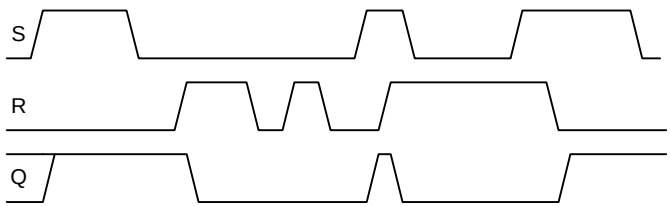
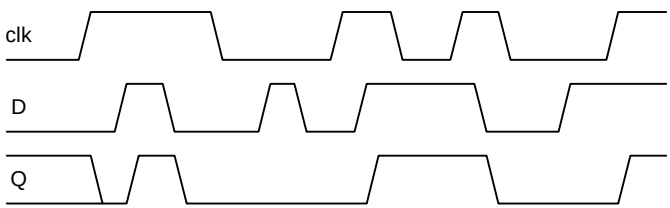
CHAPTER 3

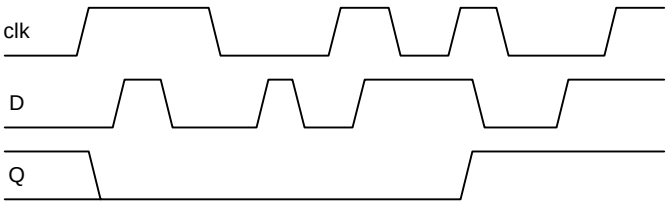
Exercise 3.1



Exercise 3.3



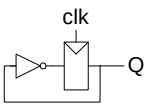
Exercise 3.5



Exercise 3.7

The circuit is sequential because it involves feedback and the output depends on previous values of the inputs. This is a SR latch. When $\overline{S} = 0$ and $\overline{R} = 1$, the circuit sets Q to 1. When $\overline{S} = 1$ and $\overline{R} = 0$, the circuit resets Q to 0. When both $\overline{S}$ and $\overline{R}$ are 1, the circuit remembers the old value. And when both $\overline{S}$ and $\overline{R}$ are 0, the circuit drives both outputs to 1.

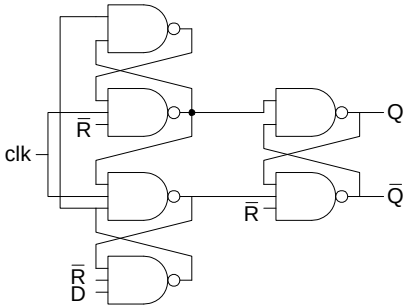
Exercise 3.9



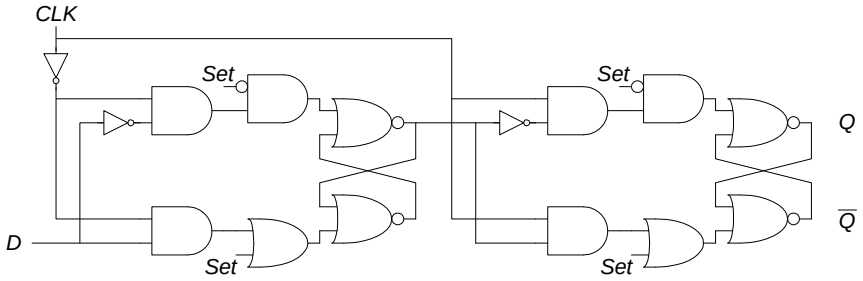
Exercise 3.11

If A and B have the same value, C takes on that value. Otherwise, C retains its old value.

Exercise 3.13



Exercise 3.15



Exercise 3.17

If N is even, the circuit is stable and will not oscillate.

Exercise 3.19

The system has at least five bits of state to represent the 24 floors that the elevator might be on.

Exercise 3.21

The FSM could be factored into four independent state machines, one for each student. Each of these machines has five states and requires 3 bits, so at least 12 bits of state are required for the factored design.

Exercise 3.23

This finite state machine asserts the output Q when A AND B is TRUE.

state	encoding $s_{1:0}$
S0	00
S1	01
S2	10

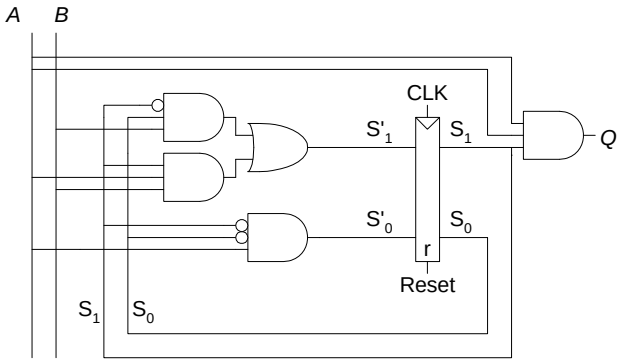
TABLE 3.1 State encoding for Exercise 3.23

current state		inputs		next state		output
s_1	s_0	a	b	s'_1	s'_0	q
0	0	0	X	0	0	0
0	0	1	X	0	1	0
0	1	X	0	0	0	0
0	1	X	1	1	0	0
1	0	1	1	1	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0

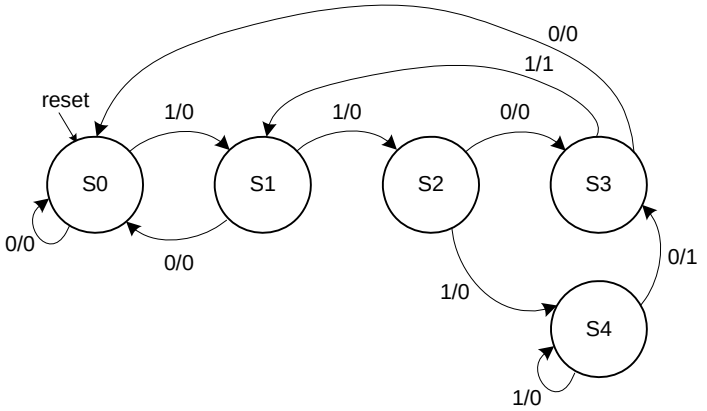
TABLE 3.2 Combined state transition and output table with binary encodings for Exercise 3.23

$$S'_1 = \overline{S}_1 S_0 B + S_1 A B$$
$$S'_0 = \overline{S}_1 \overline{S}_0 A$$

$$Q' = S_1 A B$$



Exercise 3.25



state	encoding $S_{1:0}$
S0	000
S1	001

TABLE 3.3 State encoding for Exercise 3.25

state	encoding $s_{1:0}$
S2	010
S3	100
S4	101

TABLE 3.3 State encoding for Exercise 3.25

current state			input	next state			output
s_2	s_1	s_0	a	s'_2	s'_1	s'_0	q
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	0
0	1	0	0	1	0	0	0
0	1	0	1	1	0	1	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0

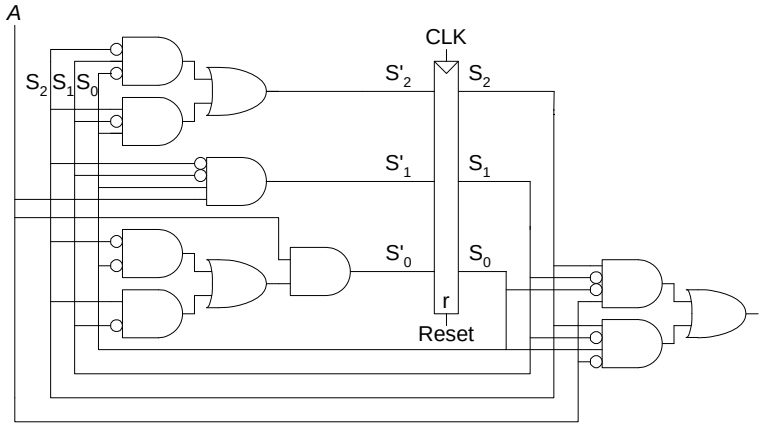
TABLE 3.4 Combined state transition and output table with binary encodings for Exercise 3.25

$$S'_2 = \overline{S}_2 S_1 \overline{S}_0 + S_2 \overline{S}_1 S_0$$

$$S'_1 = \overline{S}_2 \overline{S}_1 S_0 A$$

$$S'_0 = A(\overline{S}_2 \overline{S}_0 + S_2 \overline{S}_1)$$

$$Q = S_2 \overline{S}_1 \overline{S}_0 A + S_2 \overline{S}_1 S_0 \overline{A}$$



Exercise 3.27

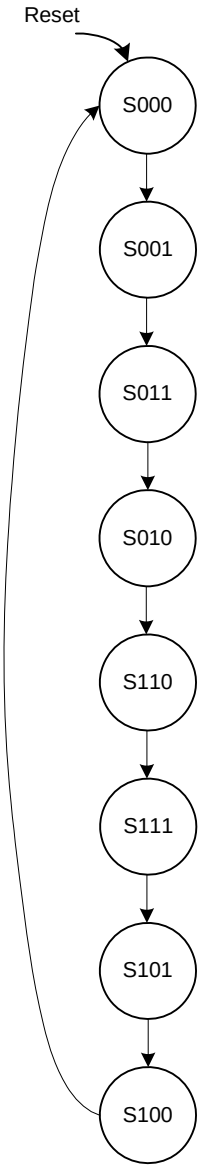


FIGURE 3.1 State transition diagram for Exercise 3.27

current state $s_{2:0}$	next state $s'_{2:0}$
000	001
001	011
011	010
010	110
110	111
111	101
101	100
100	000

TABLE 3.5 State transition table for Exercise 3.27

$$S'_2 = S_1\overline{S_0} + S_2S_0$$
$$S'_1 = \overline{S_2}S_0 + S_1\overline{S_0}$$
$$S'_0 = \overline{S_2 \oplus S_1}$$
$$Q_2 = S_2$$
$$Q_1 = S_1$$
$$Q_0 = S_0$$

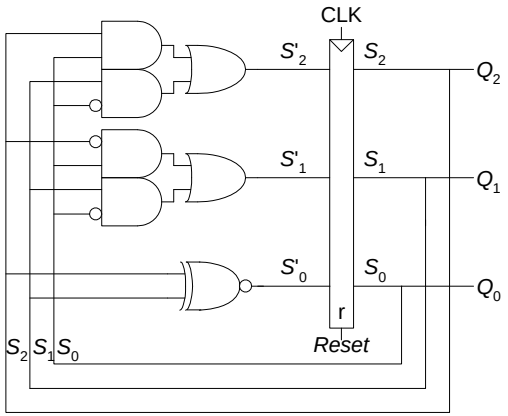


FIGURE 3.2 Hardware for Gray code counter FSM for Exercise 3.27

Exercise 3.29

(a)

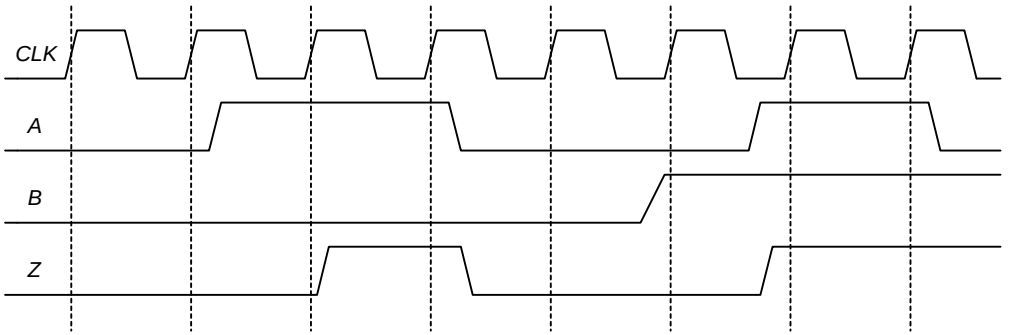


FIGURE 3.3 Waveform showing *Z* output for Exercise 3.29

(b) This FSM is a Mealy FSM because the output depends on the current value of the input as well as the current state.

(c)

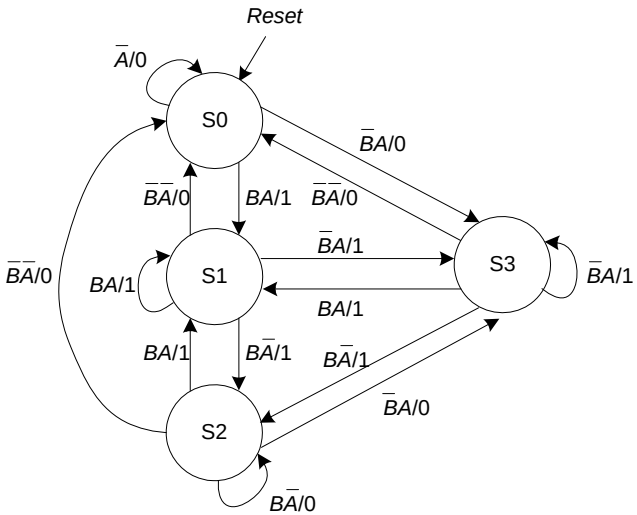


FIGURE 3.4 State transition diagram for Exercise 3.29

(Note: another viable solution would be to allow the state to transition from S0 to S1 on $B\bar{A}/0$. The arrow from S0 to S0 would then be $\bar{B}\bar{A}/0$.)

current state $s_{1:0}$	inputs		next state $s'_{1:0}$	output
	b	a		z
00	X	0	00	0
00	0	1	11	0
00	1	1	01	1
01	0	0	00	0
01	0	1	11	1
01	1	0	10	1
01	1	1	01	1
10	0	X	00	0
10	1	0	10	0

TABLE 3.6 State transition table for Exercise 3.29

current state $s_{1:0}$	inputs		next state $s'_{1:0}$	output
	b	*a*		z
10	1	1	01	1
11	0	0	00	0
11	0	1	11	1
11	1	0	10	1
11	1	1	01	1

TABLE 3.6 State transition table for Exercise 3.29

$$S'_1 = \bar{B}A(\overline{S_1} + S_0) + B\bar{A}(S_1 + \overline{S_0})$$

$$S'_0 = A(\overline{S_1} + S_0 + B)$$

$$Z = BA + S_0(A + B)$$

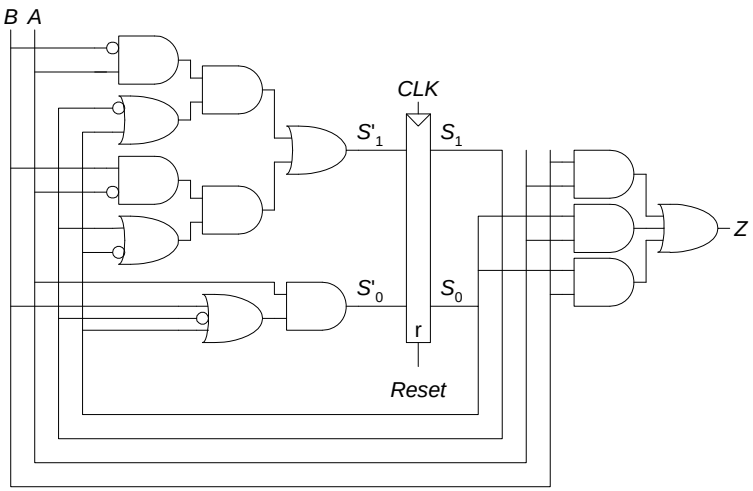


FIGURE 3.5 Hardware for FSM of Exercise 3.26

Note: One could also build this functionality by registering input *A*, producing both the logical AND and OR of input *A* and its previous (registered)

value, and then muxing the two operations using *B*. The output of the mux is *Z*: *Z* = *AA*prev (if *B* = 0); *Z* = *A* + *A*prev (if *B* = 1).

Exercise 3.31

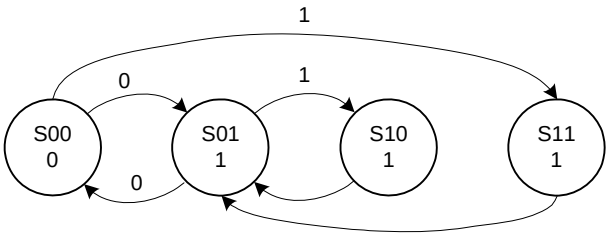
This finite state machine is a divide-by-two counter (see Section 3.4.2) when X = 0. When X = 1, the output, *Q*, is HIGH.

current state		input	next state	
s_1	s_0	x	s'_1	s'_0
0	0	0	0	1
0	0	1	1	1
0	1	0	0	0
0	1	1	1	0
1	X	X	0	1

TABLE 3.7 State transition table with binary encodings for Exercise 3.31

current state		output
s_1	s_0	q
0	0	0
0	1	1
1	X	1

TABLE 3.8 Output table for Exercise 3.31



Exercise 3.33

(a) First, we calculate the propagation delay through the combinational logic:

$t_{pd} = 3t_{pd\_XOR}$
$= 3 \times 100$ ps
$=$ **300 ps**

Next, we calculate the cycle time:

$T_c \geq t_{pcq} + t_{pd} + t_{setup}$
$\geq [70 + 300 + 60]$ ps
$= 430$ ps
$f = 1 / 430$ ps = **2.33 GHz**

(b)

$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$

Thus,

$t_{skew} \leq T_c - (t_{pcq} + t_{pd} + t_{setup})$, where $T_c = 1 / 2$ GHz = 500 ps
$\leq [500 - 430]$ ps = **70 ps**

(c)

First, we calculate the contamination delay through the combinational logic:

$t_{cd} = t_{cd\_XOR}$
$= 55$ ps

$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$

Thus,

$t_{skew} < (t_{ccq} + t_{cd}) - t_{hold}$
$< (50 + 55) - 20$
< 85 ps

(d)

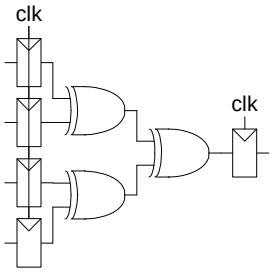


FIGURE 3.6 Alyssa's improved circuit for Exercise 3.33

First, we calculate the propagation and contamination delays through the combinational logic:

$t_{pd} = 2t_{pd\_XOR}$
$= 2 \times 100$ ps
$=$ **200 ps**
$t_{cd} = 2t_{cd\_XOR}$
$= 2 \times 55$ ps
$=$ **110 ps**

Next, we calculate the cycle time:

$T_c \geq t_{pcq} + t_{pd} + t_{setup}$
$\geq [70 + 200 + 60]$ ps
$= 330$ ps
$f = 1 / 330$ ps = **3.03 GHz**

$t_{skew} < (t_{ccq} + t_{cd}) - t_{hold}$
$< (50 + 110) - 20$
< 140 ps

Exercise 3.35

(a) $T_c = 1 / 40$ MHz = 25 ns
$T_c \geq t_{pcq} + Nt_{CLB} + t_{setup}$
25 ns $\geq [0.72 + N(0.61) + 0.53]$ ps
Thus, N < 38.9
N = 38

(b)

$t_{skew} < (t_{ccq} + t_{cd\_CLB}) - t_{hold}$
$< [(0.5 + 0.3) - 0]$ ns
< 0.8 ns = 800 ps

Exercise 3.37

P(failure)/sec = 1/MTBF = 1/(50 years * 3.15 x 10^7 sec/year) = **6.34 x 10^{-10}** (EQ 3.26)

P(failure)/sec waiting for one clock cycle: $N*(T_0/T_c)*e^{-(Tc-tsetup)/Tau}$

$= 0.5 * (110/1000) * e^{-(1000-70)/100} = 5.0 \times 10^{-6}$

P(failure)/sec waiting for two clock cycles: $N*(T_0/T_c)*[e^{-(Tc-tsetup)/Tau}]^2$

$= 0.5 * (110/1000) * [e^{-(1000-70)/100}]^2 = 4.6 \times 10^{-10}$

This is just less than the required probability of failure (6.34 x 10^{-10}). Thus, **2 cycles** of waiting is just adequate to meet the MTBF.

Exercise 3.39

We assume a two flip-flop synchronizer. The most significant impact on the probability of failure comes from the exponential component. If we ignore the T_0/T_c term in the probability of failure equation, assuming it changes little with increases in cycle time, we get:

$$P(\text{failure}) = e^{-\frac{t}{\tau}}$$

$$MTBF = \frac{1}{P(failure)} = e^{\frac{T_c - t_{setup}}{\tau}}$$

$$\frac{MTBF_2}{MTBF_1} = 10 = e^{\frac{T_{c2} - T_{c1}}{30ps}}$$

Solving for $T_{c2} - T_{c1}$, we get:

$$T_{c2} - T_{c1} = 69ps$$

Thus, the clock cycle time must increase by **69 ps**. This holds true for cycle times much larger than T0 (20 ps) and the increased time (69 ps).

Question 3.1

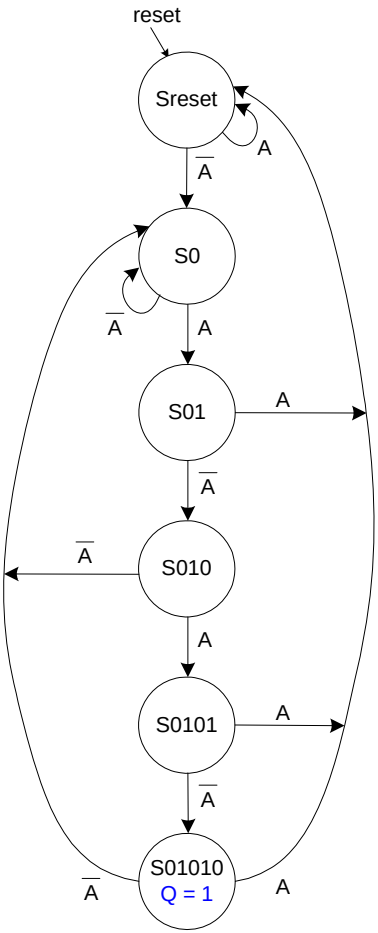


FIGURE 3.7 State transition diagram for Question 3.1

current state $s_{5:0}$	input a	next state $s'_{5:0}$
000001	0	000010
000001	1	000001
000010	0	000010
000010	1	000100
000100	0	001000
000100	1	000001
001000	0	000010
001000	1	010000
010000	0	100000
010000	1	000001
100000	0	000010
100000	1	000001

TABLE 3.9 State transition table for Question 3.1

$$S'_5 = S_4 A$$
$$S'_4 = S_3 A$$
$$S'_3 = S_2 A$$
$$S'_2 = S_1 A$$
$$S'_1 = A(S_1 + S_3 + S_5)$$
$$S'_0 = A(S_0 + S_2 + S_4 + S_5)$$
$$Q = S_5$$

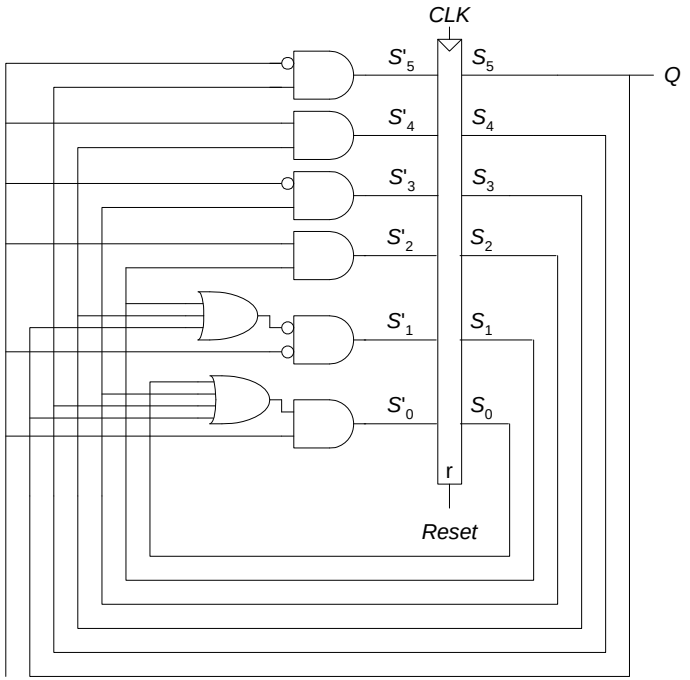


FIGURE 3.8 Finite state machine hardware for Question 3.1

Question 3.3

A latch allows input *D* to flow through to the output *Q* when the clock is HIGH. A flip-flop allows input *D* to flow through to the output *Q* at the clock edge. A flip-flop is preferable in systems with a single clock. Latches are preferable in *two-phase clocking* systems, with two clocks. The two clocks are used to eliminate system failure due to hold time violations. Both the phase and frequency of each clock can be modified independently.

Question 3.5

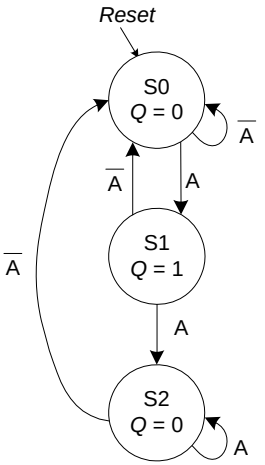


FIGURE 3.9 State transition diagram for edge detector circuit of Question 3.5

current state $s_{1:0}$	input a	next state $s'_{1:0}$
00	0	00
00	1	01
01	0	00
01	1	10
10	0	00
10	1	10

TABLE 3.10 State transition table for Question 3.5

$$S'_1 = AS_1$$

$$S'_0 = AS_1S_0$$

$Q = S_1$

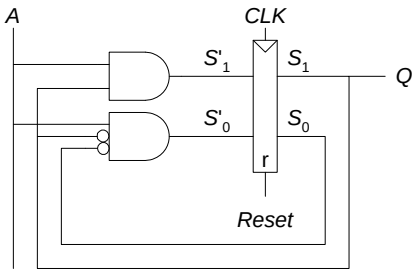


FIGURE 3.10 Finite state machine hardware for Question 3.5

Question 3.7

A flip-flop with a negative hold time allows *D* to start changing *before* the clock edge arrives.

Question 3.9

Without the added buffer, the propagation delay through the logic, t_{pd}, must be less than or equal to $T_c - (t_{pcq} + t_{\text{setup}})$. However, if you add a buffer to the clock input of the receiver, the clock arrives at the receiver later. The earliest that the clock edge arrives at the receiver is $t_{\text{cd\_BUF}}$ after the actual clock edge. Thus, the propagation delay through the logic is now given an extra t_{cd_BUF}. So, t_{pd} now must be less than $T_c + t_{cd\_\text{BUF}} - (t_{pcq} + t_{\text{setup}})$.