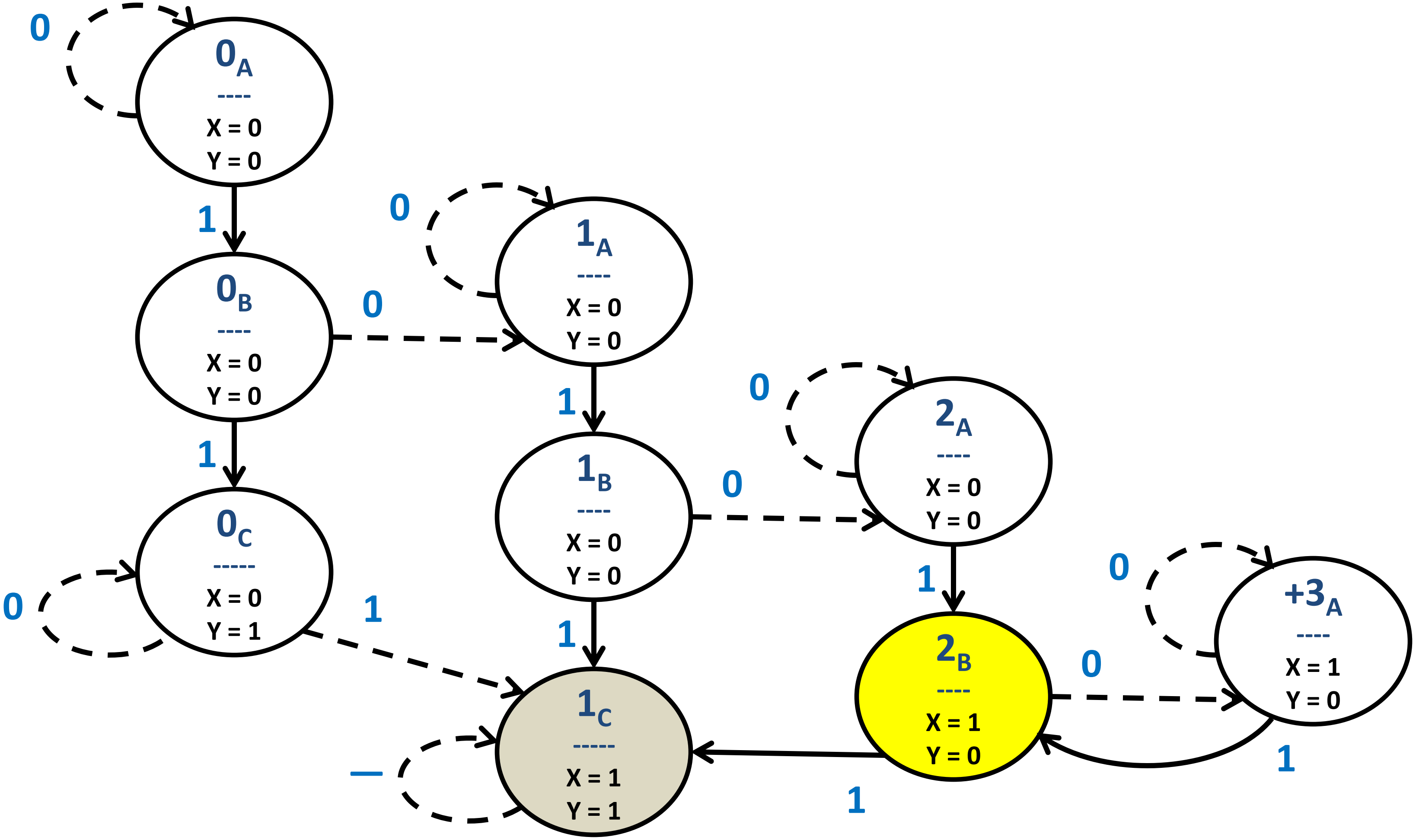
X = 1
Y = 0
$+3_B$

X = 1
Y = 0
0
1
0
1
0
1
0
1
—
0
1
1
0
1
0
1
0
1
0
1

Combining equivalent states 2b, 3b

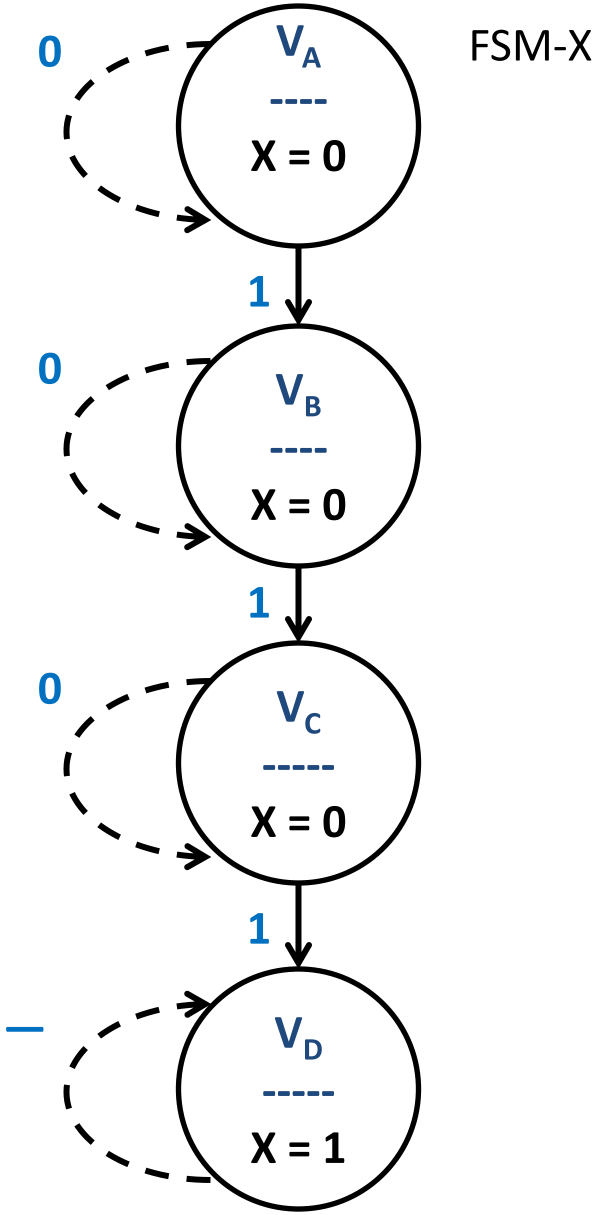
Reason : the next states and the outputs are the same for both the states.




0_A
X = 0
Y = 0
0_B
X = 0
Y = 0
0_C
X = 0
Y = 1
1_A
X = 0
Y = 0
1_B
X = 0
Y = 0
1_C
X = 1
Y = 1
2_A
X = 0
Y = 0
2_B
X = 1
Y = 0
$+3_A$
X = 1
Y = 0
0
1
—

- Another approach is to design an FSM for X (FSM-X) and a separate FSM for Y (FSM-Y)

- Then “simulate” the execution from the “initial states”

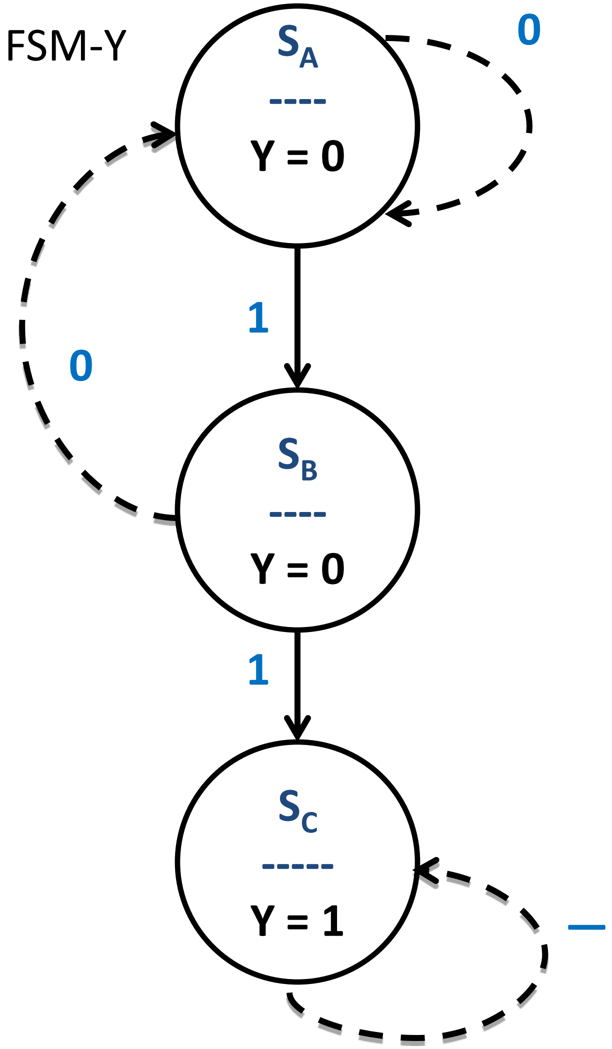

FSM-X
V_A

X = 0
0
1
V_B

X = 0
0
1
V_C

X = 0
0
1
V_D

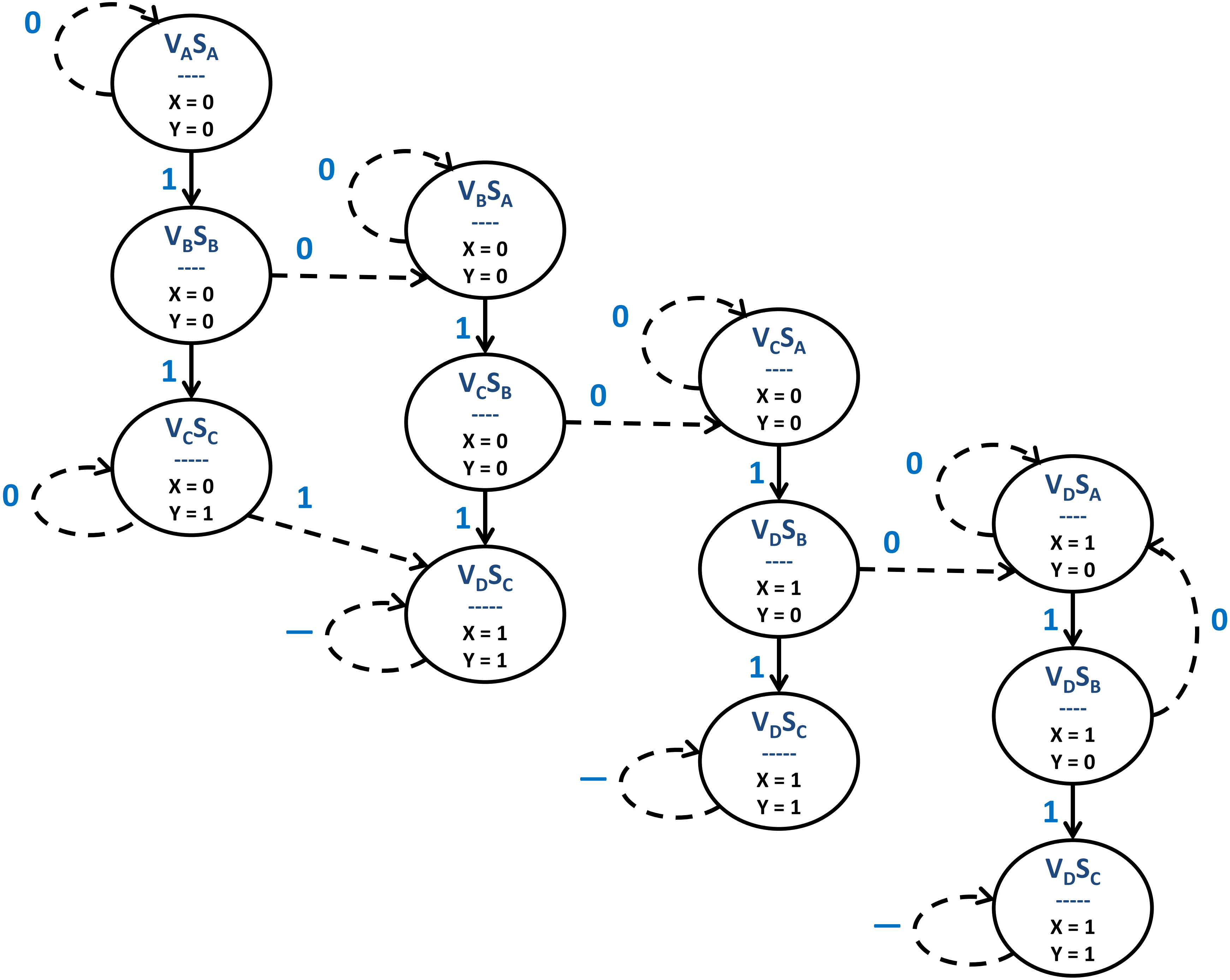
X = 1
—


FSM-Y
S_A

Y = 0
0
1
0
S_B

Y = 0
1
S_C

Y = 1
—


0
V_AS_A

X = 0
Y = 0
1
V_BS_B

X = 0
Y = 0
1
V_CS_C

X = 0
Y = 1
0
1
0
0
V_BS_A

X = 0
Y = 0
1
V_CS_B

X = 0
Y = 0
1
V_DS_C

X = 1
Y = 1
—
0
0
V_CS_A

X = 0
Y = 0
1
V_DS_B

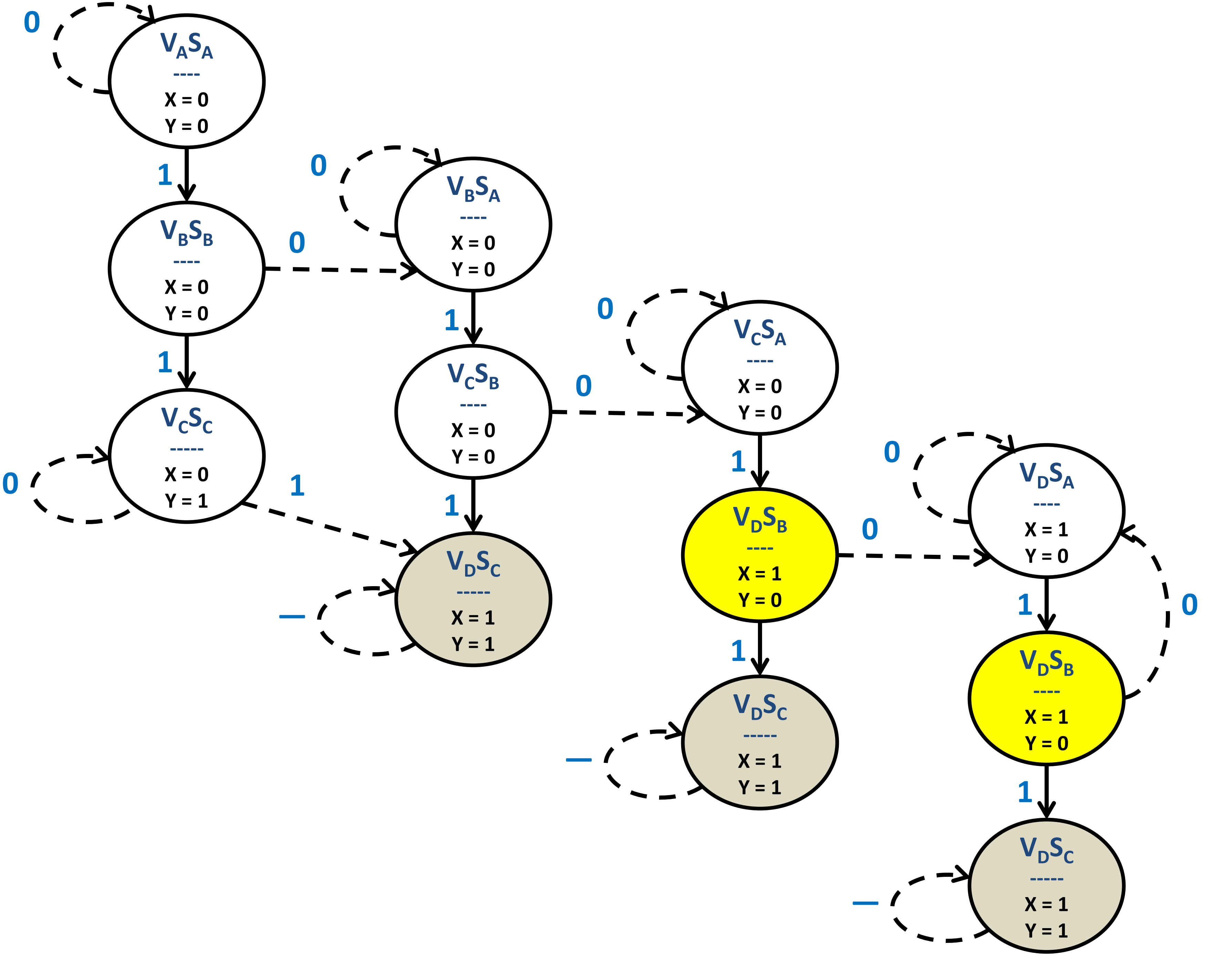
X = 1
Y = 0
1
V_DS_C

X = 1
Y = 1
—
0
0
V_DS_A

X = 1
Y = 0
1
V_DS_B

X = 1
Y = 0
0
1
V_DS_C

X = 1
Y = 1
—


V_AS_A

X = 0
Y = 0
0
1
V_BS_B

X = 0
Y = 0
0
1
V_CS_C

X = 0
Y = 1
0
1
V_BS_A

X = 0
Y = 0
0
1
V_CS_B

X = 0
Y = 0
0
1
V_DS_C

X = 1
Y = 1
—
V_CS_A

X = 0
Y = 0
0
1
V_DS_B

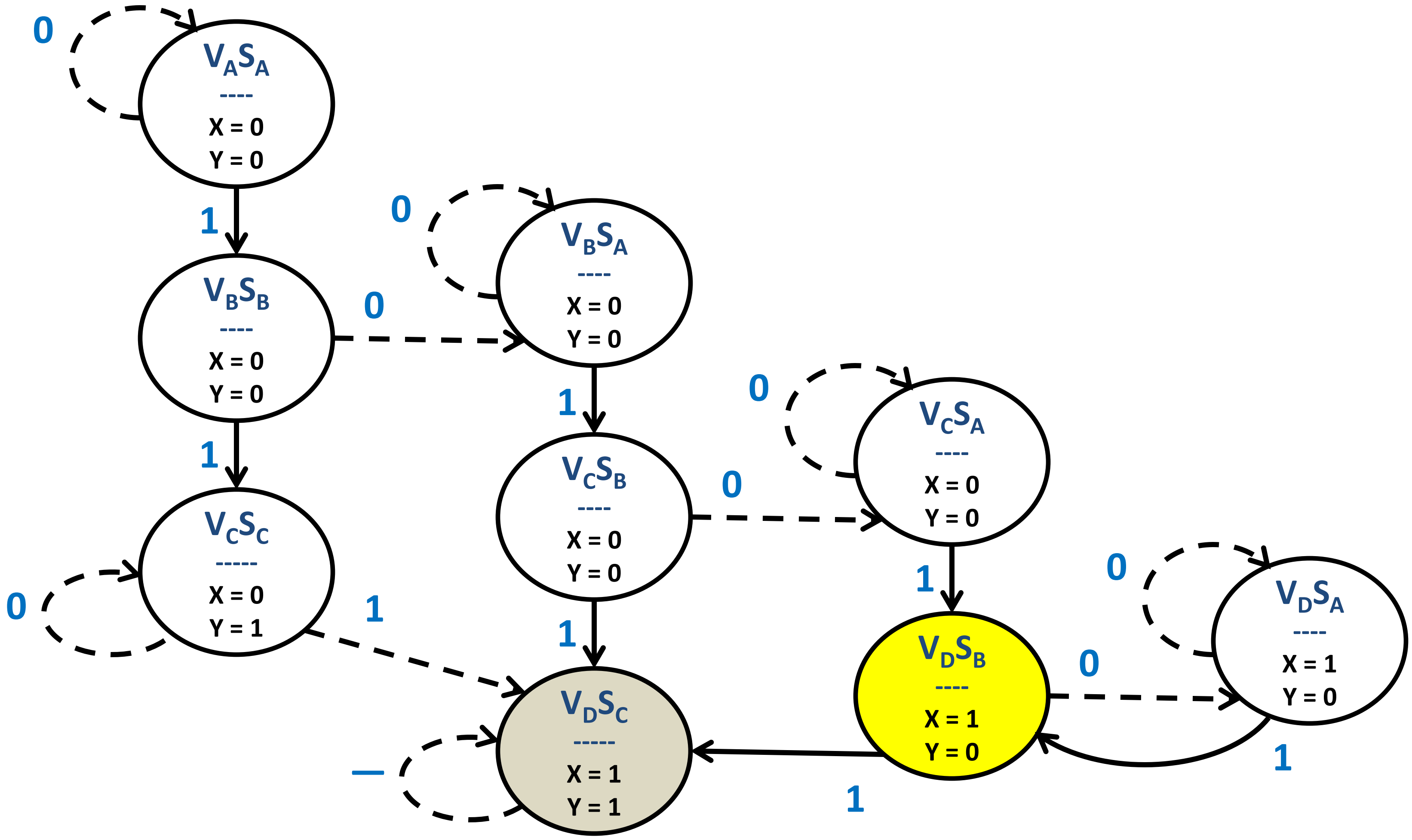
X = 1
Y = 0
0
1
V_DS_C

X = 1
Y = 1
—
V_DS_A

X = 1
Y = 0
0
1
V_DS_B

X = 1
Y = 0
0
1
V_DS_C

X = 1
Y = 1
—



V_AS_A

X = 0
Y = 0
V_BS_B

X = 0
Y = 0
V_CS_C

X = 0
Y = 1
V_BS_A

X = 0
Y = 0
V_CS_B

X = 0
Y = 0
V_DS_C

X = 1
Y = 1
V_CS_A

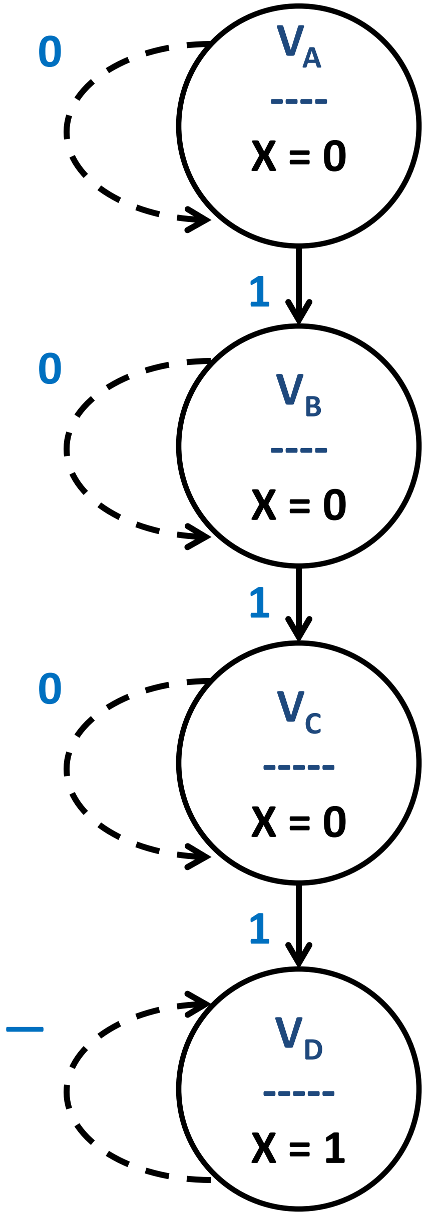
X = 0
Y = 0
V_DS_B

X = 1
Y = 0
V_DS_A

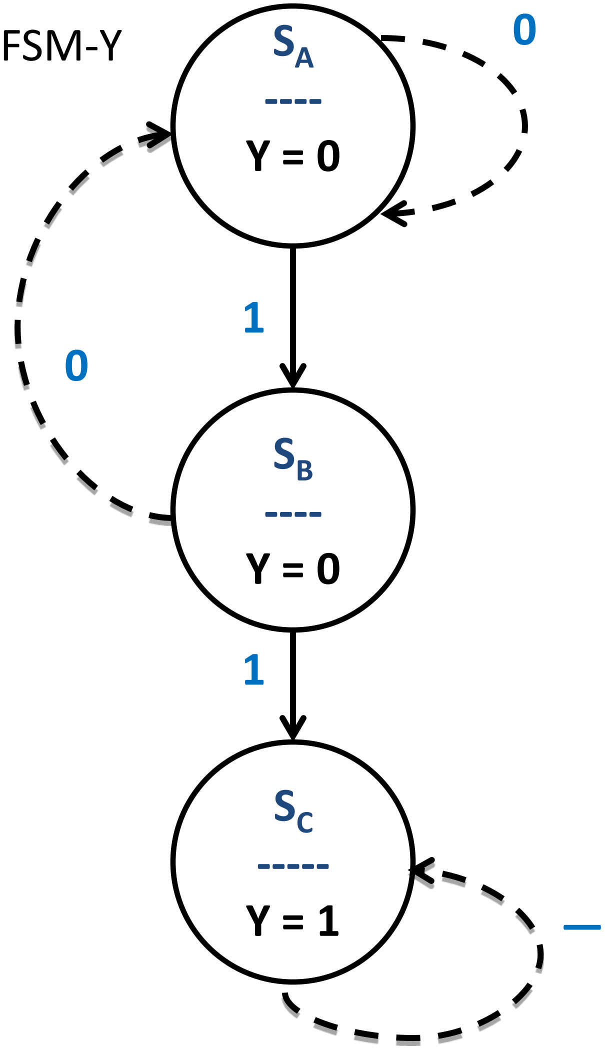
X = 1
Y = 0
0
1
1
0
0
1
1
0
1
0
1
0
1
0
0
1
1
—

- Can also synthesize an FSM for X (FSM-X) and an FSM for Y (FSM-Y) separately (although this is different than what’s asked in this question)

FSM-X

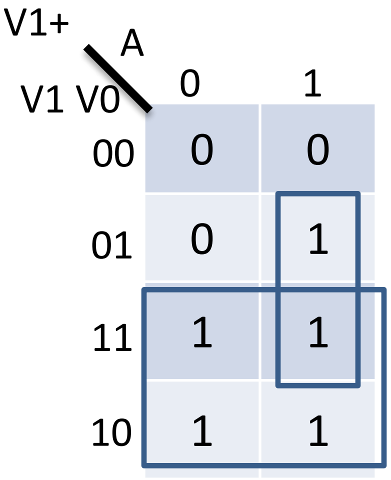


FSM-Y



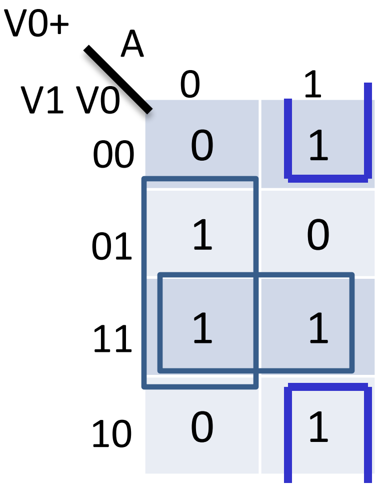
State transition and output table, **K-Maps for X, V1+, V0+** (for FSM-X)

Present State V1 V0	A = 0	A = 1	X
	NS V1+ V0+	NS V1+ V0+	
V_A = 0 0	0 0	0 1	0
V_B = 0 1	0 1	1 0	0
V_C = 1 0	1 0	1 1	0
V_D = 1 1	1 1	1 1	1



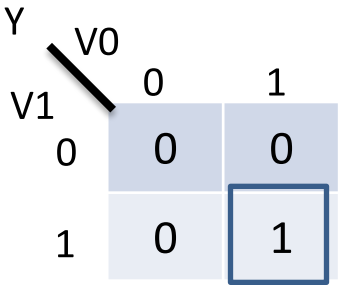
V1 V0 \ A	0	1
00	0	0
01	0	1
11	1	1
10	1	1

V1+ = V1 + V0.A



V1 V0 \ A	0	1
00	0	1
01	1	0
11	1	1
10	0	1

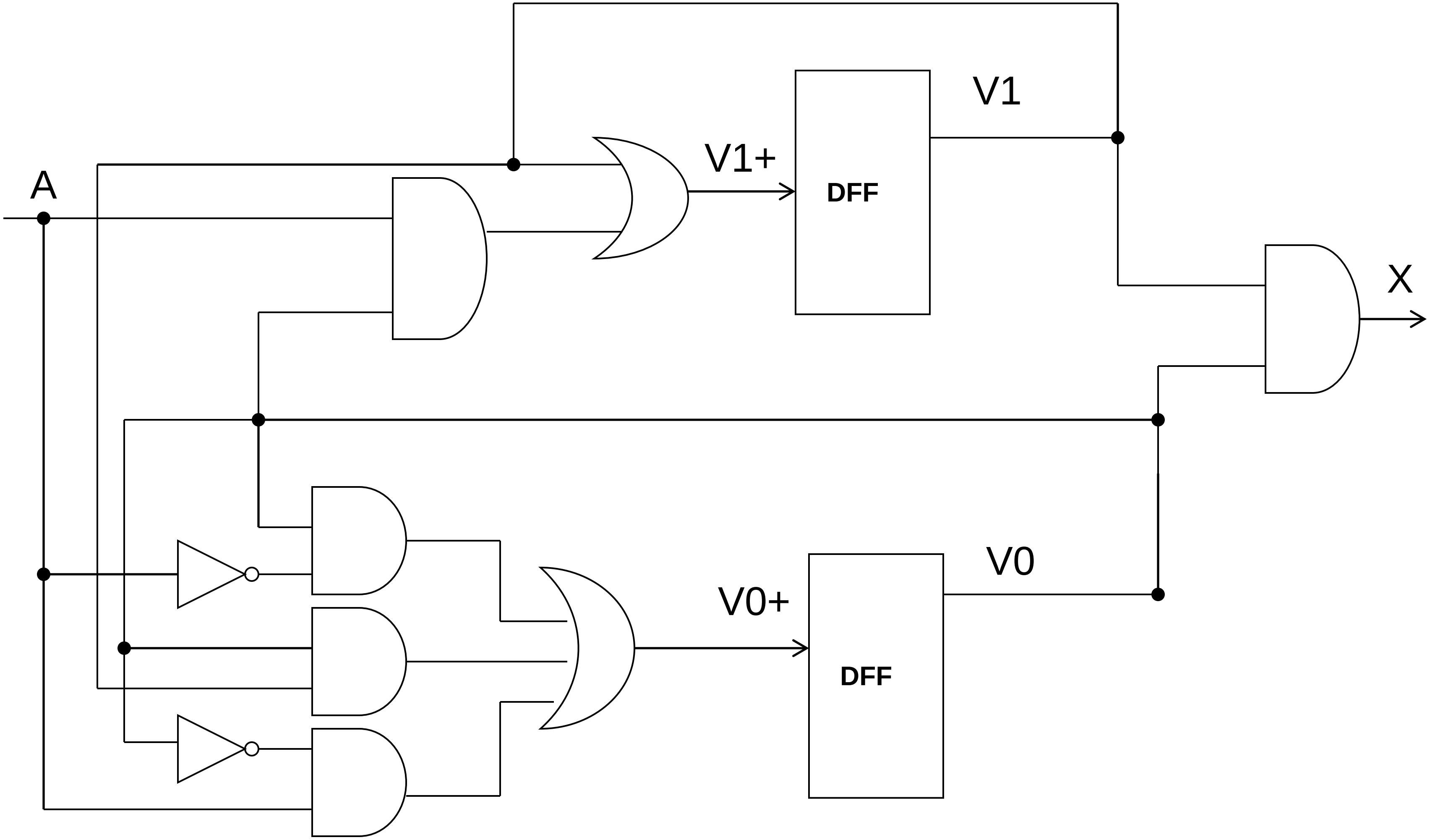
V0+ = V0.A' + V1V0 + V0'.A



V1 \ V0	0	1
0	0	0
1	0	1

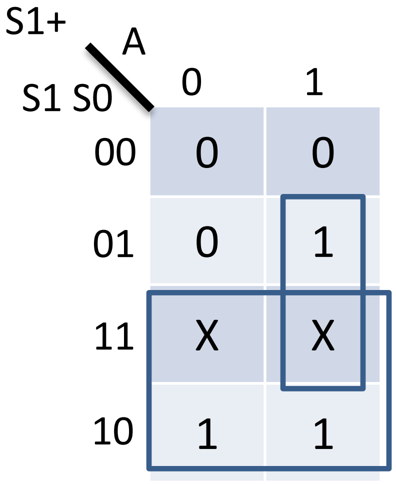
Y = V1.V0

Schematic for X, V1, V0



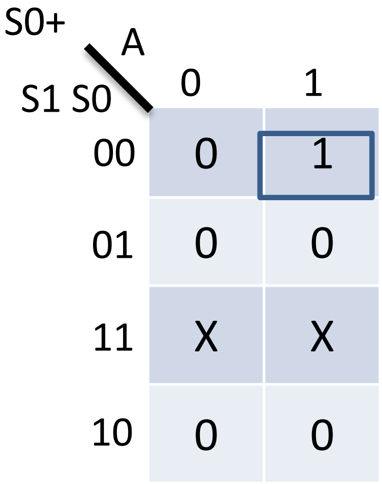
State transition and output table, **K-Maps for Y, S1+, S0+** (for FSM-Y)

Present State S1 S0	A = 0	A = 1	Y
	NS S1+ S0+	NS S1+ S0+	
S_A = 0 0	0 0	0 1	0
S_B = 0 1	0 0	1 0	0
S_C = 1 0	1 0	1 0	1



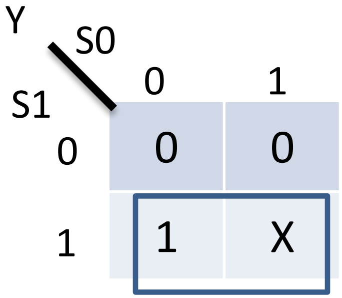
S1 S0	0	1
00	0	0
01	0	1
11	X	X
10	1	1

S1+ = S1 + S0.A



S1 S0	0	1
00	0	1
01	0	0
11	X	X
10	0	0

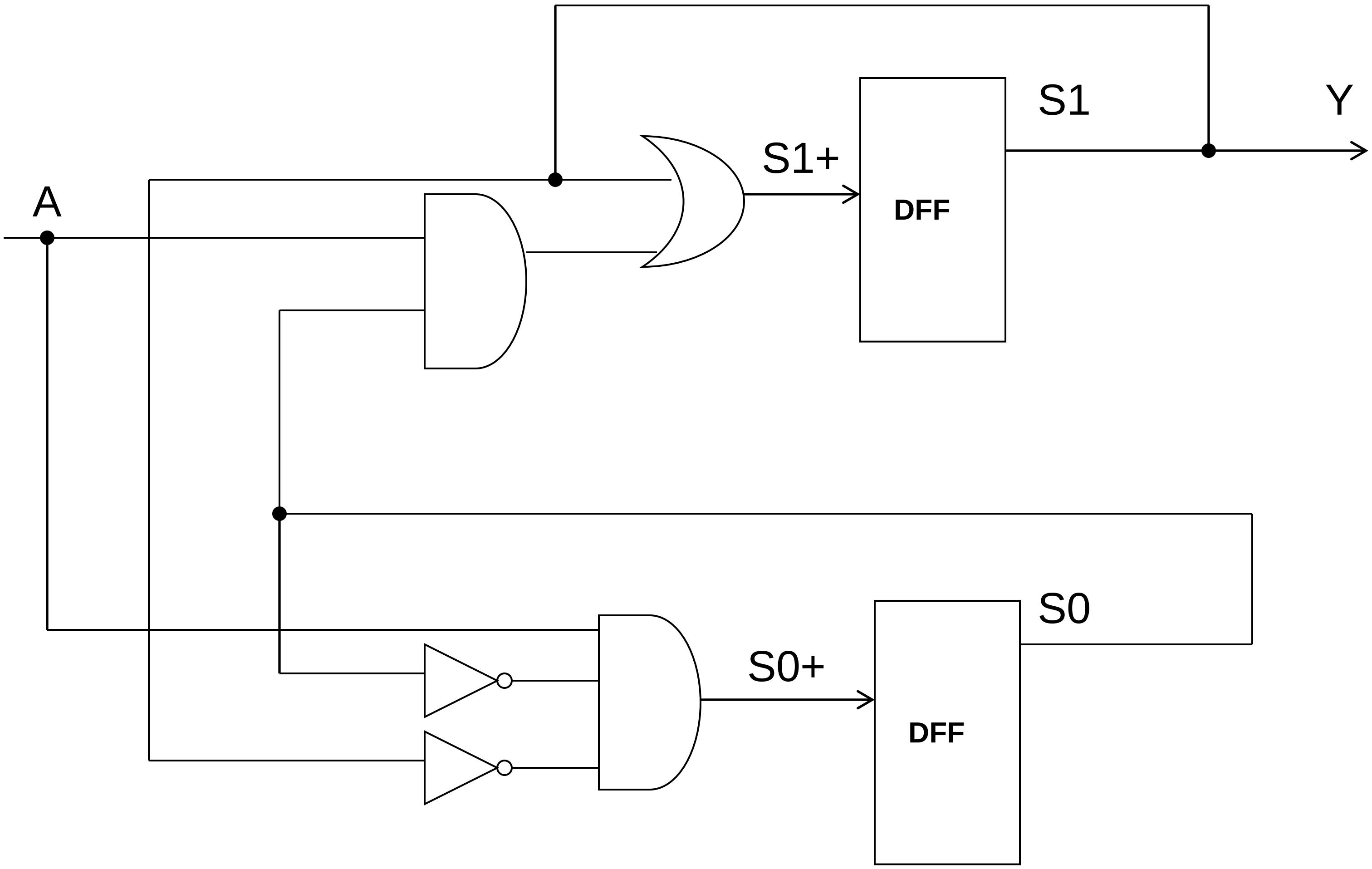
S0+ = A.S0’.S1’



S1	0	1
0	0	0
1	1	X

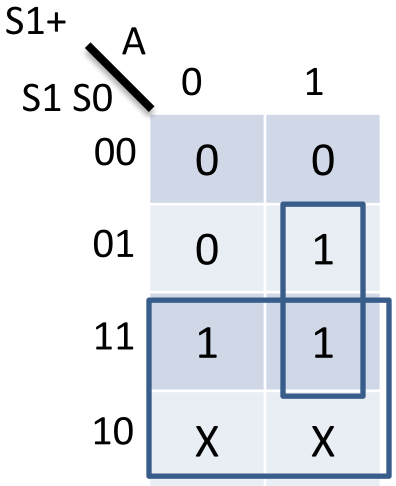
Y = S1

Schematic for Y, S1, S0



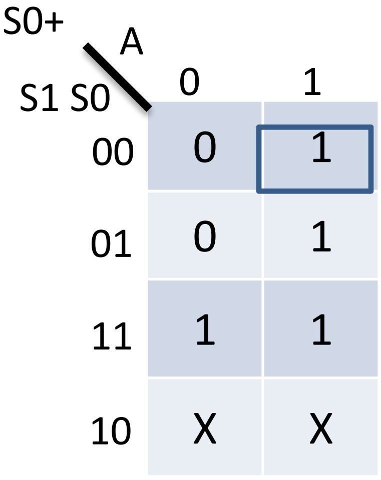
State transition and output table, **K-Maps for Y, S1+, S0+** (for FSM-Y) :
Using a different State Assignment for Sc. (Using 11 instead of 10)
This helps in reducing the number of literals required to compute S0+. (requires 2 literals instead of 3)
Efficient State Assignment problem is sometimes taken care of by EDA tools.

Present State S1 S0	**A = 0**	**A = 1**	**Y**
	NS **S1+ S0+**	**NS** **S1+ S0+**	
S_A = 0 0	**0 0**	**0 1**	**0**
S_B = 0 1	**0 0**	**1 1**	**0**
S_C = 1 1	**1 1**	**1 1**	**1**



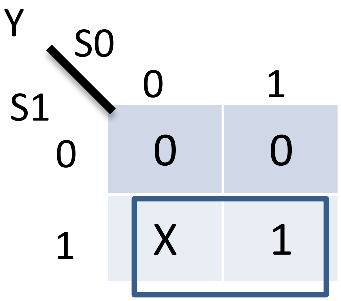
S1 S0 \ A	0	1
00	0	0
01	0	1
11	1	1
10	X	X

S1+ = S1 + S0.A



S1 S0 \ A	0	1
00	0	1
01	0	1
11	1	1
10	X	X

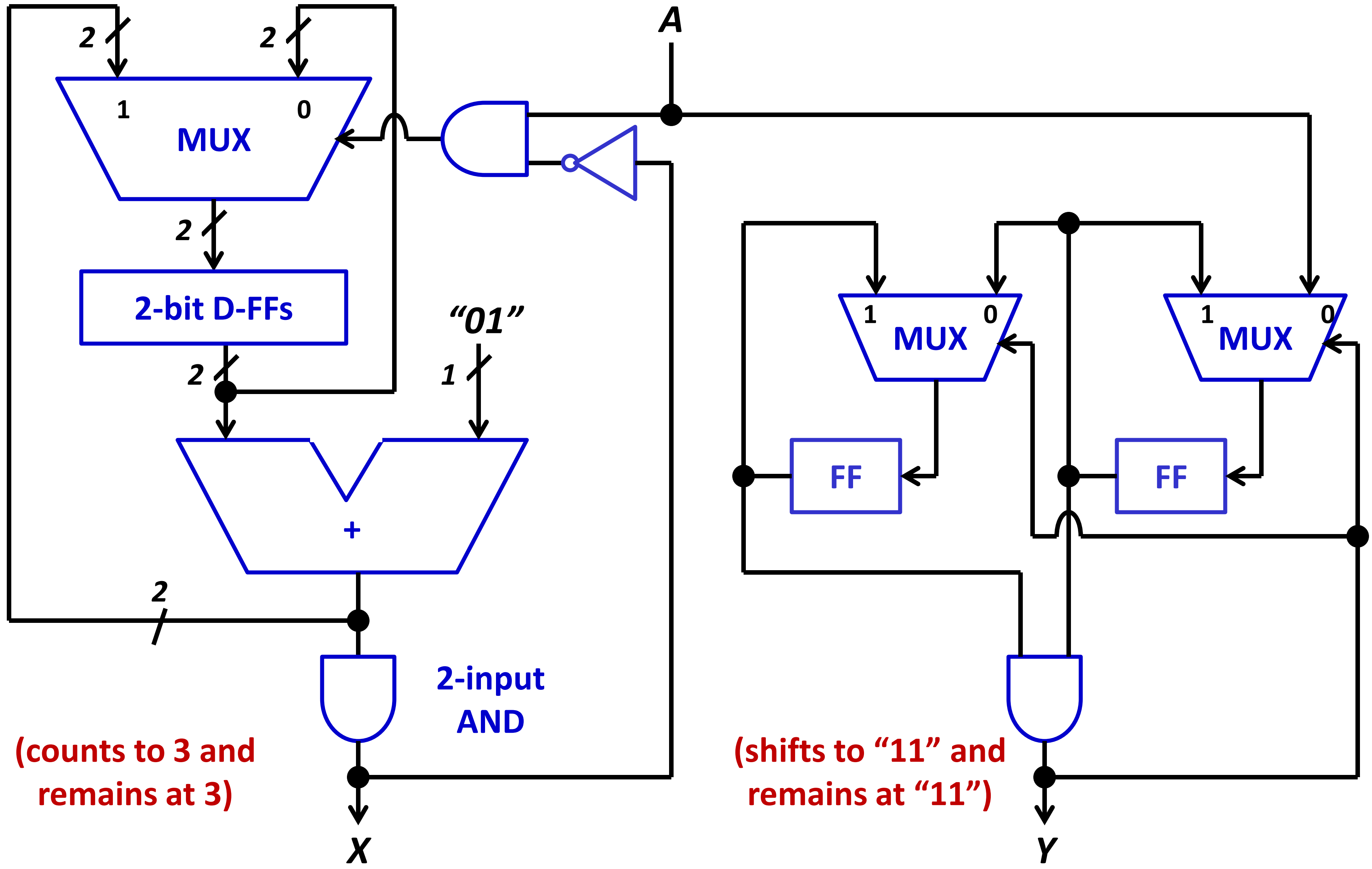
S0+ = S1 + A



S1 \ S0	0	1
0	0	0
1	X	1

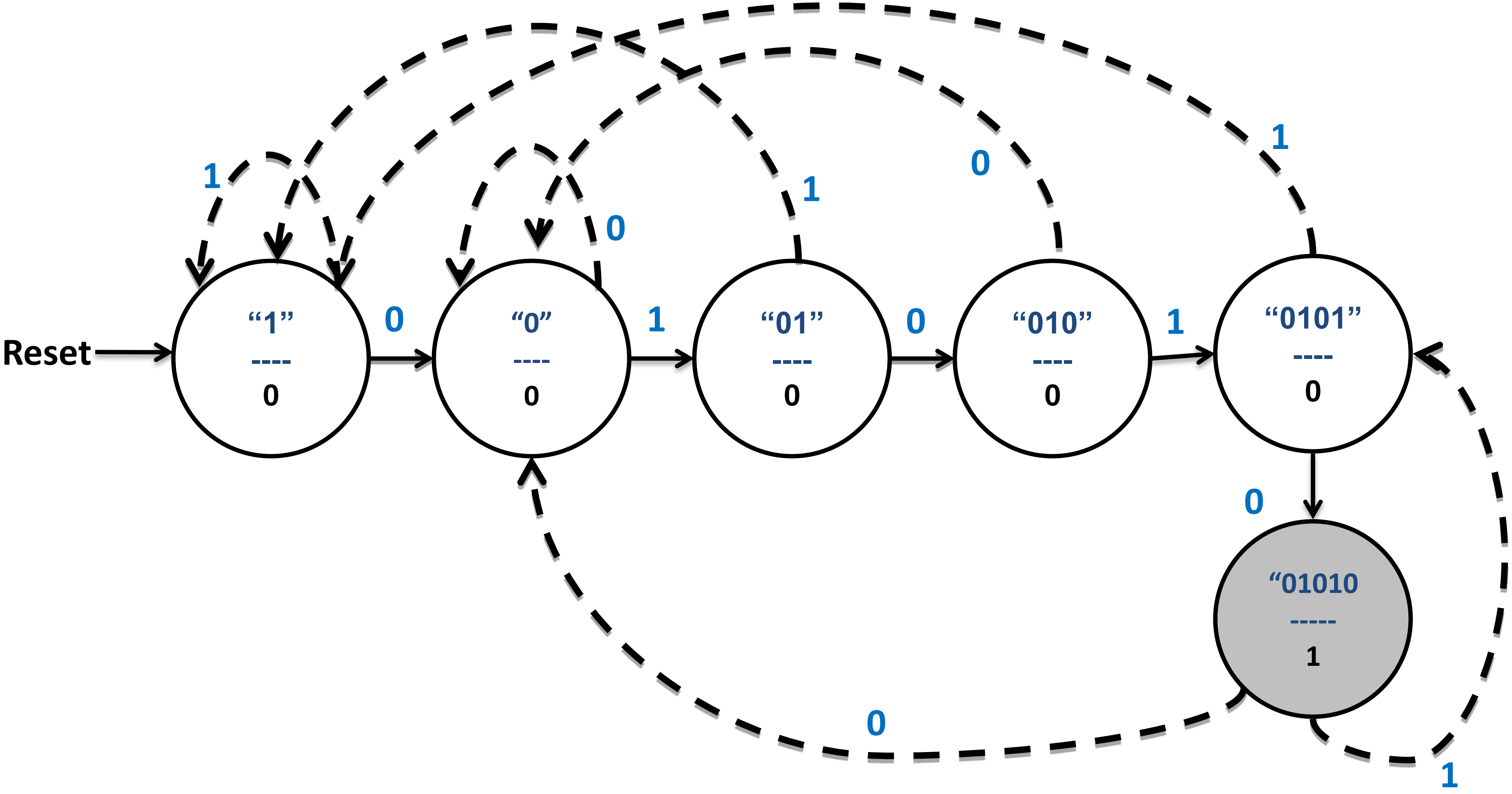
Y = S1

- Can also directly implement using datapath (e.g. counters and shift registers) – all FF's initialize to "0"


A
2
2
1
0
MUX
2
2-bit D-FFs
2
"01"
1
+
2
2-input AND
(counts to 3 and remains at 3)
X
1
0
MUX
1
0
MUX
FF
FF
(shifts to "11" and remains at "11")
Y

Interview Q 3.1

Design an FSM that recognizes 01010 when it is received serially.


1
1
0
0
1
Reset
"1"

0
0
"0"

0
1
"01"

0
0
"010"

0
1
"0101"

0
0
"01010

1
0
1