Digital Design Hwk 4

Wednesday, October 2, 2019 4:30 PM

3.11 Trace the behavior of a D latch (see Figure 3.19) for the input pattern in Figure 3.98. Assume Q is initially 0. Complete the timing diagram, assuming logic gates have a tiny but nonzero delay.

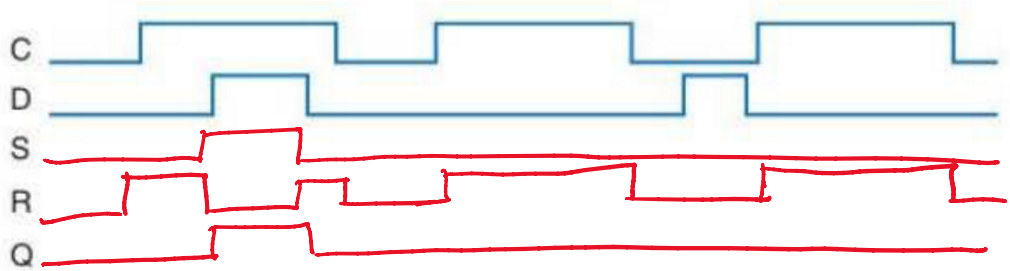


Figure 3.98 D latch input pattern timing diagram.

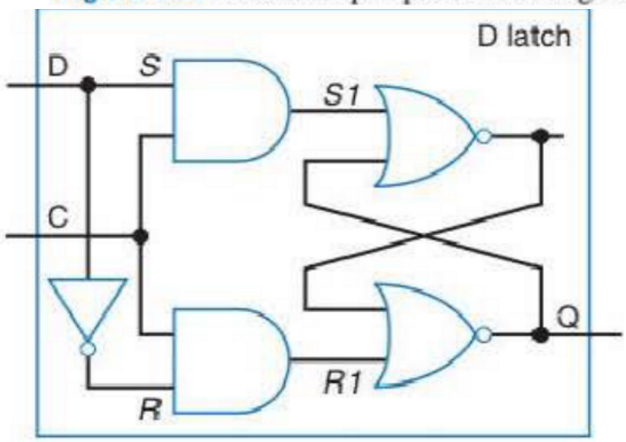


Figure 3.19 D latch internal circuit

3.13 Trace the behavior of an edge-triggered D flip-flop using the master-servant design (see Figure 3.25) for the input pattern in Figure 3.100. Assume each internal latch initially stores a 0. Complete the timing diagram, assuming logic gates have a tiny but nonzero delay.

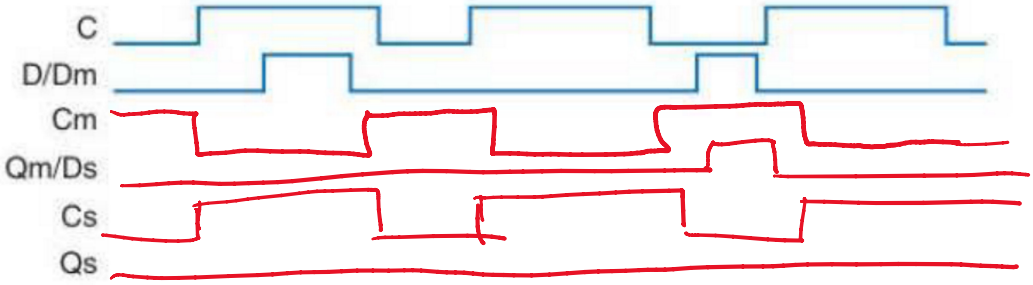


Figure 3.100 Edge-triggered D flip-flop input pattern timing diagram.

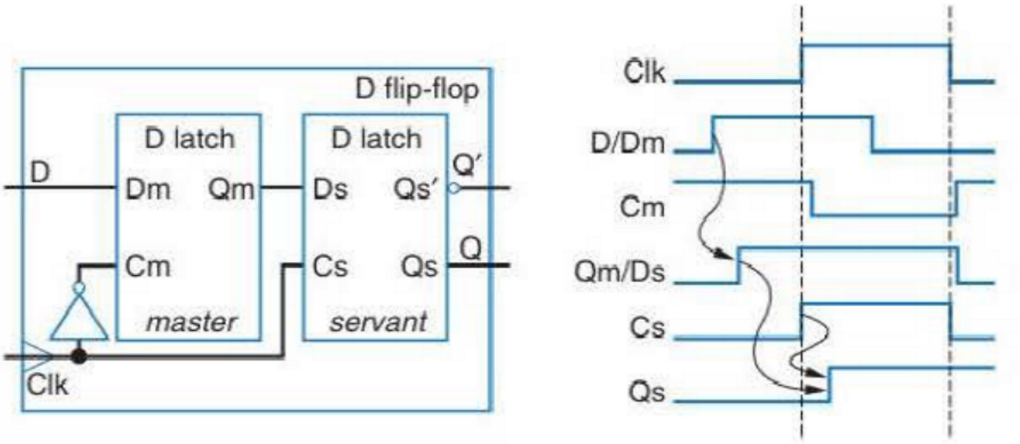


Figure 3.25 A D flip-flop implementing an edge-triggered bit storage block, internally using two D latches in a master-servant arrangement. The master D latch stores its `Dm` input while `Clk = 0`, but the new value appearing at `Qm`, and hence at `Ds`, does not get stored into the servant latch, because the servant latch is disabled when `Clk = 0`. When `Clk` becomes `1`, the servant D latch becomes enabled and thus gets loaded with whatever value was in the master latch at the instant that `Clk` changed from `0` to `1`.

3.15 Compare the behavior of D latch and D flip-flop devices by completing the timing diagram in Figure 3.102. Assume each device initially stores a `0`. Provide a brief explanation of the behavior of each device.

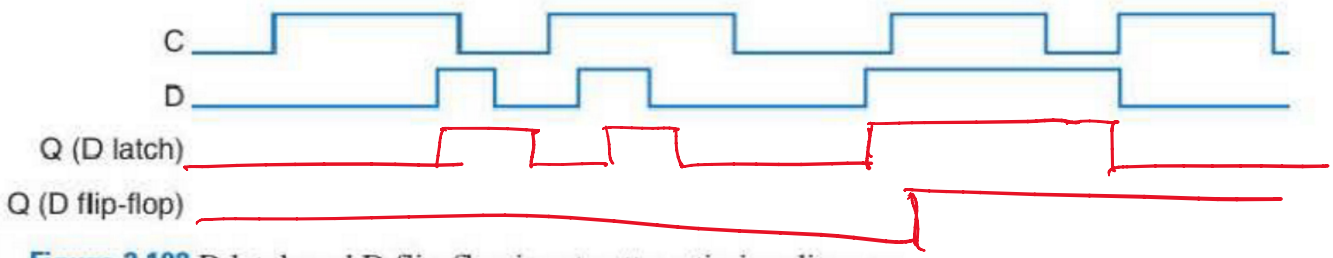
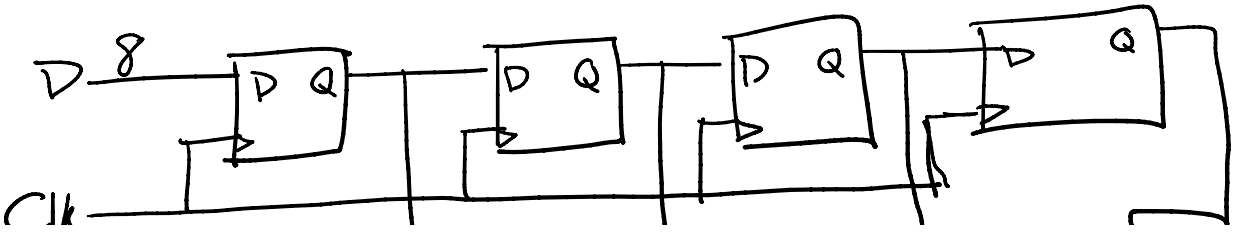
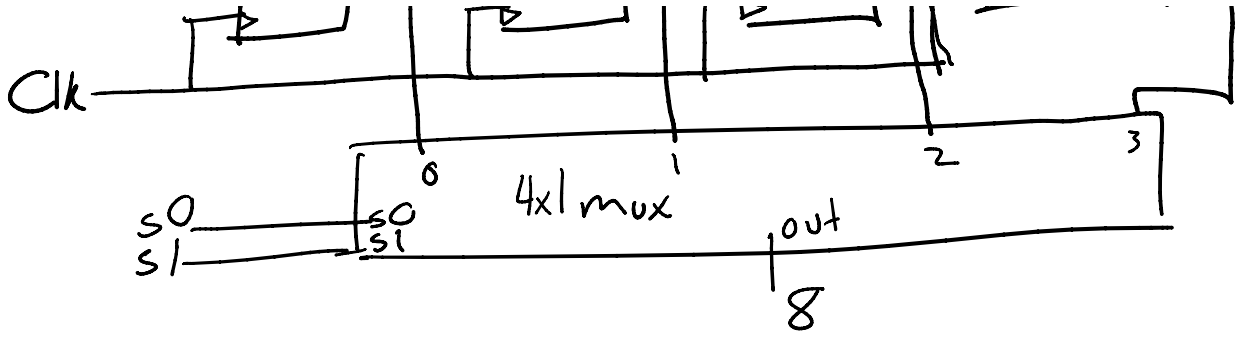


Figure 3.102 D latch and D flip-flop input pattern timing diagram.

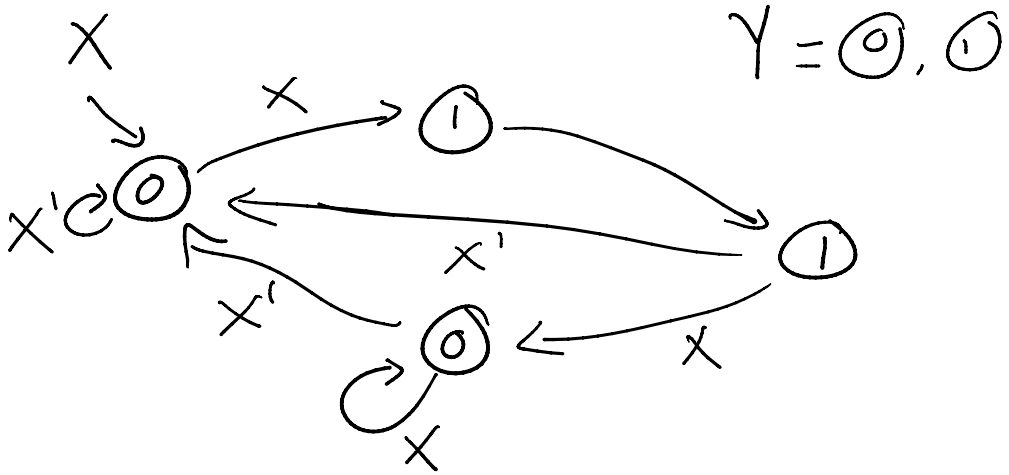
D is a rising-edge detector of C.

3.19 Using four registers, design a circuit that stores the four values present at an 8-bit input `D` during the previous four clock cycles. The circuit should have a single 8-bit output that can be configured using two inputs `s1` and `s0` to output any one of the four registers. (Hint: use an 8-bit 4x1 mux.)

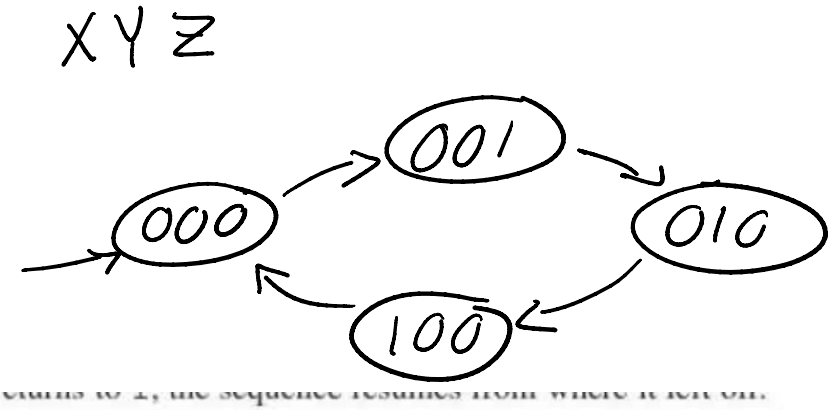




3.24 Draw a state diagram for an FSM that has an input X and an output Y. Whenever X changes from 0 to 1, Y should become 1 for two clock cycles and then return to 0—even if X is still 1. (Assume for this problem and all other FSM problems that an implicit rising clock is ANDed with every FSM transition condition.)

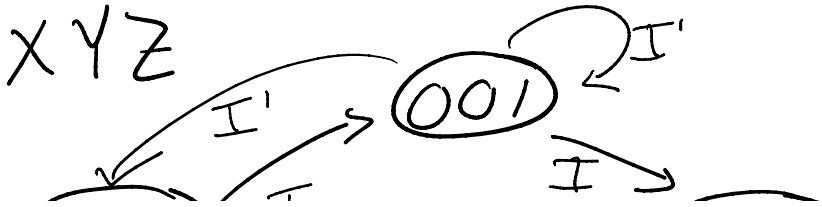


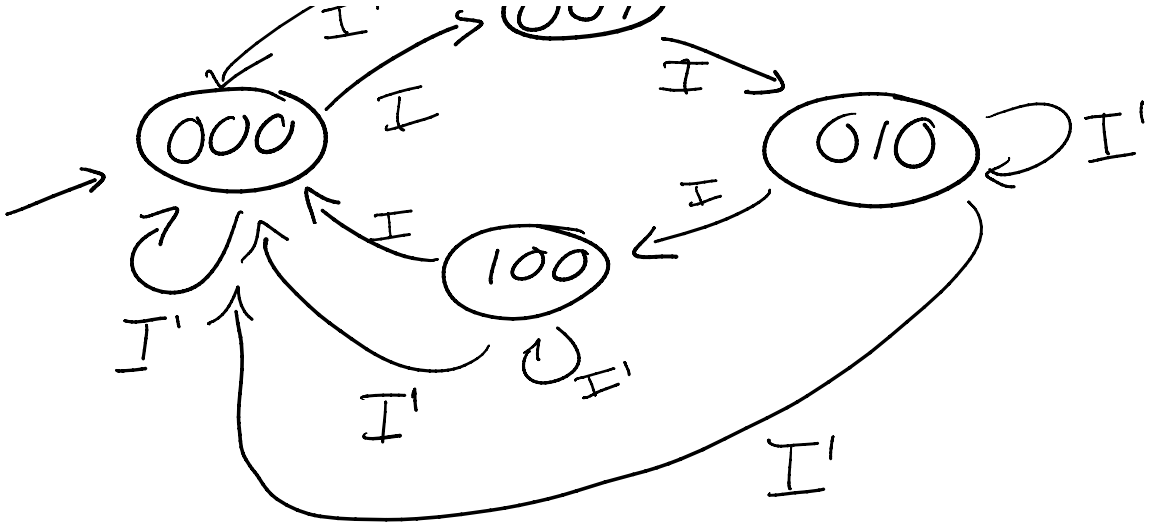
3.25 Draw a state diagram for an FSM with no inputs and three outputs x, y, and z. xyz should always exhibit the following sequence: 000, 001, 010, 100, repeat. The output should change only on a rising clock edge. Make 000 the initial state.



returns to 1, the sequence resumes from where it left off.

3.27 Do Exercise 3.25, but add an input I that can stop the sequence when set to 0. When I returns to 1, the sequence starts from 000 again.

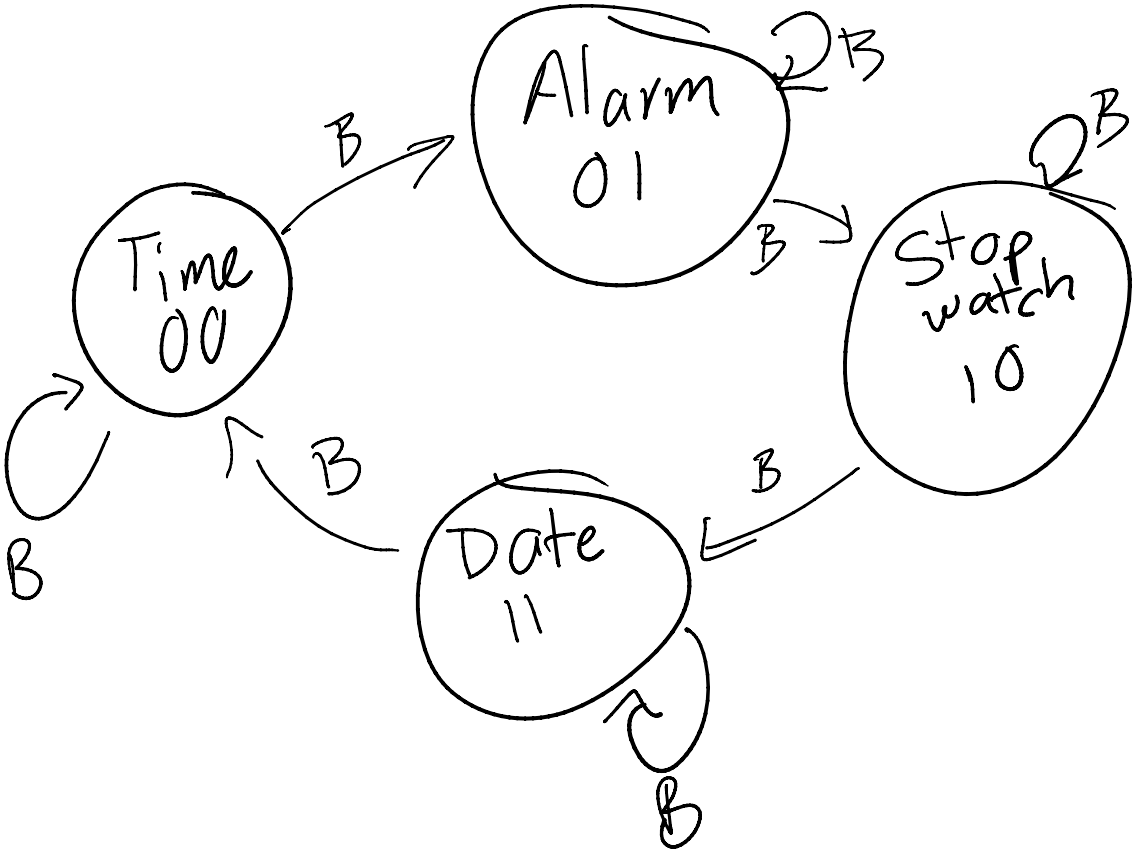




returns to 1, the sequence starts from 000 again.

3.28 A wristwatch display can show one of four items: the time, the alarm, the stopwatch, or the date, controlled by two signals `s1` and `s0` (`00` displays the time, `01` the alarm, `10` the stopwatch, and `11` the date—assume `s1` and `s0` control an *N*-bit mux that passes through the appropriate register). Pressing a button `B` (which sets `B = 1`) sequences the display to the next item. For example, if the presently displayed item is the date, the next item is the current time. Create a state diagram for an FSM describing this sequencing behavior, having an input bit `B`, and two output bits `s1` and `s0`. Be sure to only sequence forward by one item each time the button is pressed, regardless of how long the button is pressed—in other words, be sure to wait for the button to be released after sequencing forward one item. Use short but descriptive names for each state. Make displaying the time be the initial state.

3.29 Extend the state diagram created in Exercise 3.28 by adding an input R. R=1 forces the FSM



3.33 For FSMs with the following numbers of states, indicate the smallest possible number of bits for a state register representing those states:

(a) 4
(b) 8
(c) 9
(d) 23
(e) 900

(a) 2 bits
(b) 3 bits
(c) 4 bits
(d) 5 bits
(e) 10 bits

3.34 How many possible states can be represented by a 16-bit register?

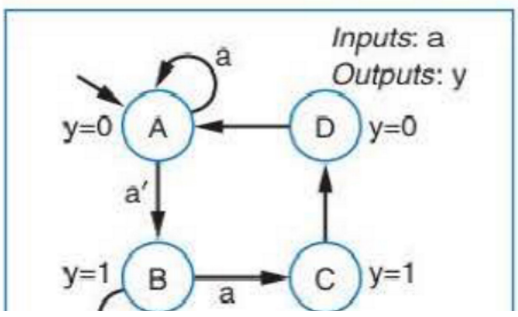
3.35 If an FSM has N states, what is the maximum number of possible transitions that could exist in the FSM? Assume that no pair of states has more than one transition in the same direction, and that no state has a transition point back to itself. Assuming there are a large number of inputs, meaning the number of transitions is not limited by the number of inputs? Hint: try for small N, and then generalize.

A, B
A → B
B → A

A, B, C
A → B
A → C
B → A
B → C
C → A
C → B

$N \rightarrow N-1$

$N(N-1)$



3.39 Using the process for designing a controller, convert

3.39 Using the process for designing a controller, convert the FSM of Figure 3.109 to a controller, implementing the controller using a state register and logic gates.

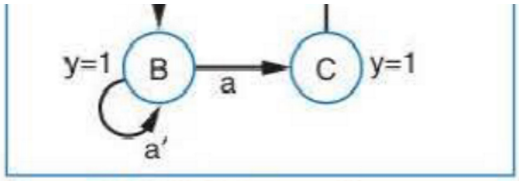
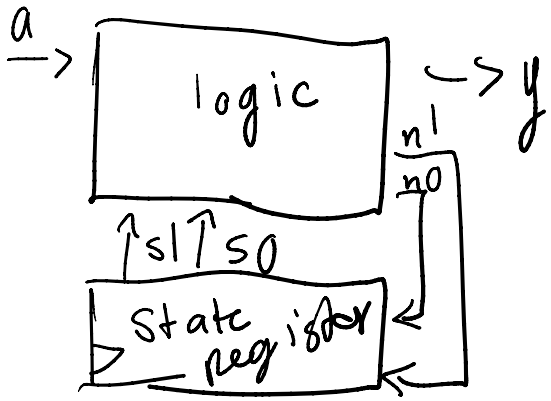


Figure 3.109 FSM example.



A = 00
B = 01
C = 10
D = 11

s1	s0	a	n1	n0	y
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	0

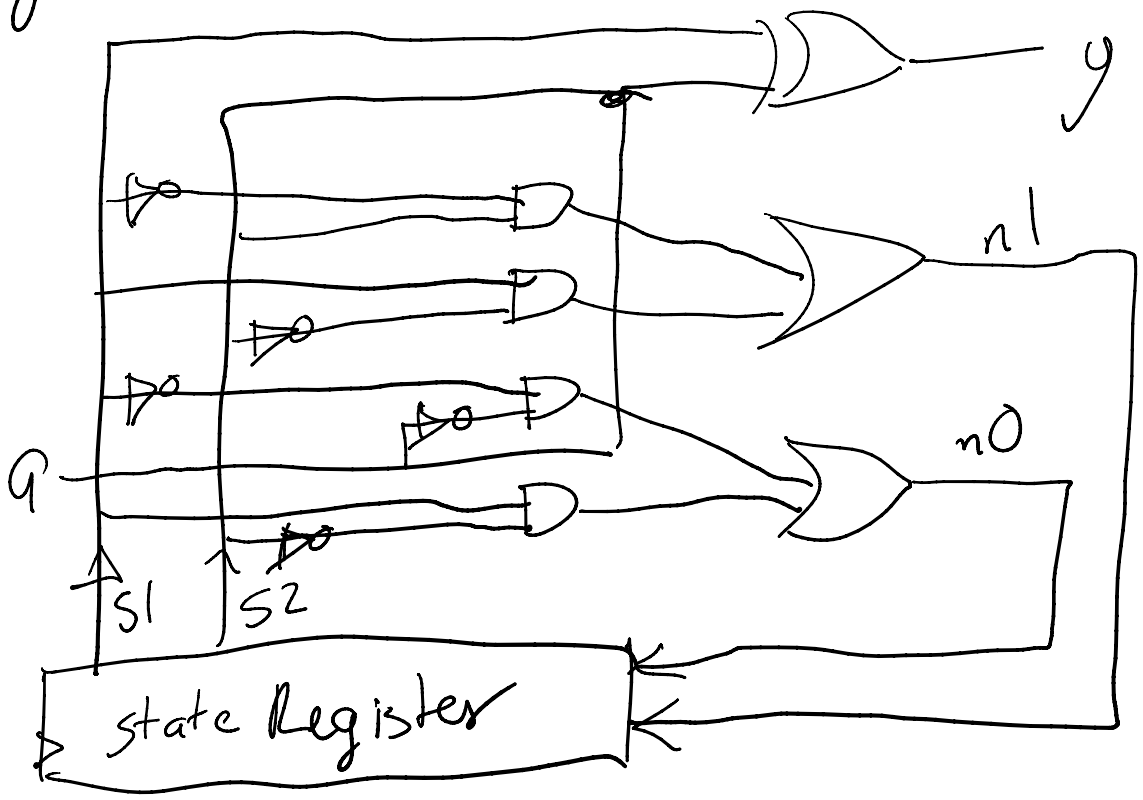
$n1 = s1's0a + s1s0'a' + s1s0'a$

$n0 = s1's0'a' + s1's0a' + s1s0'a' + s1s0'a$

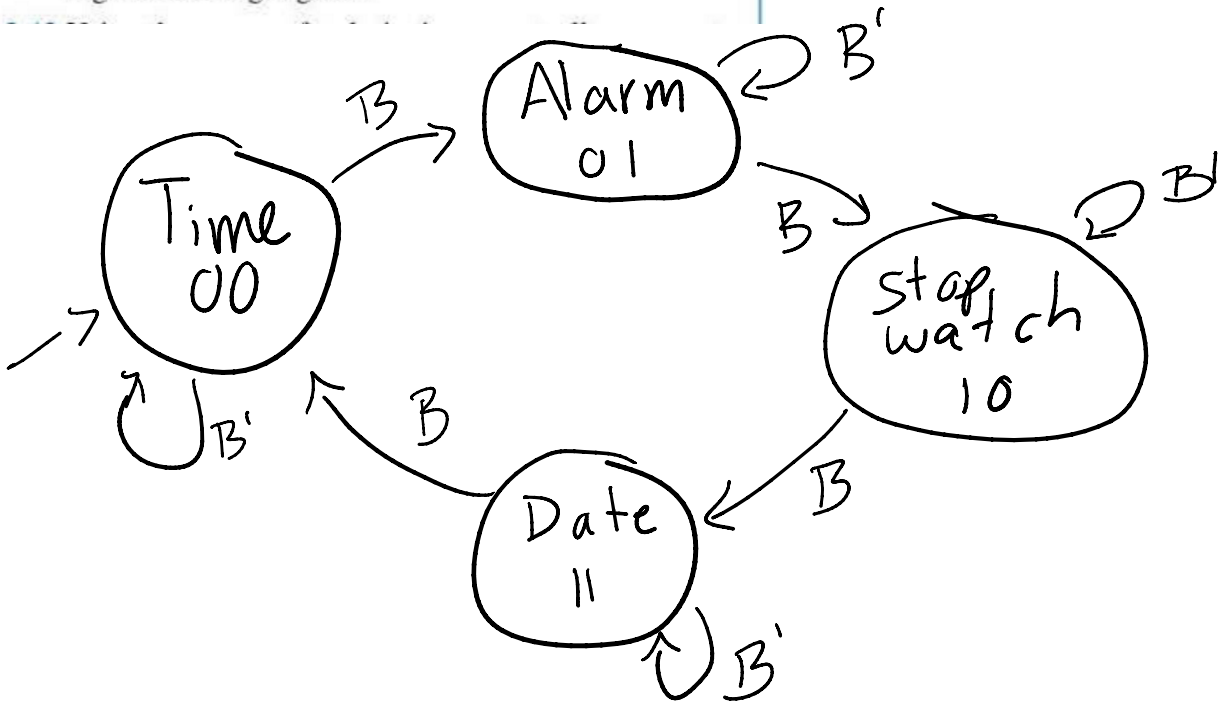
$y = s1's0a + s1s0'a' + s1s0'a$

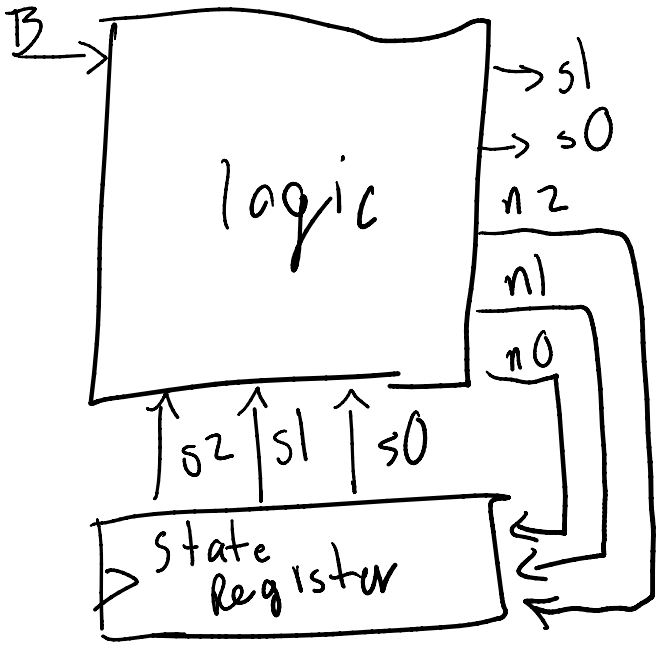
n0 = ...

$y = s1's0a' + s1's0a + s1s0'a' + s1s0'a$



3.42 Using the process for designing a controller, convert the FSM you created for Exercise 3.28 to a controller, implementing the controller using a state register and logic gates.





s2	s1	s0	B	n2	n1	n0	s1	s0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1
0	0	1	1	0	0	1	0	1
0	1	0	0	0	1	0	0	1
0	1	0	1	0	1	1	0	1
0	1	1	0	1	0	0	1	0
0	1	1	1	0	1	1	1	0
1	0	0	0	1	0	0	1	0
1	0	0	1	1	0	1	1	0
1	0	1	0	1	0	1	1	1
1	0	1	1	1	1	0	1	1
1	1	0	0	1	1	0	1	1

1	1	0	0	1	1	0	1	1
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0

$n2 = s2's1s0B' + s2s1' + s2s0' + s2B$

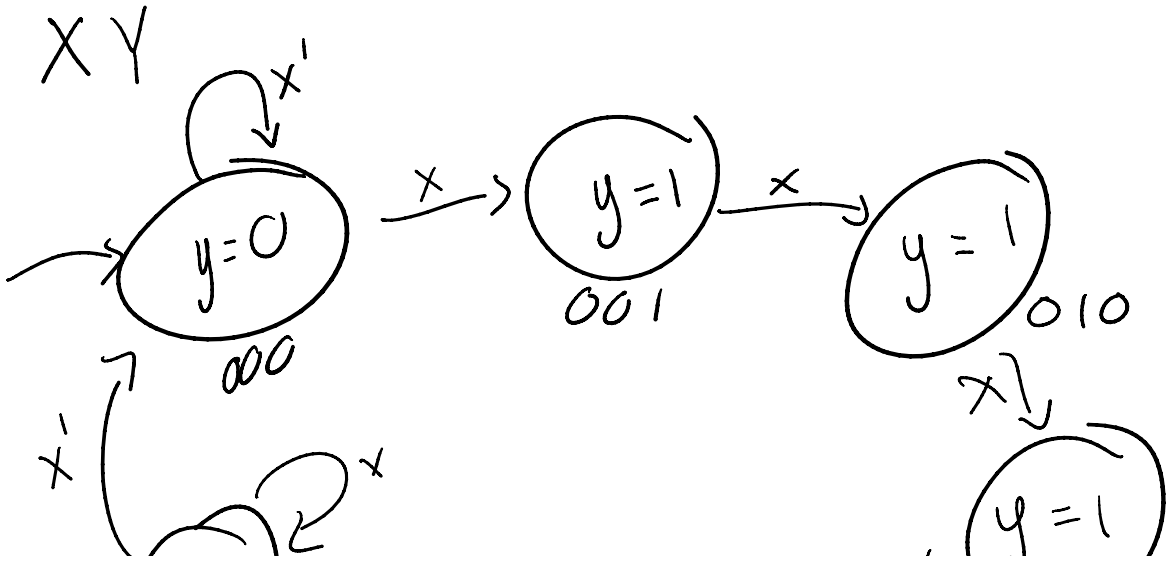
$n1 = s1s0' + s1B + s2s0B + s2's1's0B'$

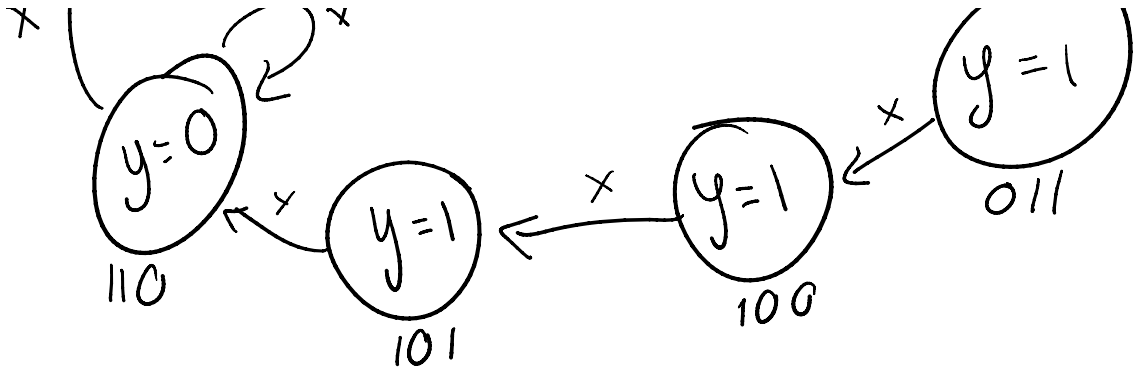
$n0 = s0'B + s2'B + s1B + s2s1's0B'$

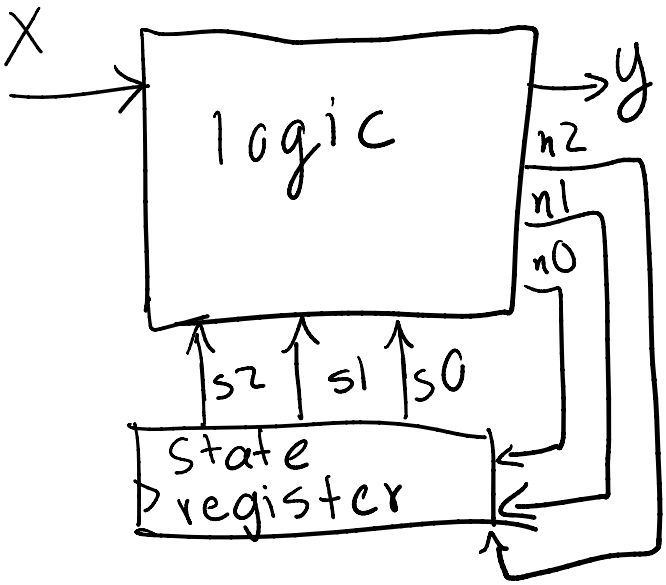
$s1 = s2s0' + s2s1' + s2's1s0$

$s0 = s1 \text{ XOR } s0$

3.45 Create an FSM that has an input X and an output Y. Whenever X changes from 0 to 1, Y should become 1 for five clock cycles and then return to 0—even if X is still 1. Using the process for designing a controller, convert the FSM to a controller, stopping once you have created the truth table.







s2	s1	s0	X	n2	n1	n0	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	0	1
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	1
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	1

0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	1
1	0	0	1	1	0	1	1
1	0	1	0	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	0	1	1	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

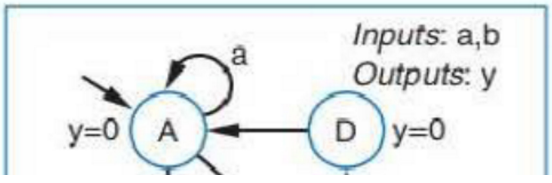
$$n2 = s2s1' + s2's1s0 + s2s0'x'$$

$$n1 = s1's0 + s2's1s0' + s1s0'x'$$

$$n0 = s2s1's0' + s2's1s0' + s2's0'x$$

$$Y = (s2 \text{ xor } s1) + s2's0$$

3.46 The FSM in Figure 3.112 has two problems: one state has non-exclusive transitions, and another state has incomplete transitions. By ORing and ANDing the conditions for each state's transitions, prove that



has incomplete transitions. By ORing and ANDing the conditions for each state's transitions, prove that these problems exist. Then fix these problems by refining the FSM, taking your best guess as to what was the FSM creator's intent.

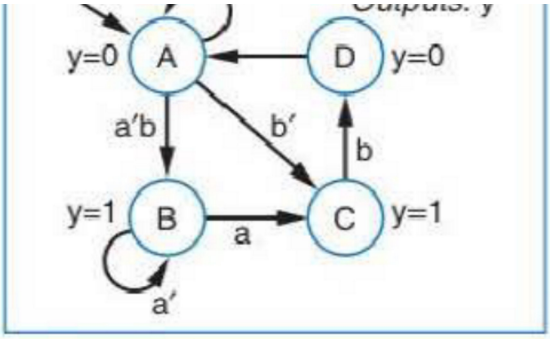


Figure 3.112 FSM example.

AND

$a \cdot ab' = 0 \cdot b = 0$

$a'b \cdot b' = a' \cdot 0 = 0$

ab' is not 0

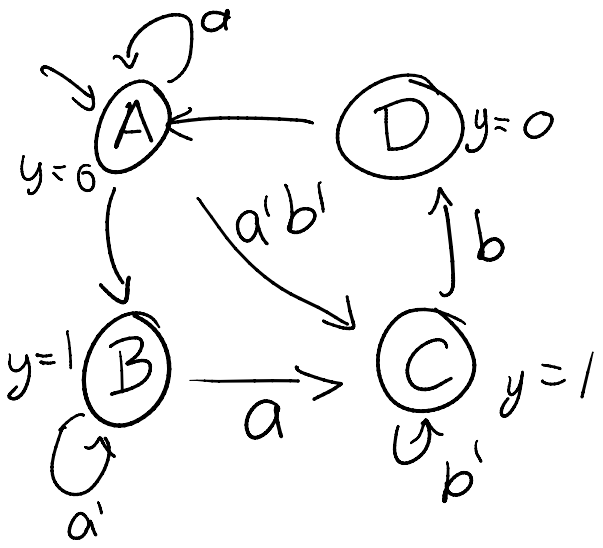
Guess; stay on C when b=0 and go to C on a'b

OR (B)

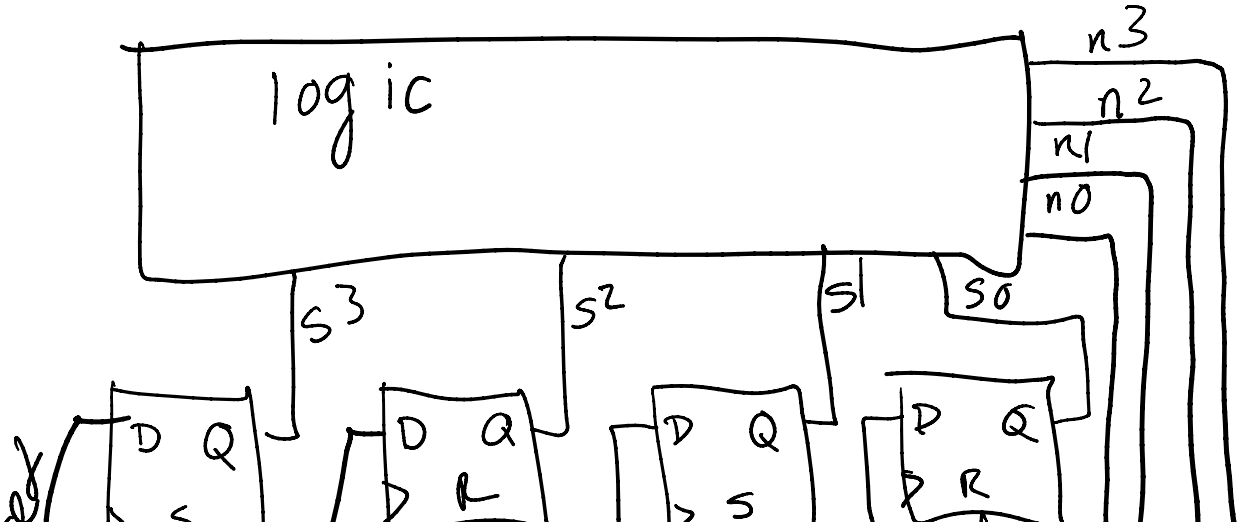
$a + a' = 1$

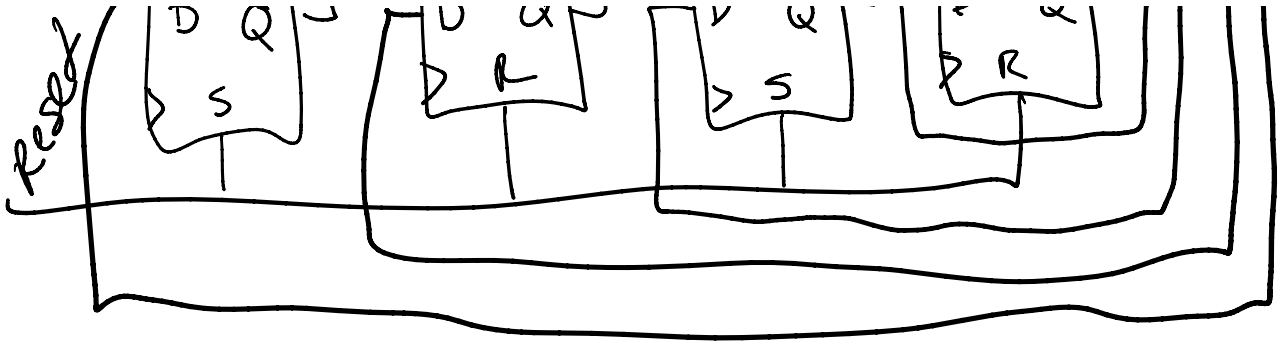
OR (C)

b=?



3.50 Design a controller with a 4-bit state register that gets synchronously initialized to state 1010 when an input *reset* is set to 1.





6.16 Reduce the number of states for the FSM in Figure 6.89 using the partitioning method.

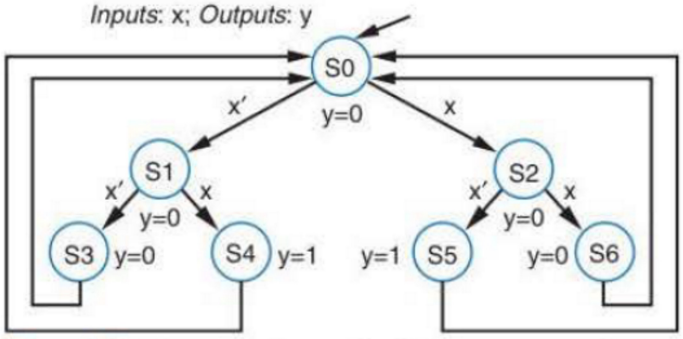


Figure 6.89 Sequence detector for bit patterns 01 and 10.

G1 S0 S1 S2 S3 S6 | (Not in same group)

x=0 G1 G1 G2 G1 G1

G1 S0 S1 S3 S6 | G2 S4 S5 | G3 S2

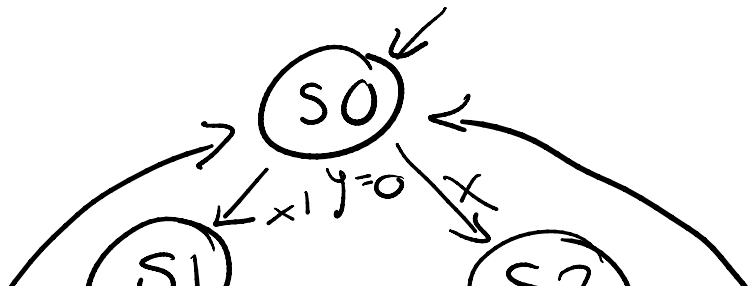
x=0 G1 G1 G1 G1 G1 G1 ✓

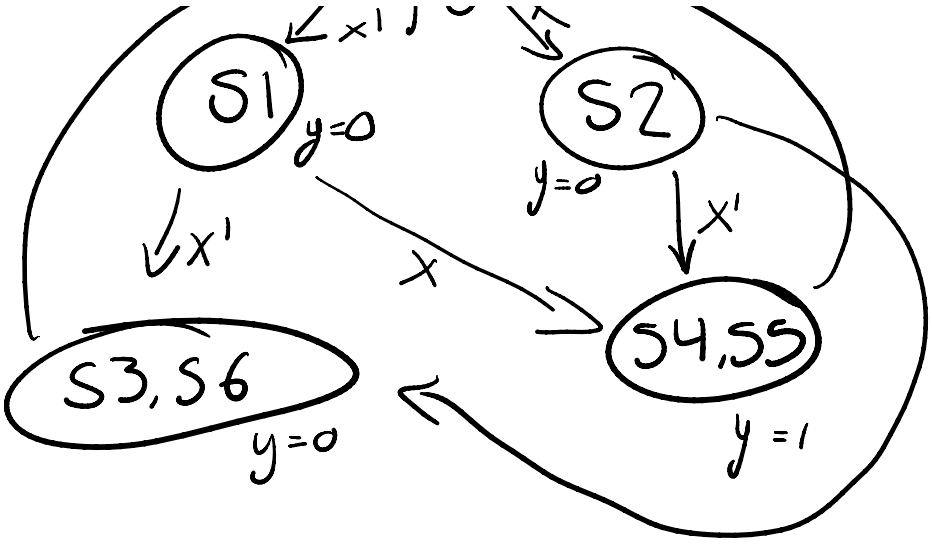
x=1 G3 G2 G1 G1 (Not in same group

G1 S0 | G2 S4 S5 | G3 S2 | G4 S1 | G5 S3 S6

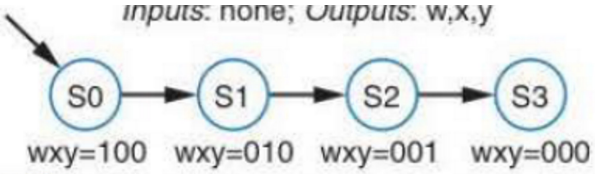
x=0 ✓ G1 G1 ✓ ✓ G1 G1

x=1 ✓ G1 G1 ✓ ✓ G1 G1





6.18 Compare the logic size (number of gate inputs) and the delay (number of gate-delays) of a straightforward 2-bit binary encoding of the FSM in Figure 6.91 using a 3-bit output encoding versus using a one-hot encoding.



ure 6.91 FSM example.

2-bit binary encoding

encodings

S0: 00
S1: 01
S2: 10
S3: 11

10 inputs
2 delays

n1 = S1 + S2

n0 = S1'S0' + S1

w = S1'S0'

x = S1'S0

y = S1S0'

S1	S2	n1	n0	w	x	y
0	0	0	1	1	0	0
0	1	1	0	0	1	0
1	0	1	1	0	0	1
1	1	1	1	0	0	0

0 inputs

1 1 0 0 0

0 inputs
0 delays

3-bit output encoding

encodings

s0: 100
s1: 010
s2: 001
s3: 000

n2 = 0
n1 = s2
n0 = s1
w = s2
x = s1
y = s0

s2	s1	s0	n2	n1	n0	w	x	y
1	0	0	0	1	0	1	0	0
0	1	0	0	0	1	0	1	0
0	0	1	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0

One-hot encoding

encodings

s0: 0001
s1: 0010
s2: 0100
s3: 1000

s3	s2	s1	s0	n3	n2	n1	n0	w	x	y
0	0	0	1	0	0	1	0	1	0	0
0	0	1	0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0

$n3 = s3 + s2$

$n2 = s1$

$n1 = s0$

$n0 = 0$

$w = s0$

$x = s1$

$y = s2$

2 inputs

1 delay

6.20 Compare the logic size (number of gate inputs) and the delay (number of gate-delays) of a minimum binary encoding, an output encoding (if it is possible; if not, indicate why not), and a one-hot encoding of the laser timer FSM in Figure 3.47.

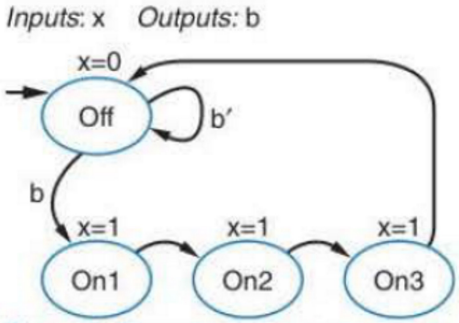


Figure 3.47 Laser timer state diagram assuming every transition is ANDed with a rising clock.

Minimum binary encoding

encodings off: 00

on1: 01

on2: 10

$n1 = s1 \text{ xor } s0$

on2: 10

on3: 11

n1 = s1 xor s0

n0 = s1's0'b + s1s0'

x = s1 + s0

s1	s2	b	n1	n0	x
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

11 inputs

2 delays

One-hot encoding

encodings

s0: 0001

s1: 0010

s2: 0100

s3: 1000

n3 = s2

n2 = s1

n1 = s0b

n0 = s0b' + s3

x = s3 + s2 + s1

9 inputs

2 delays

s3 s2 s1 s0 b | n3 n2 n1 n0 x

s3	s2	s1	s0	b	n3	n2	n1	n0	x
0	0	0	1	0	0	0	0	1	1
0	0	0	1	1	0	0	1	0	0
0	0	1	0	x	0	1	0	0	1
0	1	0	0	x	1	0	0	0	1
1	0	0	0	x	0	0	0	1	1