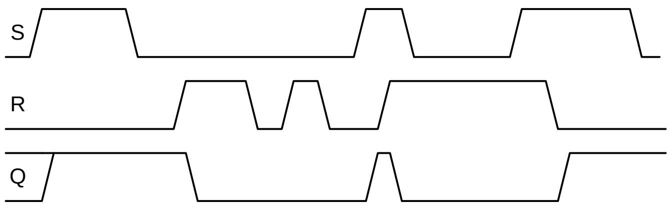
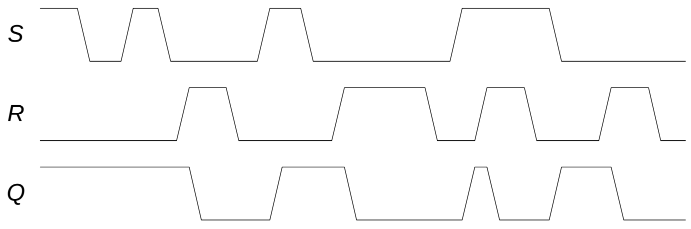
CHAPTER 3

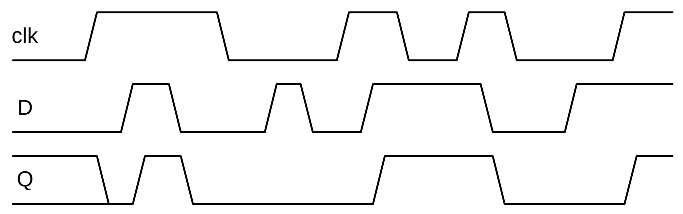
Exercise 3.1



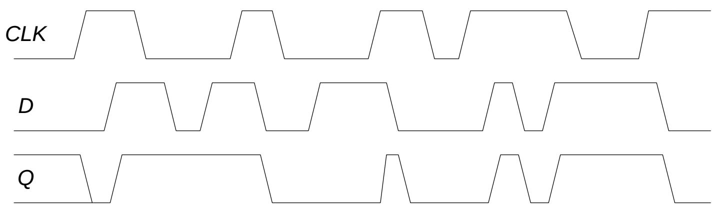
Exercise 3.2



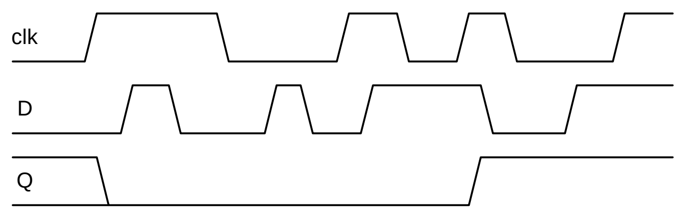
Exercise 3.3



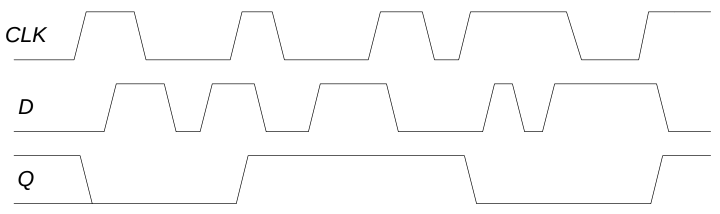
Exercise 3.4



Exercise 3.5



Exercise 3.6



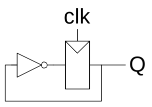
Exercise 3.7

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.

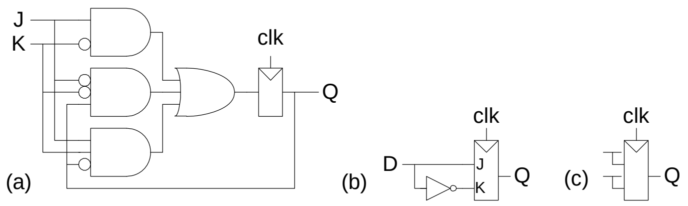
Exercise 3.8

Sequential logic. This is a D flip-flop with active low asynchronous set and reset inputs. If $\overline{S}$ and $\overline{R}$ are both 1, the circuit behaves as an ordinary D flip-flop. If $\overline{S}$ = 0, Q is immediately set to 0. If $\overline{R}$ = 0, Q is immediately reset to 1. (This circuit is used in the commercial 7474 flip-flop.)

Exercise 3.9



Exercise 3.10

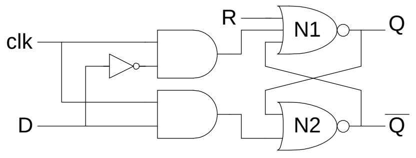


Exercise 3.11

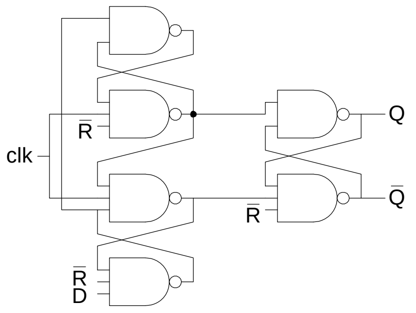
If A and B have the same value, C takes on that value. Otherwise, C retains its old value.

Exercise 3.12

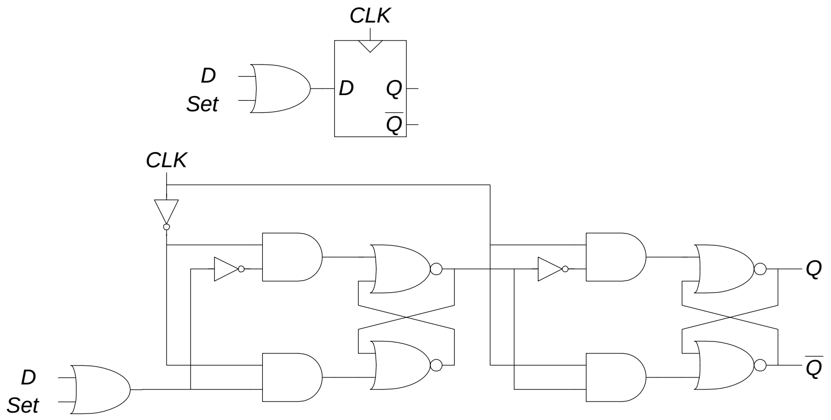
Make sure these next ones are correct too.



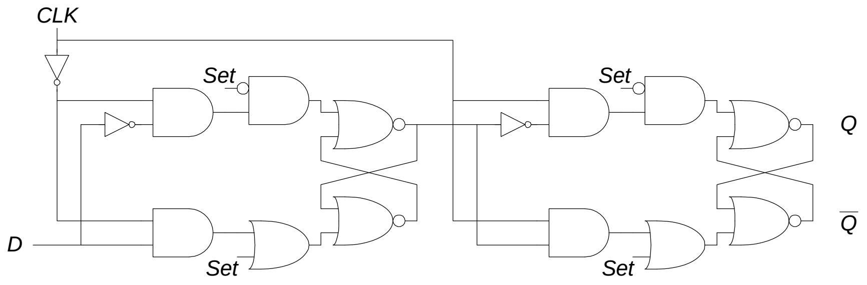
Exercise 3.13



Exercise 3.14



Exercise 3.15



Exercise 3.16

From $\dfrac{1}{2Nt_{pd}}$ to $\dfrac{1}{2Nt_{cd}}$.

Exercise 3.17

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

Exercise 3.18

(a) No: no register. (b) No: feedback without passing through a register. (c) Yes. Satisfies the definition. (d) Yes. Satisfies the definition.

Exercise 3.19

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

Exercise 3.20

The FSM has 5^4 = 625 states. This requires at least 10 bits to represent all the states.

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 3.22

This finite state machine asserts the output *Q* for one clock cycle if *A* is TRUE followed by *B* being TRUE.

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

TABLE 3.1 State encoding for Exercise 3.22

.

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

TABLE 3.2 State transition table with binary encodings for Exercise 3.22

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

TABLE 3.2 State transition table with binary encodings for Exercise 3.22

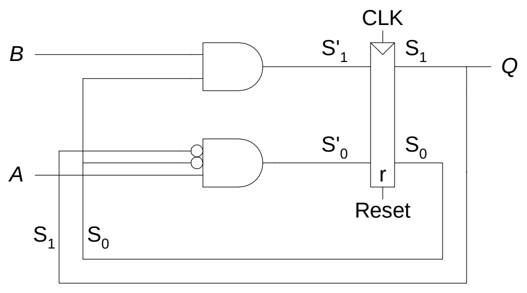
.

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

TABLE 3.3 Output table with binary encodings for Exercise 3.22

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

$$Q = S_1$$



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 3.4 State encoding for Exercise 3.23

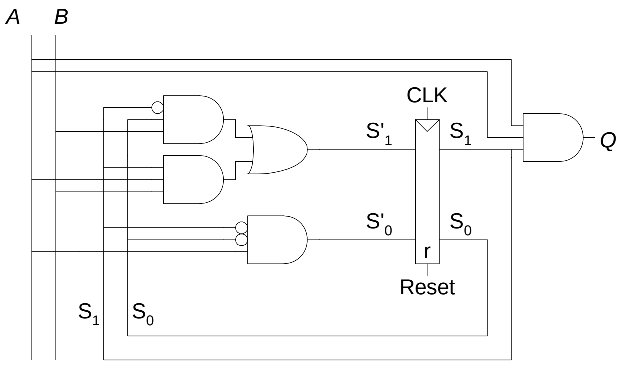
current state		inputs		next state		output
s_1	s_0	a	b	s'_1	s'_0	q
0	0	0	X	0	0	0
0	0	1	X	0	1	0
0	1	X	0	0	0	0
0	1	X	1	1	0	0
1	0	1	1	1	0	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0

TABLE 3.5 Combined state transition and output table with binary encodings for Exercise 3.23

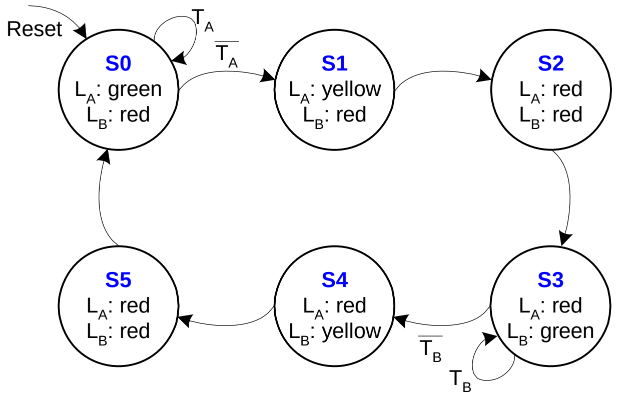
$$S'_1 = \overline{S_1}S_0B + S_1AB$$

$$S'_0 = \overline{S_1}\,\overline{S_0}A$$

$$Q' = S_1AB$$



Exercise 3.24



state	encoding $S_{1:0}$
S0	000
S1	001
S2	010

TABLE 3.6 State encoding for Exercise 3.24

state	encoding $S_{1:0}$
S3	100
S4	101
S5	110

TABLE 3.6 State encoding for Exercise 3.24

current state			inputs		next state		
S_2	S_1	S_0	t_a	t_b	S'_2	S'_1	S'_0
0	0	0	0	X	0	0	1
0	0	0	1	X	0	0	0
0	0	1	X	X	0	1	0
0	1	0	X	X	1	0	0
1	0	0	X	0	1	0	1
1	0	0	X	1	1	0	0
1	0	1	X	X	1	1	0
1	1	0	X	X	0	0	0

TABLE 3.7 State transition table with binary encodings for Exercise 3.24

$$S'_2 = S_2 \oplus S_1$$

$$S'_1 = \overline{S_1}S_0$$

$$S'_0 = \overline{S_1}\,\overline{S_0}(\overline{S_2}\,\overline{t_a} + S_2\overline{t_b})$$

current state			outputs			
s_2	s_1	s_0	l_{a1}	l_{a0}	l_{b1}	l_{b0}
0	0	0	0	0	1	0
0	0	1	0	1	1	0
0	1	0	1	0	1	0
1	0	0	1	0	0	0
1	0	1	1	0	0	1
1	1	0	1	0	1	0

TABLE 3.8 Output table for Exercise 3.24

$$\begin{aligned} L_{A1} &= S_1\overline{S_0} + S_2\overline{S_1} \\ L_{A0} &= \overline{S_2}S_0 \\ L_{B1} &= \overline{S_2}\,\overline{S_1} + S_1\overline{S_0} \\ L_{B0} &= S_2\overline{S_1}S_0 \end{aligned} \qquad (3.1)$$

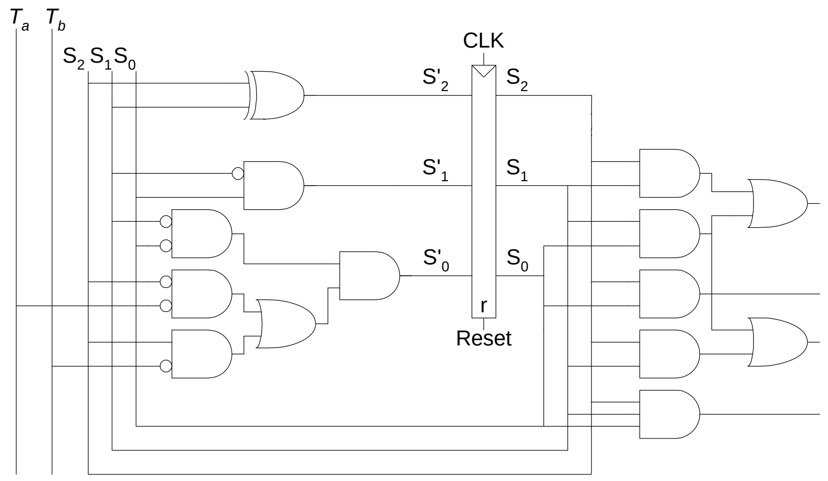
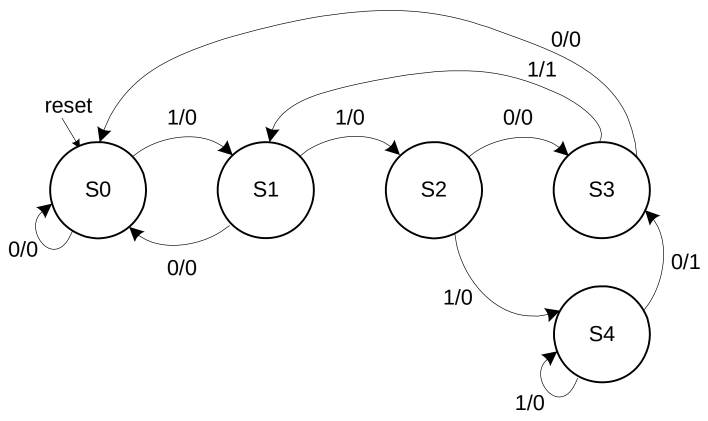


FIGURE 3.1 State machine circuit for traffic light controller for Exercise 3.21

Exercise 3.25



state	encoding $s_{1:0}$
S0	000
S1	001
S2	010
S3	100
S4	101

TABLE 3.9 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

TABLE 3.10 Combined state transition and output table with binary encodings 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	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

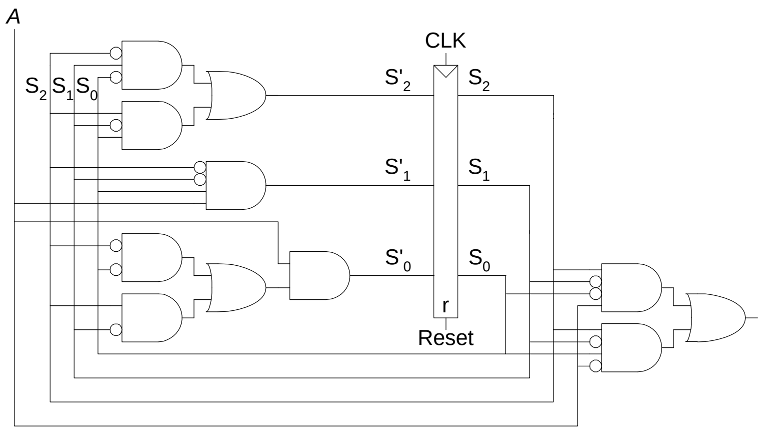
TABLE 3.10 Combined state transition and output table with binary encodings for Exercise 3.25

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

$$S'_1 = \overline{S_2}\overline{S_1}S_0A$$

$$S'_0 = A(\overline{S_2}\overline{S_0} + S_2\overline{S_1})$$

$$Q = S_2\overline{S_1}\overline{S_0}A + S_2\overline{S_1}S_0\overline{A}$$



Exercise 3.26

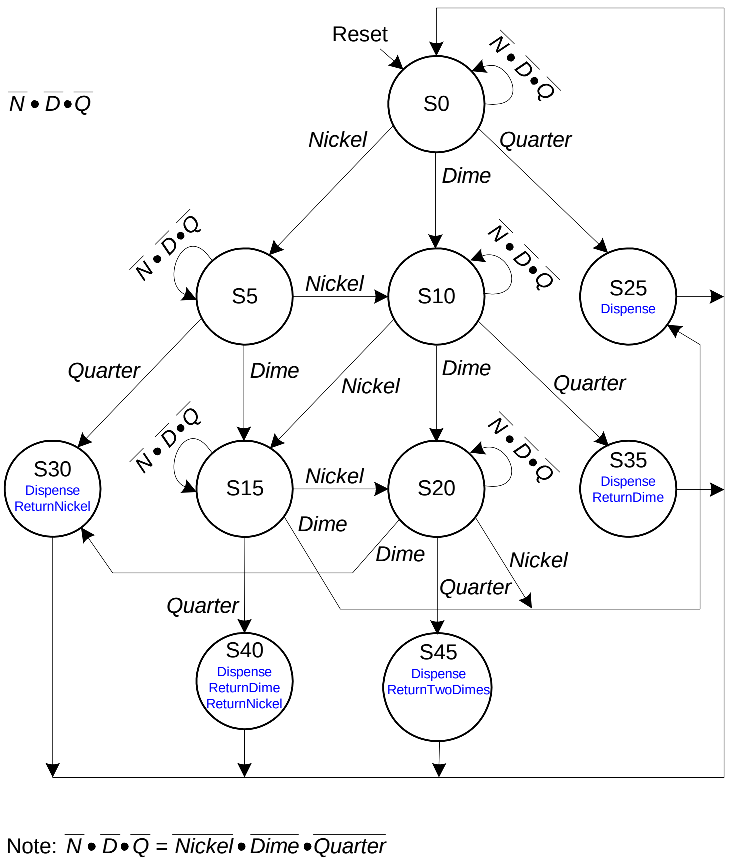

Reset
$\overline{N} \bullet \overline{D} \bullet \overline{Q}$
S0
Nickel
Dime
Quarter
S5
S10
S25
Dispense
S30
Dispense
ReturnNickel
S15
S20
S35
Dispense
ReturnDime
S40
Dispense
ReturnDime
ReturnNickel
S45
Dispense
ReturnTwoDimes
Note: $\overline{N} \bullet \overline{D} \bullet \overline{Q} = \overline{Nickel} \bullet \overline{Dime} \bullet \overline{Quarter}$

FIGURE 3.2 State transition diagram for soda machine dispense of Exercise 3.23

state	encoding $s_{9:0}$
S0	0000000001
S5	0000000010
S10	0000000100
S25	0000001000
S30	0000010000
S15	0000100000
S20	0001000000
S35	0010000000
S40	0100000000
S45	1000000000

FIGURE 3.3 State Encodings for Exercise 3.26

current state *s*	inputs			next state *s'*
	nickel	*dime*	*quarter*	
S0	0	0	0	S0
S0	0	0	1	S25
S0	0	1	0	S10
S0	1	0	0	S5
S5	0	0	0	S5
S5	0	0	1	S30
S5	0	1	0	S15
S5	1	0	0	S10
S10	0	0	0	S10

TABLE 3.11 State transition table for Exercise 3.26

current state *s*	inputs			next state *s'*
	nickel	*dime*	*quarter*	
S10	0	0	1	S35
S10	0	1	0	S20
S10	1	0	0	S15
S25	X	X	X	S0
S30	X	X	X	S0
S15	0	0	0	S15
S15	0	0	1	S40
S15	0	1	0	S25
S15	1	0	0	S20
S20	0	0	0	S20
S20	0	0	1	S45
S20	0	1	0	S30
S20	1	0	0	S25
S35	X	X	X	S0
S40	X	X	X	S0
S45	X	X	X	S0

TABLE 3.11 State transition table for Exercise 3.26

current state *s*	inputs			next state *s'*
	nickel	*dime*	*quarter*	
0000000001	0	0	0	0000000001
0000000001	0	0	1	0000001000
0000000001	0	1	0	0000000100
0000000001	1	0	0	0000000010

TABLE 3.12 State transition table for Exercise 3.26

current state *s*	inputs			next state *s'*
	nickel	*dime*	*quarter*	
0000000010	0	0	0	0000000010
0000000010	0	0	1	0000010000
0000000010	0	1	0	0000100000
0000000010	1	0	0	0000000100
0000000100	0	0	0	0000000100
0000000100	0	0	1	0010000000
0000000100	0	1	0	0001000000
0000000100	1	0	0	0000100000
0000001000	X	X	X	0000000001
0000010000	X	X	X	0000000001
0000100000	0	0	0	0000100000
0000100000	0	0	1	0100000000
0000100000	0	1	0	0000001000
0000100000	1	0	0	0001000000
0001000000	0	0	0	0001000000
0001000000	0	0	1	1000000000
0001000000	0	1	0	0000010000
0001000000	1	0	0	0000001000
0010000000	X	X	X	0000000001
0100000000	X	X	X	0000000001
1000000000	X	X	X	0000000001

TABLE 3.12 State transition table for Exercise 3.26

$$S'_9 = S_6 Q$$

$$S'_8 = S_5 Q$$

$$S'_7 = S_2Q$$

$$S'_6 = S_2D + S_5N + S_6\overline{N}\overline{D}\overline{Q}$$

$$S'_5 = S_1D + S_2N + S_5NDQ$$

$$S'_4 = S_1Q + S_6D$$

$$S'_3 = S_0Q + S_5D + S_6N$$

$$S'_2 = S_0D + S_1N + S_2\overline{N}\overline{D}\overline{Q}$$

$$S'_1 = S_0N + S_1NDQ$$

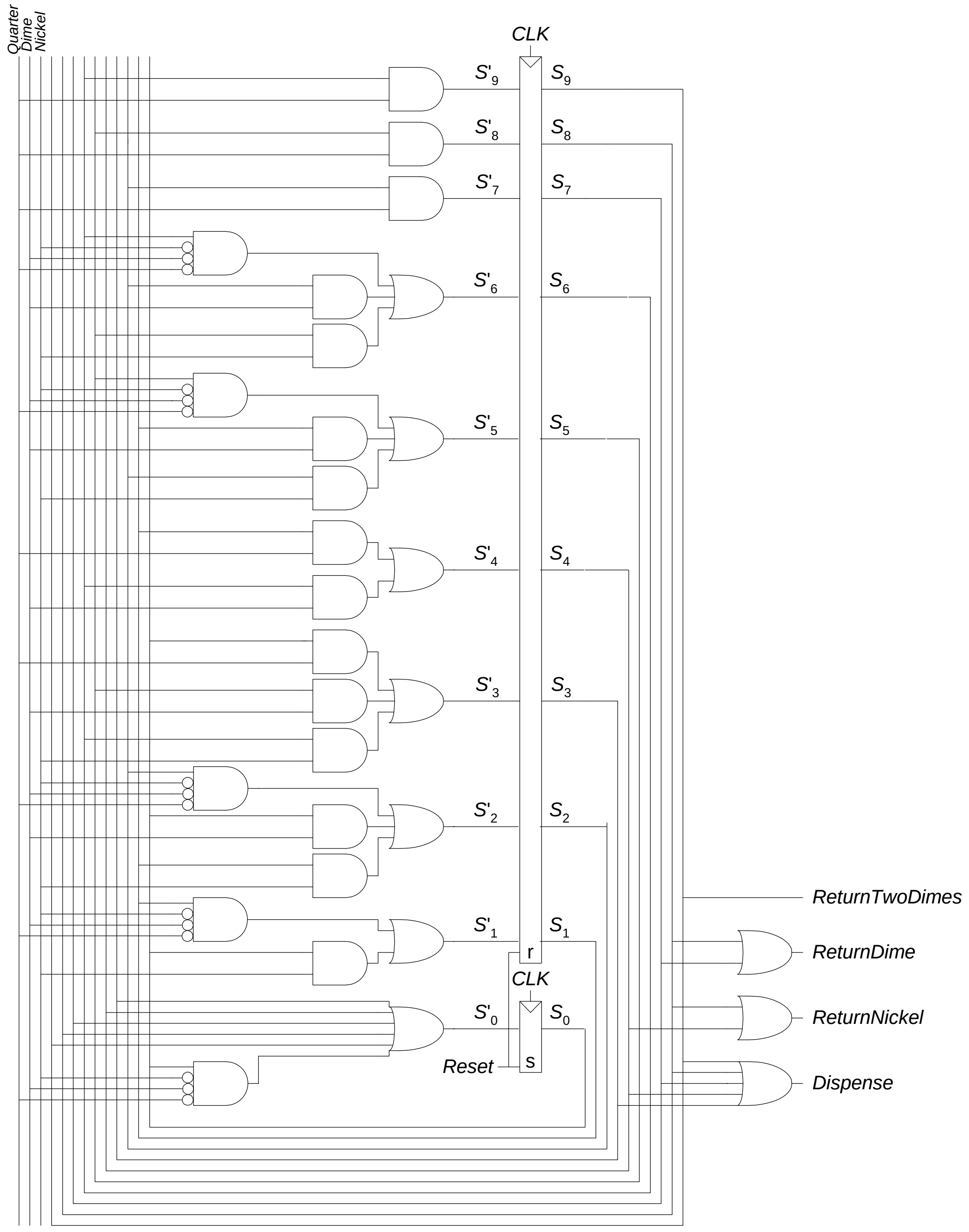
$$S'_0 = S_0\overline{N}\overline{D}\overline{Q} + S_3 + S_4 + S_7 + S_8 + S_9$$

$$Dispense = S_3 + S_4 + S_7 + S_8 + S_9$$

$$ReturnNickel = S_4 + S_8$$

$$ReturnDime = S_7 + S_8$$

$$ReturnTwoDimes = S_9$$


Quarter
Dime
Nickel
CLK
S'9
S9
S'8
S8
S'7
S7
S'6
S6
S'5
S5
S'4
S4
S'3
S3
S'2
S2
S'1
S1
r
CLK
S'0
S0
s
Reset
ReturnTwoDimes
ReturnDime
ReturnNickel
Dispense

Exercise 3.27

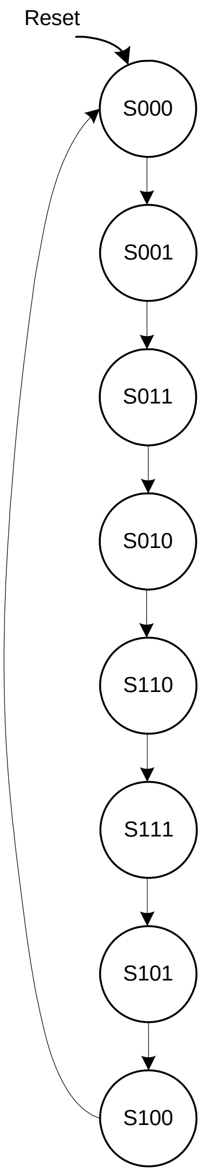


FIGURE 3.4 State transition diagram for Exercise 3.27

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 3.13 State transition table for Exercise 3.27

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

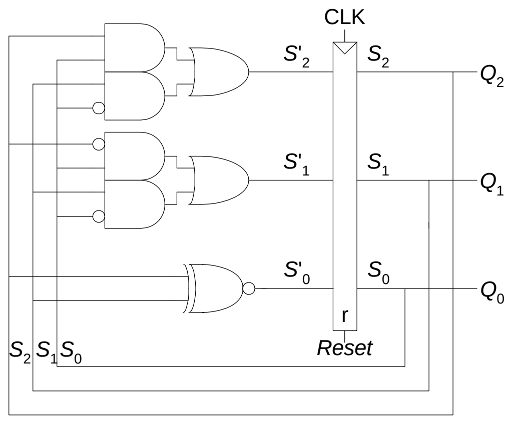


FIGURE 3.5 Hardware for Gray code counter FSM for Exercise 3.27

Exercise 3.28

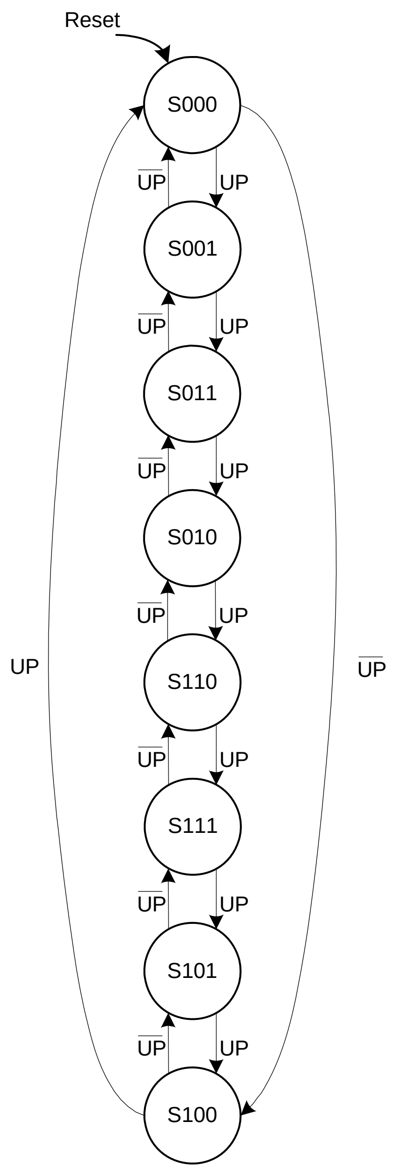


FIGURE 3.6 State transition diagram for Exercise 3.28

current state $S_{2:0}$	input up	next state $S'_{2:0}$
000	1	001
001	1	011
011	1	010
010	1	110
110	1	111
111	1	101
101	1	100
100	1	000
000	0	100
001	0	000
011	0	001
010	0	011
110	0	010
111	0	110
101	0	111
100	0	101

TABLE 3.14 State transition table for Exercise 3.28

$$S'_2 = UPS_1\overline{S_0} + \overline{UP}\overline{S_1}\overline{S_0} + S_2S_0$$

$$S'_1 = S_1\overline{S_0} + UP\overline{S_2}S_0 + \overline{UP}S_2S_1$$

$$S'_0 = UP \oplus S_2 \oplus S_1$$

$$Q_2 = S_2$$

$$Q_1 = S_1$$

$$Q_0 = S_0$$

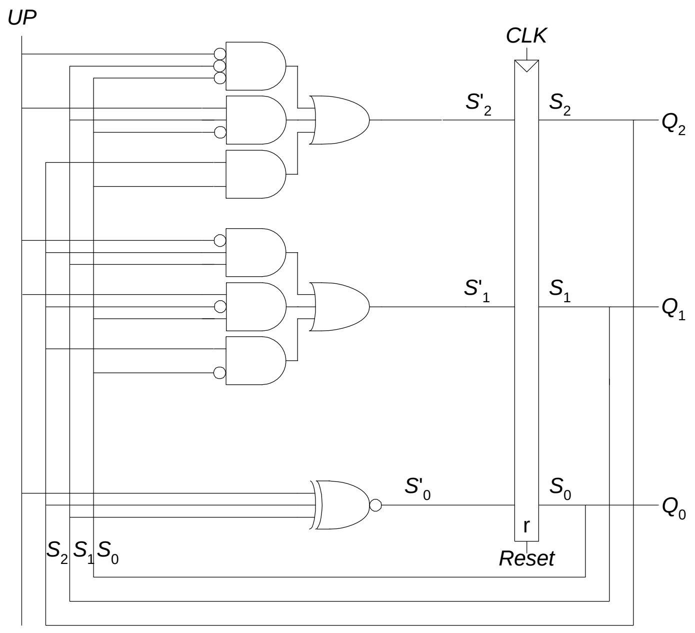


FIGURE 3.7 Finite state machine hardware for Exercise 3.28

Exercise 3.29

(a)

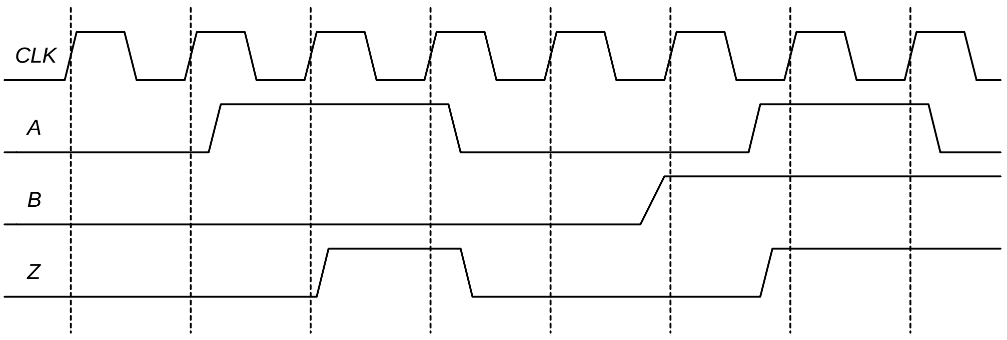


FIGURE 3.8 Waveform showing *Z* output for Exercise 3.29

(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)

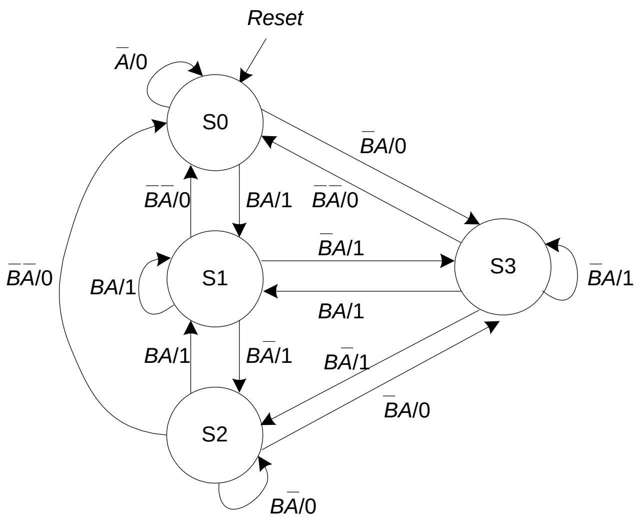


FIGURE 3.9 State transition diagram for Exercise 3.29

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

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 3.15 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 3.15 State transition table for Exercise 3.29

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

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

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

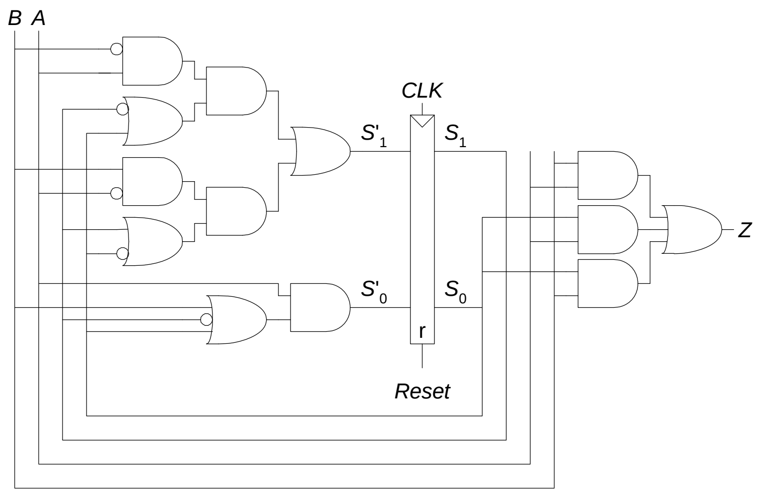


FIGURE 3.10 Hardware for FSM of Exercise 3.26

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: $Z = A$Aprev (if $B = 0$); $Z = A$ + Aprev (if $B = 1$).

Exercise 3.30

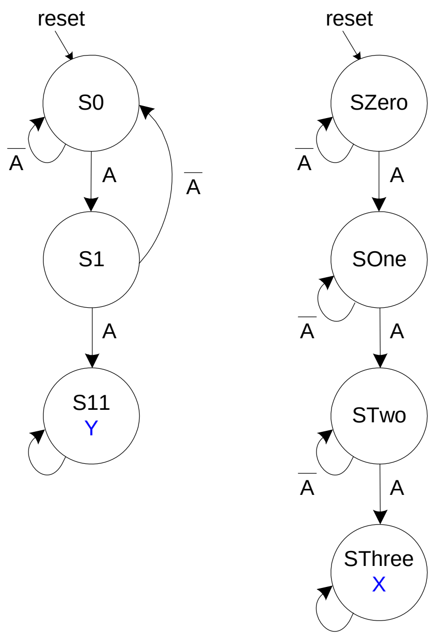


FIGURE 3.11 Factored state transition diagram for Exercise 3.30

current state $S_{1:0}$	input a	next state $S'_{1:0}$
00	0	00
00	1	01
01	0	00

TABLE 3.16 State transition table for output Y for Exercise 3.30

current state $s_{1:0}$	input a	next state $s'_{1:0}$
01	1	11
11	X	11

TABLE 3.16 State transition table for output *Y* for Exercise 3.30

current state $t_{1:0}$	input a	next state $t'_{1:0}$
00	0	00
00	1	01
01	0	01
01	1	10
10	0	10
10	1	11
11	X	11

TABLE 3.17 State transition table for output *X* for Exercise 3.30

$$S'_1 = S_0(S_1 + A)$$

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

$$Y = S_1$$

$$T'_1 = T_1 + T_0A$$

$$T'_0 = A(T_1 + \overline{T_0}) + \overline{A}T_0 + T_1T_0$$

$$X = T_1T_0$$

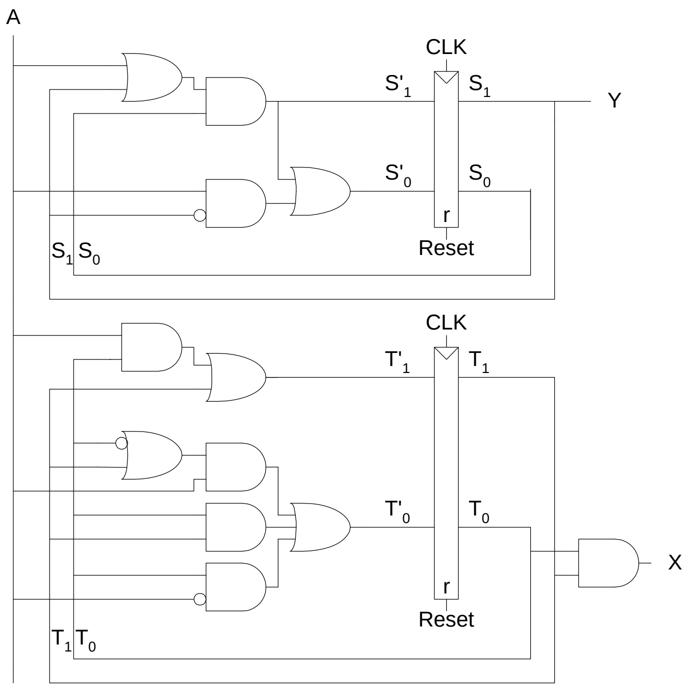


FIGURE 3.12 Finite state machine hardware for Exercise 3.30

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

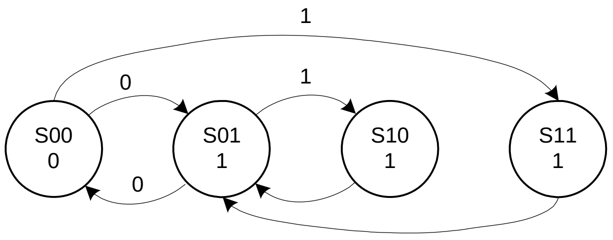
TABLE 3.18 State transition table with binary encodings for Exercise 3.31

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

TABLE 3.18 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 3.19 Output table for Exercise 3.31



Exercise 3.32

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

TABLE 3.20 State transition table with binary encodings for Exercise 3.32

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

TABLE 3.20 State transition table with binary encodings for Exercise 3.32

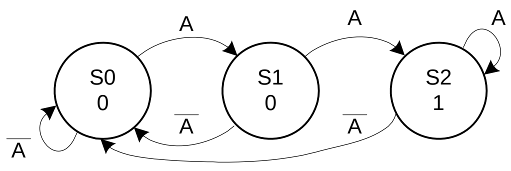


FIGURE 3.13 State transition diagram for Exercise 3.32

Q asserts whenever *A* is HIGH for two or more consecutive cycles.

Exercise 3.33

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

$t_{pd} = 3t_{pd\_\text{XOR}}$
$= 3 \times 100$ ps
$=$ **300 ps**

Next, we calculate the cycle time:

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

(b)

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

Thus,

$t_{\text{skew}} \leq T_c - (t_{pcq} + t_{pd} + t_{\text{setup}})$, 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:

$t_{cd} = t_{cd\_\text{XOR}}$
$= 55$ ps

$t_{ccq} + t_{cd} > t_{\text{hold}} + t_{\text{skew}}$
Thus,
$t_{\text{skew}} < (t_{ccq} + t_{cd}) - t_{\text{hold}}$
$< (50 + 55) - 20$
< 85 ps

(d)

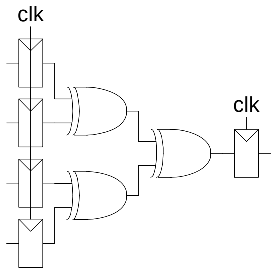


FIGURE 3.14 Alyssa's improved circuit for Exercise 3.33

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

$t_{pd} = 2t_{pd\_\text{XOR}}$
$= 2 \times 100$ ps
$= 200$ ps
$t_{cd} = 2t_{cd\_\text{XOR}}$
$= 2 \times 55$ ps
$= 110$ ps

Next, we calculate the cycle time:
$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}}$
$\geq [70 + 200 + 60]$ ps
$= 330$ ps
$f = 1 / 330$ ps $=$ **3.03 GHz**

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

Exercise 3.34

(a) 9.09 GHz
(b) 15 ps
(c) 26 ps

Exercise 3.35

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

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

Exercise 3.36

1.138 ns

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: $N*(T_0/T_c)*e^{-(Tc\text{-}tsetup)/Tau}$

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

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

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

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 3.38

(a) You know you've already entered metastability, so the probability that the sampled signal is metastable is 1. Thus,

$$P(\text{failure}) = 1 \times e^{-\frac{t}{\tau}}$$

Solving for the probability of still being metastable (failing) to be 0.01:

$$P(\text{failure}) = e^{-\frac{t}{\tau}} = 0.01$$

Thus,

$$t = -\tau \times \ln(P(failure)) = -20 \times \ln((0.01)) = \textbf{92 seconds}$$

(b) The probability of death is the chance of still being metastable after 3 minutes

P(failure) = 1 × e $^{-(3\text{ min} \times 60\text{ sec}) / 20\text{ sec}}$ = **0.000123**

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).

Exercise 3.40

Alyssa is correct. Ben's circuit does not eliminate metastability. After the first transition on D, D2 is always 0 because as D2 transitions from 0 to 1 or 1 to 0, it enters the forbidden region and Ben's "metastability detector" resets the first flip-flop to 0. Even if Ben's circuit could correctly detect a metastable output, it would asynchronously reset the flip-flop which, if the reset occurred around the clock edge, this could cause the second flip-flop to sample a transitioning signal and become metastable.

Question 3.1

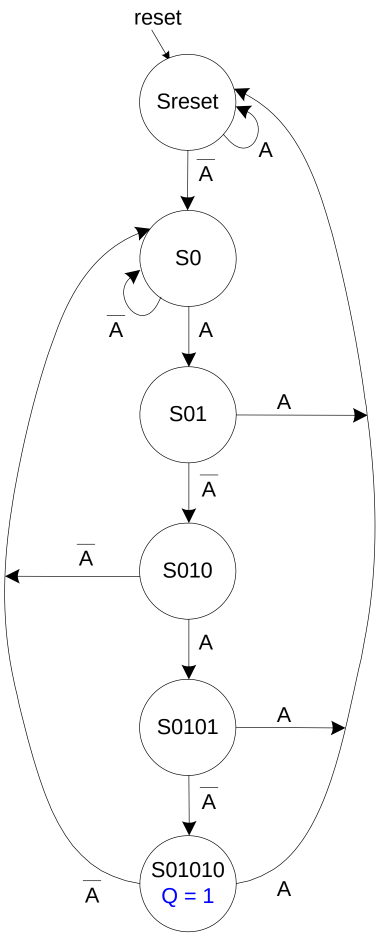


FIGURE 3.15 State transition diagram for Question 3.1

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

TABLE 3.21 State transition table for Question 3.1

current state $S_{5:0}$	input A	next state $S'_{5:0}$
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 3.21 State transition table for Question 3.1

$$S'_5 = S_4A$$
$$S'_4 = S_3A$$
$$S'_3 = S_2A$$
$$S'_2 = S_1A$$
$$S'_1 = A(S_1 + S_3 + S_5)$$
$$S'_0 = A(S_0 + S_2 + S_4 + S_5)$$
$$Q = S_5$$

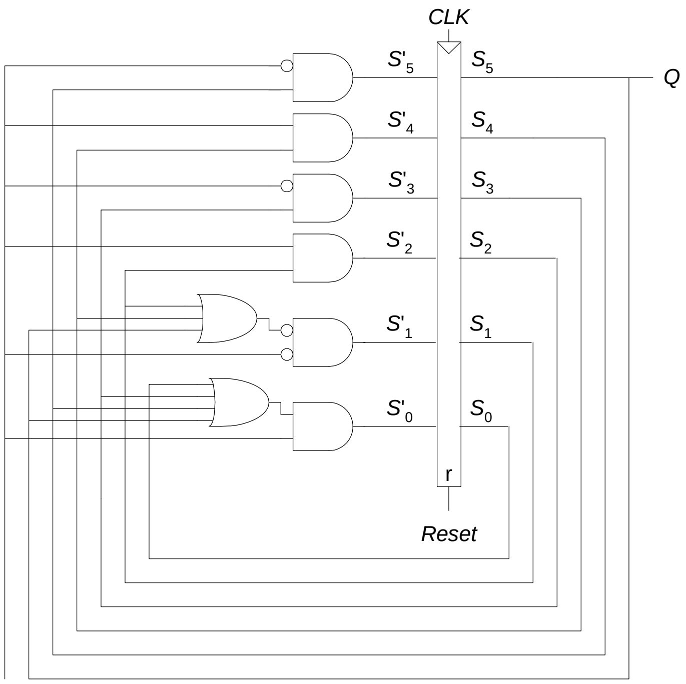


FIGURE 3.16 Finite state machine hardware for Question 3.1

Question 3.2

The FSM should output the value of *A* until after the first 1 is received. It then should output the inverse of *A*. For example, the 8-bit two's complement of the number 6 (00000110) is (11111010). Starting from the least significant bit on the far right, the two's complement is created by outputting the same value of the input until the first 1 is reached. Thus, the two least significant bits of the two's complement number are "10". Then the remaining bits are inverted, making the complete number 11111010.

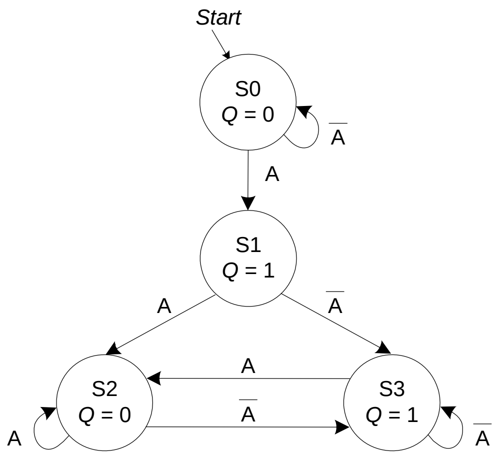


FIGURE 3.17 State transition diagram for Question 3.2

current state $s_{1:0}$	input a	next state $s'_{1:0}$
00	0	00
00	1	01
01	0	11
01	1	10
10	0	11
10	1	10
11	0	11
11	1	10

TABLE 3.22 State transition table for Question 3.2

$$S'_1 = S_1 + S_0$$
$$S'_0 = A \oplus (S_1 + S_0)$$

$$Q = S_0$$

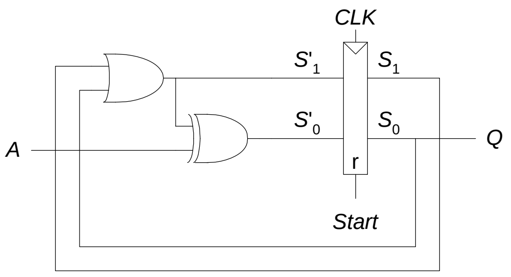


FIGURE 3.18 Finite state machine hardware for Question 3.2

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 3.4

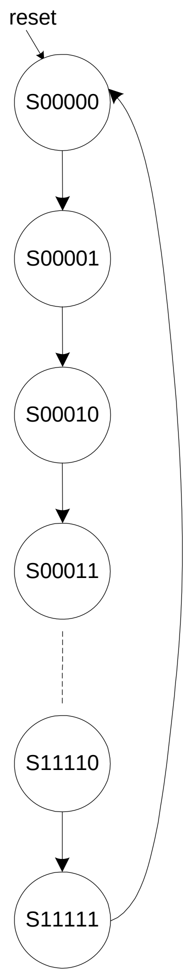


FIGURE 3.19 State transition diagram for Question 3.4

current state $S_{4:0}$	next state $S'_{4:0}$
00000	00001
00001	00010

TABLE 3.23 State transition table for Question 3.4

current state $S_{4:0}$	next state $S'_{4:0}$
00010	00011
00011	00100
00100	00101
...	...
11110	11111
11111	00000

TABLE 3.23 State transition table for Question 3.4

$$S'_4 = S_4 \oplus S_3 S_2 S_1 S_0$$

$$S'_3 = S_3 \oplus S_2 S_1 S_0$$

$$S'_2 = S_2 \oplus S_1 S_0$$

$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = \overline{S_0}$$

$$Q_{4:0} = S_{4:0}$$

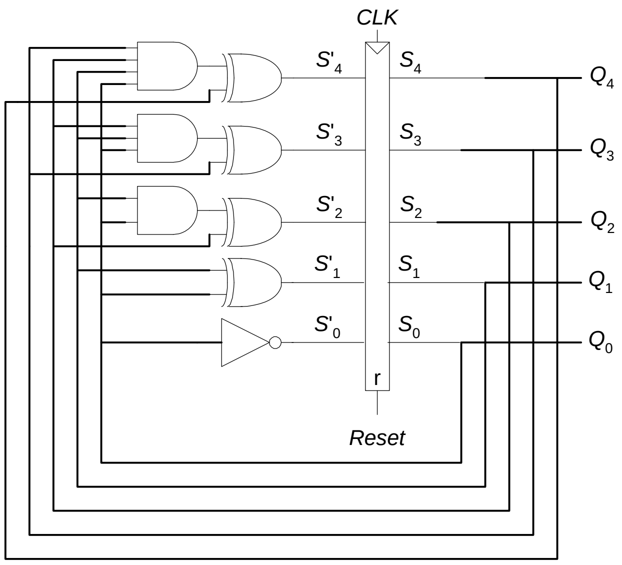


FIGURE 3.20 Finite state machine hardware for Question 3.4

Question 3.5

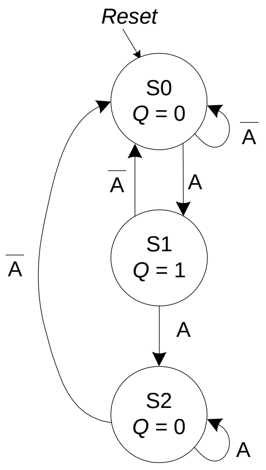


FIGURE 3.21 State transition diagram for edge detector circuit of Question 3.5

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

TABLE 3.24 State transition table for Question 3.5

$$S'_1 = AS_1$$

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

$$Q = S_1$$

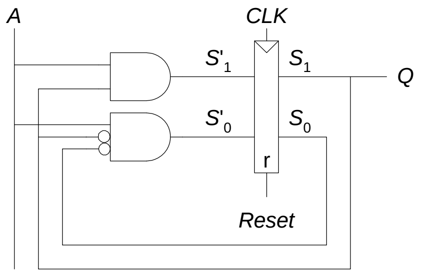


FIGURE 3.22 Finite state machine hardware for Question 3.5

Question 3.6

Pipelining divides a block of combinational logic into *N* stages, with a register between each stage. Pipelining increases throughput, the number of tasks that can be completed in a given amount of time. Ideally, pipelining increases throughput by a factor of *N*. But because of the following three reasons, the speedup is usually less than N: (1) The combinational logic usually cannot be divided into N equal stages. (2) Adding registers between stages adds delay called the *sequencing overhead*, the time it takes to get the signal into and out of the register, $t_{\text{setup}} + t_{pcq}$. (3) The pipeline is not always operating at full capacity: at the beginning of execution, it takes time to fill up the pipeline, and at the end it takes time to drain the pipeline. However, pipelining offers significant speedup at the cost of little extra hardware.

Question 3.7

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

Question 3.8

We use a divide-by-three counter (see Example 3.6 on page 155 of the textbook) with A as the clock input followed by a *negative edge-triggered* flip-flop, which samples the input, D, on the negative or falling edge of the clock, or in this case, A. The output is the output of the divide-by-three counter, S_0, OR the output of the negative edge-triggered flip-flop, N1. Figure 3.24 shows the waveforms of the internal signals, S_0 and N1.

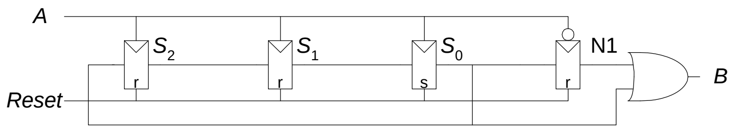


FIGURE 3.23 Hardware for Question 3.8

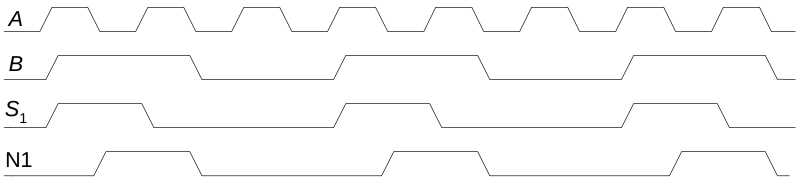


FIGURE 3.24 Waveforms for Question 3.8

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_{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 $T_c + t_{cd\_\text{BUF}} - (t_{pcq} + t_{\text{setup}})$.