Chapter 8

Exercise 8.1: A minimum clock width of 5 ns and a latch delay of 2 ns are considered in this problem.

(a) The minimum delay of the combinational network (t_p) considered in this problem is obtained by the following equation

$$t_p + 2ns \geq 5ns \rightarrow t_p \geq 3ns$$

(b) If the delay of the combinational network can decrease by 30% and the latch delay can decrease by 10%, the maximum clock pulse width (T_{max}) is calculated by the following equation:

$$0.7t_p + 0.9 \times 2ns = 0.7 \times 3 + 1.8 = T_{max} \rightarrow T_{max} = 3.9ns$$

Exercise 8.3

(a) There are no races because of the nonoverlapping nature of the two clocks. This assures that in each clock cycle only one state change occurs.

(b) The number of states depends on the mode of operation of the network. We consider two modes.

i) The clock period of the system corresponds to one of the phases (say phase 1). In this case, the gated latch loaded during the phase two clock acts just as a temporary buffer to prevent races. The state register corresponds to the register loaded during phase 1. Consequently, the number of states is 2^n. The division of the combinational network into two can, in some cases, make sense because of implementation restrictions. For example, if the combinational network is implemented using pass transistors, there is a limitation on the number of them that can be connected in series; in such a case the division in two networks might help.

ii) The clock period is the time between both clocks (phase 1 and phase 2). This results in a system that has a smaller period than that in (i) (and, therefore, in a faster system). In this case each register stores part of the state and the number of potential states is 2^{2n}, but these states cannot be utilized in general. The state can be described by two components s_1 (stored in register 1) and s_2 (stored in register 2). Each component changes in alternate clock cycles.

The output is a combination of two components z_1 and z_2 expressed as:

$$\begin{aligned} z_1(t) &= G_1(s_1(t), x_1(t)) \\ z_2(t) &= G_2(s_2(t), x_2(t)) \end{aligned}$$

(c) Using the first model, the implementation of the system of Exercise 8.4 (Figure 8.40 of the textbook) is straightforward if we only replace the D-type cells for two gated latches in a master/slave configuration, as shown on page 203 of the textbook. Doing this modification, the new design will behave the same way as the one shown in Exercise 8.4, with a slower clock. One way to improve this design would be to split the combinational network, reducing the propagation delay between latches, as shown in Figure 8.1. Observe that one more latch is used in this case.

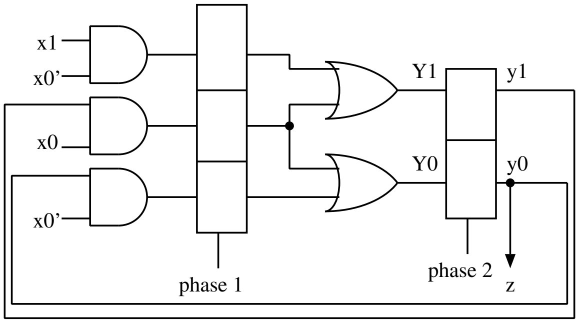


Figure 8.1: Redesign of system in Exercise 8.4 using latches - Exercise 8.3

Exercise 8.5

The state diagram of the pattern recognizer for the sequence 0101011 is shown in Figure 8.2 and has seven states. Each state was labeled with the sequence that it “recognizes”. We encode these states using three state variables (y_2, y_1, y_0) so that the state assignment of state S_i is the radix-2 representation of i. The correspondence between the state assignment and sequence that it detects is shown in the next table.

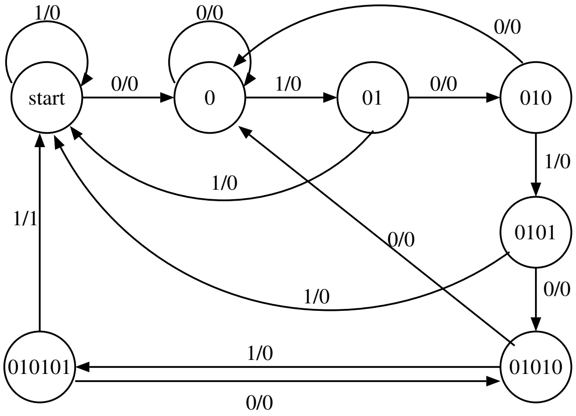


Figure 8.2: State diagram for system in Exercise 8.5

State	Sequence
S_0	start
S_1	0
S_2	01
S_3	010
S_4	0101
S_5	01010
S_6	010101

The state and transition table is:

	PS	Input	
	$y_2y_1y_0$	$x = 0$	$x = 1$
S_0	000	001,0	000,0
S_1	001	001,0	010,0
S_2	010	011,0	000,0
S_3	011	001,0	100,0
S_4	100	101,0	000,0
S_5	101	001,0	110,0
S_6	110	101,0	000,1
		$Y_2Y_1Y_0, z$	

The switching expressions for the next state and output are:

$$\begin{aligned} Y_2 &= y_2 y_0' x' + y_2 y_0 x + y_1 y_0 x \\ Y_1 &= y_1' y_0 x + y_2' y_1 y_0' x' \\ Y_0 &= x' \\ z &= y_2 y_1 x \end{aligned}$$

These expressions are implemented by AND-OR networks and the state is stored in a 3-bit register. The corresponding sequential network is shown in Figure 8.3.

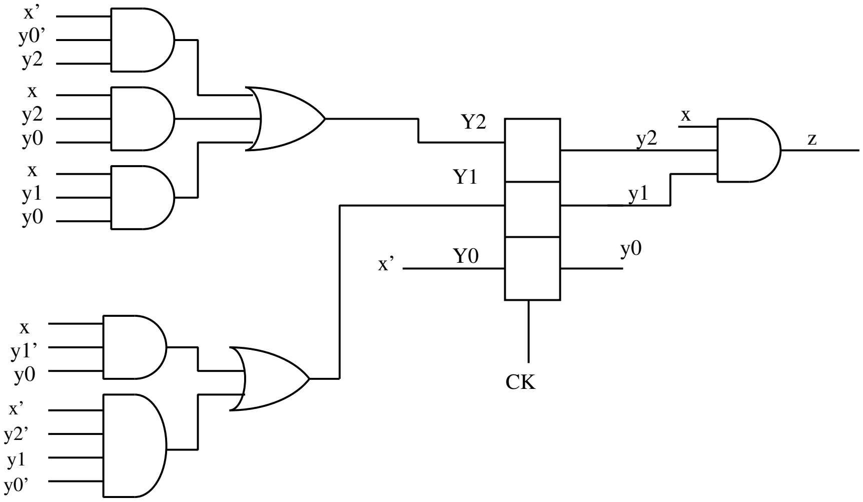
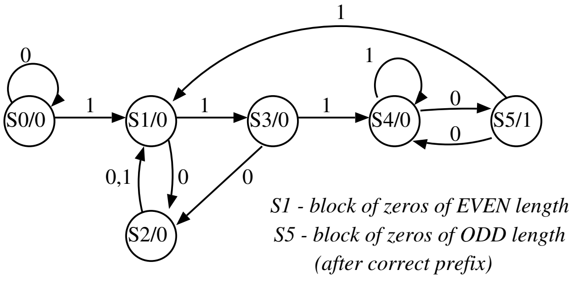


Figure 8.3: Network for Exercise 8.5

Exercise 8.7

The state diagram for this system is presented in Figure 8.4. State S_1 represents a 1 followed by a EVEN block of zeros, and state S_5 indicates that the system received the required sequence. The corresponding state table is:



S1 - block of zeros of EVEN length
S5 - block of zeros of ODD length
(after correct prefix)

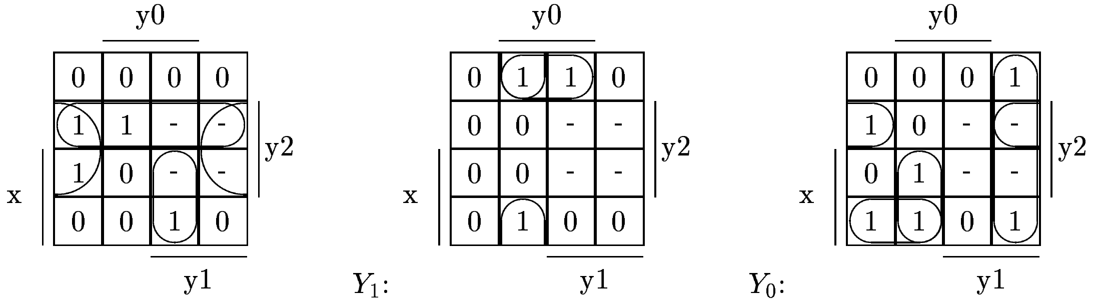
Figure 8.4: State diagram for Exercise 8.7

	Input		
PS	$x = 0$	$x = 1$	
S_0	S_0	S_1	0
S_1	S_2	S_3	0
S_2	S_1	S_1	0
S_3	S_2	S_4	0
S_4	S_5	S_4	0
S_5	S_4	S_1	1
	NS		output(z)

We use the state code i (in binary) for the state S_i. The PS is represented by the 3-bit vector $(y_2y_1y_0)$ and the NS by the vector $(Y_2Y_1Y_0)$. The output switching expression is:

$$z = y_2y_0$$

The K-maps for the next state bits are shown next.



Y_2:

0	0	0	0
1	1	-	-
1	0	-	-
0	0	1	0

Y_1:

0	1	1	0
0	0	-	-
0	0	-	-
0	1	0	0

Y_0:

0	0	0	1
1	0	-	-
0	1	-	-
1	1	0	1

The switching expression for Y_2,Y_1, and Y_0 are:

$$\begin{aligned} Y_2 &= y_2y_0' + x'y_2 + xy_1y_0 \\ Y_1 &= y_2'y_1'y_0 + x'y_2'y_0 \\ Y_0 &= xy_2'y_1' + xy_1'y_0 + y_1y_0' + x'y_2y_0' \end{aligned}$$

The corresponding sequential network is shown in Figure 8.5.

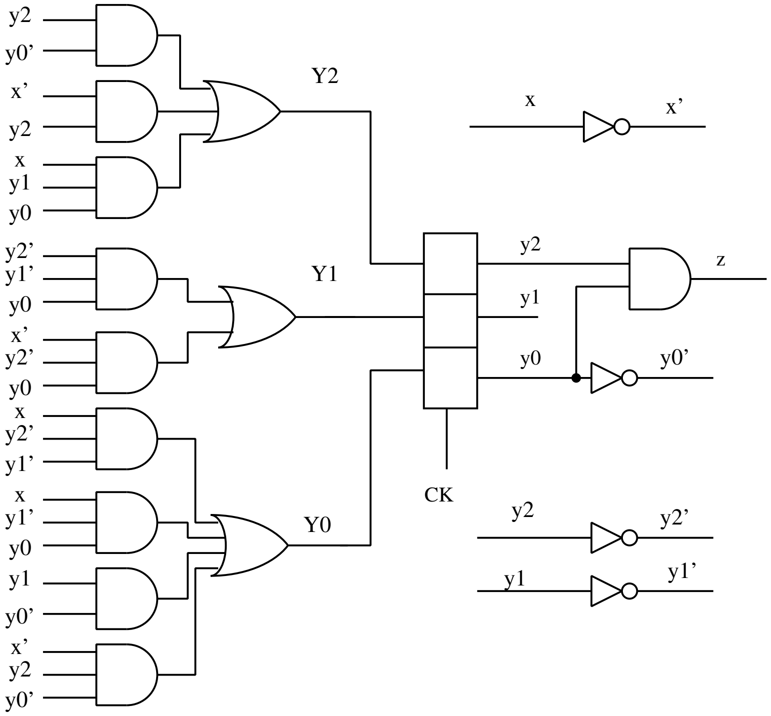


Figure 8.5: Network for Exercise 8.7

Exercise 8.9:

Direct application of K-maps is not possible for this problem. To design this network it is important to decompose it into smaller parts, as shown in Figure 8.6. The NOTBCD module detects when the input code, or the stored minimum value is not a valid BCD code. The input vector is represented by the vector $\underline{x} = (x_3, x_2, x_1, x_0)$, and the state vector is $\underline{y} = (y_3, y_2, y_1, y_0)$. Input x (y) is not a valid BCD code if $x > 9$ ($y > 9$). This condition is represented by the expression $x\_is\_not\_BCD = x_3x_2 + x_3x_1$ ($y\_is\_not\_BCD = y_3y_2 + y_3y_1$). The NOTBCD module is implemented by an expression that combines both cases:

$$NOTBCD = x\_is\_not\_BCD + y\_is\_not\_BCD = x_3x_2 + x_3x_1 + y_3y_2 + y_3y_1$$

The MIN module is specified as:
Inputs: $x, y \in \{0, 1, 2, \cdots, 15\}$
Output: $z \in \{0, 1, 2, \cdots, 15\}$
Function:

$$z = \begin{cases} x & \text{if } x < y \\ y & \text{otherwise} \end{cases}$$

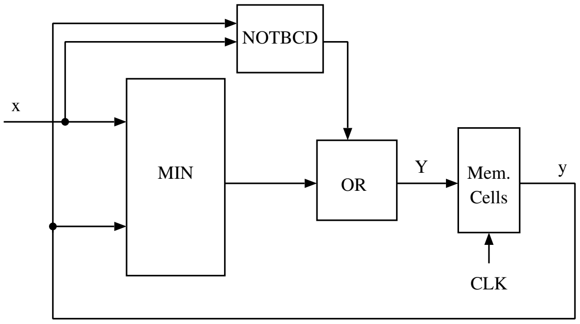


Figure 8.6: Network for Exercise 8.9

The MIN module may be implemented as an iterative array, comparing bits from most-significant to least-significant. Each bit slice has two "data" inputs x_i and y_i and two "carry" inputs e_i (equal) and s_i ($x_j < y_j$ for some $j < i$), and the outputs: m_i, e_{i-1}, and s_{i-1}. The folllowing expressions are used for each output:

$$\begin{aligned} e_{i-1} &= e_i(x_i \oplus y_i)' \\ s_{i-1} &= s_i + (x_i'y_i)e_i \\ m_i &= s_{i-1}x_i + s_{i-1}'y_i \\ e_4 &= 1 \\ s_4 &= 0 \end{aligned}$$

The final gate network for the system is shown in Figure 8.7.

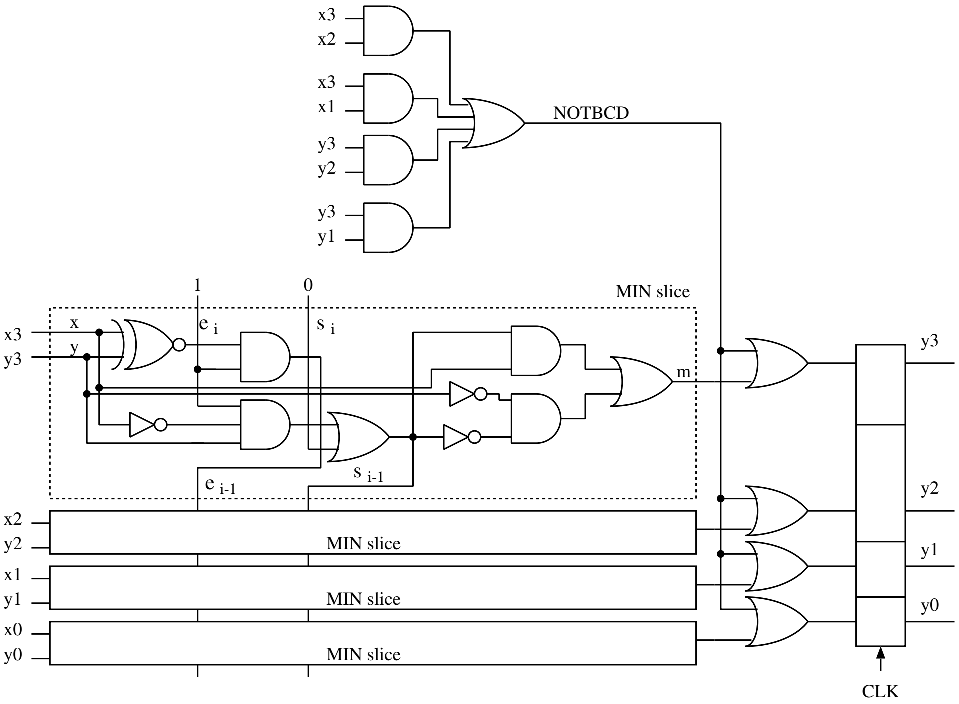


Figure 8.7: Minimum detector for Exercise 8.9

Exercise 8.11

To perform subtraction it is necessary to have a representation of negative integers, since the result can be negative. Several of these representations are given in Chapter 10. Here we simplify the problem by assuming that the result is positive.

Given two n-bit integers represented in the binary number system by $\underline{x} = (x_{n-1}, \ldots, x_0)$ and $\underline{y} = (y_{n-1}, \ldots, y_0)$, and the result represented by $(s_{n-1}, \ldots, s_0)$, the function to be performed serially in each bit position, for addition and subtraction, is described by the following tables:

			Addition	
x_i	y_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

			Subtraction	
x_i	y_i	b_i	b_{i+1}	s_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

where c_i is the carry-in bit at position i, and b_i is the borrow-in bit.

Let the variable k indicate the operation to be performed as follows:

$$k = \begin{cases} 1 & \text{for } x + y \\ 0 & \text{for } x - y \end{cases}$$

Since we want to combine both operations in the same module, let us make $c = b$. A switching expression for the result s_i is:

$$\begin{aligned} s_i &= x_i \oplus x_i \oplus c_i \quad \text{for } i = 1, \ldots, n-1 \\ s_n &= c_n \end{aligned}$$

providing that $a_n = b_n = 0$.

A switching expression for c_{i+1} is

$$c_{i+1} = c_i y_i + (k \oplus x_i)'(c_i + y_i)$$

The initial condition $c_0 = 0$ is set with $INIT = 1$.

The sequential network is given in Figure 8.8.

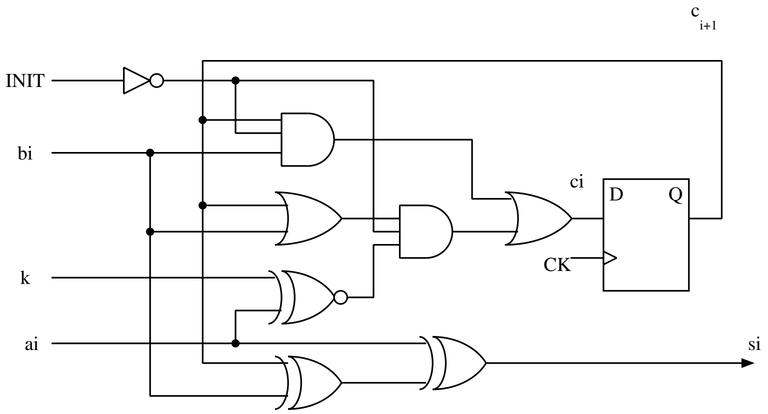


Figure 8.8: Adder/subtractor, Exercise 8.11

Another implementation of this sequential adder/subtractor is possible using complementation and addition. This approach will become clear after the discussion in Chapter 10.

Exercise 8.13

In order to simplify the design of this system, we decompose the state ($S(t)$) into two components: the number of digits being inserted ($S_1(t)$), and the correctness of the input ($S_2(t)$). The first component is implemented by a modulo-4 counter with states $S_1(t) \in \{0,1,2,3\}$, and the second component by a two-state machine with states $S_2(t) \in \{Y, N\}$. The output is generated as a function os these states.

The initial state is $S(t) = (0, Y)$, and the combination $(0, N)$ never happens. The state transition table for the two-component system is:

	Inputs			
$S_1(t)S_2(t)$	$x = 0$	$x = 5$	$x = 6$	*others*
0Y	1Y	1N	1N	1N
1Y	2N	2Y	2N	2N
1N	2N	2N	2N	2N
2Y	3N	3N	3Y	3N
2N	3N	3N	3N	3N
3-	0Y	0Y	0Y	0Y
	$S_1(t+1)S_2(t+1)$			

The state diagram for the lock is shown in Figure 8.9. We consider that the counter state $S_1(t)$ is represented by the vector (c_1, c_0), and $S_2(t)$ assumes the values 1 for Y and 0 for N. The expression for the next state of the two-state component and the outputs are:

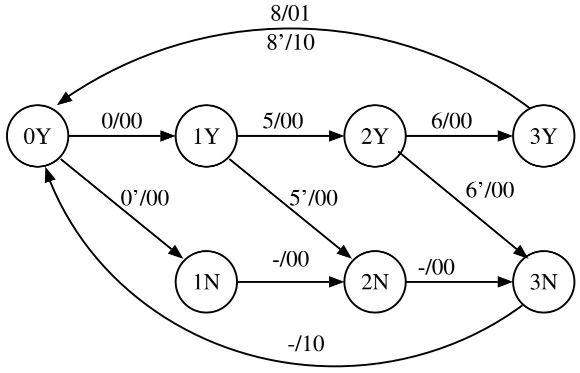


Figure 8.9: State diagram for lock – Exercise 8.13

$$S_2(t+1) \quad = \quad d_0 c_1' c_0' S_2(t) + d_5 c_1' c_0 S_2(t) + d_6 c_1 c_0' S_2(t) + c_1 c_0$$

where $d_i = 1$ when $x = i$. Thus,

$$\begin{aligned} S_2(t+1) &= S_2(t)(x_3' x_2' x_1' x_0' c_1' c_0' + x_3' x_2 x_1' x_0 c_1' c_0 + x_3' x_2 x_1 x_0' c_1 c_0') + c_1 c_0 \\ z_2(t) &= c_1 c_0 (S_2(t)(x_3 x_0')' + S_2(t)') = c_1 c_0 (x_3' + x_0 + S_2(t)') \\ z_1(t) &= c_1 c_0 S_2(t) x_3 x_0' \end{aligned}$$

The gate network that implements the locker is shown in Figure 8.10.

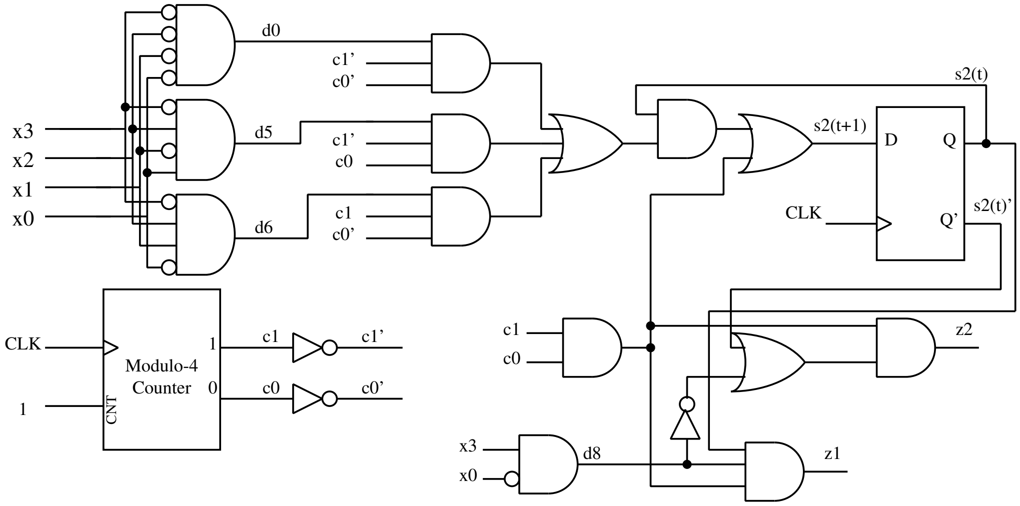


Figure 8.10: Network for lock in Exercise 8.13

Exercise 8.15 We make use of the concepts presented for the D-latch network on page 200 of the textbook and the master/slave configuration shown in Figure 8.11 of the textbook. The general network configuration is shown in Figure 8.11. The slave cell is a D-type latch, as discussed in the book. A combinational network must be designed to activate the set/reset inputs of an SR latch. C represents the clock input.

The table for the combinational network is shown next:

C	T	Q	S	R
0	-	-	0	0
1	0	0	0	-
1	0	1	-	0
1	1	0	1	0
1	1	1	0	1

It is easy to see that

$$\begin{aligned} S &= Q'CT \\ R &= QCT \end{aligned}$$

The combinational network is composed of two 3-input AND gates.

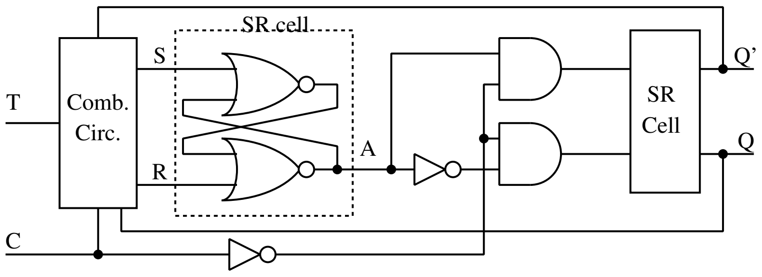


Figure 8.11: Gate implementation of a master-slave T flip-flop

Exercise 8.17

The expression for the flip-flop inputs are

$$J_A = xQ_C \qquad J_B = Q'_A \qquad J_C = xQ_B$$

$$K_A = xQ'_B \qquad K_B = Q_A \qquad K_C = x'Q'_B$$

From the characteristic expressions of the JK flip-flop we get the following expressions for the transition functions:

$$\begin{aligned} Q_A(t+1) &= Q_A(t)(x' + Q_B(t)) + xQ'_A(t)Q_C(t) \\ Q_B(t+1) &= Q_B(t)Q'_A(t) + Q'_B(t)Q'_A(t) \\ Q_C(t+1) &= Q_C(t)(Q_B(t) + x) + Q'_C(t)xQ_B(t) \\ z &= Q'_C(t) \end{aligned}$$

The corresponding transition table is

PS	Input		Input		Output
$Q_AQ_BQ_C$	$x = 0$	$x = 1$	$x = 0$	$x = 1$	z
000	00,10,01	01,10,00	010	010	1
001	00,10,01	11,10,00	010	111	0
010	00,10,00	00,10,10	010	011	1
011	00,10,00	10,10,10	011	111	0
100	00,01,01	01,01,00	100	000	1
101	00,01,01	11,01,00	100	001	0
110	00,01,00	00,01,10	100	101	1
111	00,01,00	10,01,10	101	101	0
	J_AK_A, J_BK_B, J_CK_C		NS		

To get a high-level description we define the following code:

$Q_AQ_BQ_C$	state
000	S_0
001	S_1
010	S_2
011	S_3
100	S_4
101	S_5
110	S_6
111	S_7

The resulting state table is

PS	Input		Output
	$x = 0$	$x = 1$	z
S_0	S_2	S_2	1
S_1	S_2	S_7	0
S_2	S_2	S_3	1
S_3	S_3	S_7	0
S_4	S_4	S_0	1
S_5	S_4	S_1	0
S_6	S_4	S_5	1
S_7	S_5	S_5	0
	NS		

The state diagram is shown in Figure 8.12.

A timing diagram can be obtained from the following input/output sequence pairs (the input is arbitrary):

$x(t)$	0	1	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	1	1	
$s(t)$	0	2	3	3	7	5	4	4	0	2	3	7	5	4	0	2	2	2	3	7
z	1	1	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1	0	1

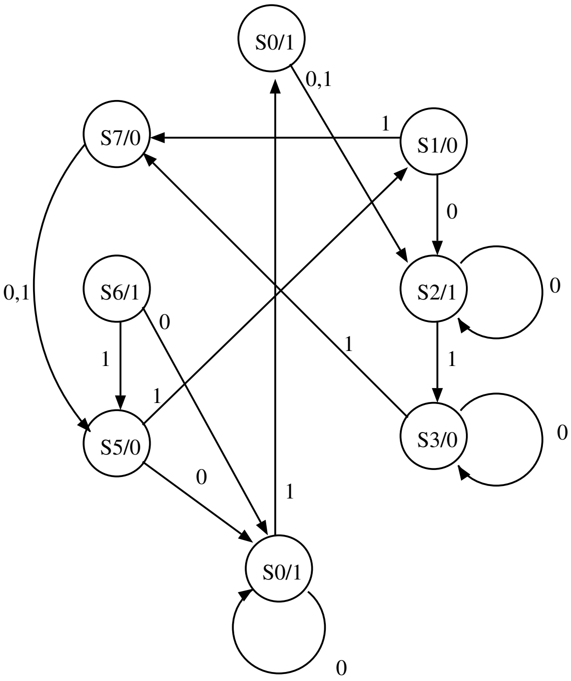


Figure 8.12: State diagram for Exercise 8.17

Exercise 8.19

The expressions for the flip-flop inputs and for the output are

$$\begin{array}{lcl@{\qquad\qquad}lcl} J_A & = & 1 & K_A & = & 1 \\ J_B & = & Q'_C & K_B & = & 1 \\ J_C & = & Q_B & K_C & = & 1 \\ z & = & Q'_A Q'_B Q'_C & & & \end{array}$$

The sequential network does not have any input; therefore, the state register changes in each clock pulse depending only on the previous state. The transition table is

PS	FF inputs			NS	Output
$Q_A(t)Q_B(t)Q_C(t)$	$J_A(t)K_A(t)$	$J_B(t)K_B(t)$	$J_C(t)K_C(t)$	$Q_A(t+1)Q_B(t+1)Q_C(t+1)$	$z(t)$
000	11	11	01	110	1
001	11	01	01	100	0
010	11	11	11	101	0
011	11	01	11	100	0
100	11	11	01	010	0
101	11	01	01	000	0
110	11	11	11	001	0
111	11	01	11	000	0

Let us define the following encoding:

$Q_AQ_BQ_C$	state
000	A
001	B
010	C
011	D
100	E
101	F
110	G
111	H

The resulting transition table is

PS	NS	z
A	G	1
B	E	0
C	F	0
D	E	0
E	C	0
F	A	0
G	B	0
H	A	0

Let us try to reduce the number of states:

$$P_1 = \underset{2}{\frac{1}{(A)}} \qquad\qquad \underset{2\quad 2\quad 2\quad 2\quad 1\quad 2\quad 1}{\frac{2}{(B,\ \ C,\ \ D,\ \ E,\ \ F,\ \ G,\ \ H)}}$$

$$P_2 = \begin{array}{cccccccc} 1 & \multicolumn{2}{c}{2} & \multicolumn{5}{c}{3} \\ \hline (A) & (F, & H) & (B, & C, & D, & E, & G) \\ 3 & 1 & 1 & 3 & 2 & 3 & 3 & 3 \end{array}$$

$$P_3 = \begin{array}{cccccccc} 1 & \multicolumn{2}{c}{2} & 3 & \multicolumn{4}{c}{4} \\ \hline (A) & (F, & H) & (C) & (B, & D, & E, & G) \\ 4 & 1 & 1 & 2 & 4 & 4 & 3 & 4 \end{array}$$

$$P_4 = \begin{array}{cccccccc} 1 & \multicolumn{2}{c}{2} & 3 & 4 & \multicolumn{3}{c}{5} \\ \hline (A) & (F, & H) & (C) & (E) & (B, & D, & G) \\ 5 & 1 & 1 & 2 & 3 & 4 & 4 & 5 \end{array}$$

$$P_5 = \begin{array}{cccccccc} 1 & \multicolumn{2}{c}{2} & 3 & 4 & 5 & \multicolumn{2}{c}{6} \\ \hline (A) & (F, & H) & (C) & (E) & (G) & (B, & D) \\ 5 & 1 & 1 & 2 & 3 & 6 & 4 & 4 \end{array}$$

$$P_6 = P_5 = \{(A), (E), (C), (F, H), (G), (B, D)\}$$

The reduced state table is

PS	NS	z
$(A) = S_0$	S_1	1
$(E) = S_1$	S_2	0
$(C) = S_2$	S_3	0
$(F, H) = S_3$	S_0	0
$(G) = S_4$	S_5	0
$(B, D) = S_5$	S_1	0

The state diagram is shown in Figure 8.13.

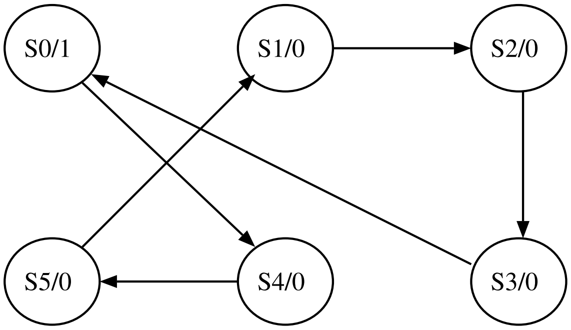


Figure 8.13: State diagram for Exercise 8.19

The network in its equilibrium state produces a 1 every six clock pulses, that is, it implements a modulo-6 frequency divider.

Exercise 8.21
The expressions for the flip-flop inputs and for the output are

$$
\begin{aligned}
S_2 &= xQ_1' + xQ_2'Q_0' \\
R_2 &= Q_2Q_1 \\
J_1 &= Q_0(x' + Q_2) \\
K_1 &= Q_0 + Q_2 \\
T_0 &= (x' + Q_2 + Q_1)(Q_2' + Q_1' + Q_0)(x + Q_2' + Q_1 + Q_0') \\
z &= (Q_2' \oplus Q_1)(x'Q_0') + Q_1'(x'Q_0) + (Q_2 \oplus Q_1)(xQ_0') + Q_1(xQ_0)
\end{aligned}
$$

The transition table is

PS	Input						Input	
$Q_2Q_1Q_0$	$x = 0$			$x = 1$			$x = 0$	$x = 1$
000	00	00	1	10	00	0	001,1	100,0
001	00	11	1	10	01	0	010,1	101,0
010	00	00	1	10	00	1	011,0	111,1
011	00	11	1	00	01	1	000,0	000,1
100	00	01	1	10	01	1	101,0	101,1
101	00	11	0	10	11	1	111,1	110,0
110	01	01	0	01	01	0	000,1	000,0
111	01	11	1	01	11	1	000,0	000,1
	S_2R_2 J_1K_1 T_0						NS,z	

Let us define the following encoding:

$Q_2Q_1Q_0$	
000	A
001	B
010	C
011	D
100	E
101	F
110	G
111	H

The resulting state table is

PS	Input	
	$x = 0$	$x = 1$
A	B,1	E,0
B	C,1	F,0
C	D,0	H,1
D	A,0	A,1
E	F,0	F,1
F	H,1	G,0
G	A,1	A,0
H	A,0	A,1
	NS,z	

Let us try to reduce the number of states.

P_1	group 1	group 2
	(A,B,F,G)	(C,D,E,H)
0	1 2 2 1	2 1 1 1
1	2 1 1 1	2 1 1 1

P_2	group 1	group 2	group 3	group 4	group 5
	(A)	(B,F)	(G)	(C)	(D,E,H)
0	2	4 5	1	5	1 2 1
1	5	2 3	1	5	1 2 1

P_3	group 1	group 2	group 3	group 4	group 5	group 6	group 7
	(A)	(B)	(F)	(G)	(C)	(E)	(D,H)
0	2	5	7	1	7	3	1 1
1	6	3	4	1	7	3	1 1

$$P_4 = P_3 = \{(A), (B), (F), (G), (C), (E), (D, H)\}$$

The reduced state table is

PS	Input	
	$x = 0$	$x = 1$
$(A) \equiv S_0$	$S_1, 1$	$S_4, 0$
$(B) \equiv S_1$	$S_2, 1$	$S_5, 0$
$(C) \equiv S_2$	$S_3, 0$	$S_3, 1$
$(D, H) \equiv S_3$	$S_0, 0$	$S_0, 1$
$(E) \equiv S_4$	$S_5, 0$	$S_5, 1$
$(F) \equiv S_5$	$S_3, 1$	$S_6, 0$
$(G) \equiv S_6$	$S_0, 1$	$S_0, 0$
	NS,z	

The state diagram is shown in Figure 8.14.

We now show that the network performs a serial conversion from BCD to Excess-3. Assume that the initial state is S_0 and that the BCD digit (x_3, x_2, x_1, x_0) is applied with the least significant bit (x_0) first. From the state diagram (following the corresponding paths) we get the following table (since after a sequence of length four the state is again S_0, we only consider sequences of that length):

$x_3x_2x_1x_0$	$z_3z_2z_0$
0000	0011
0001	0100
0010	0101
0011	0110
0100	0111
0101	1000
0110	1001
0111	1010
1000	1011
1001	1100

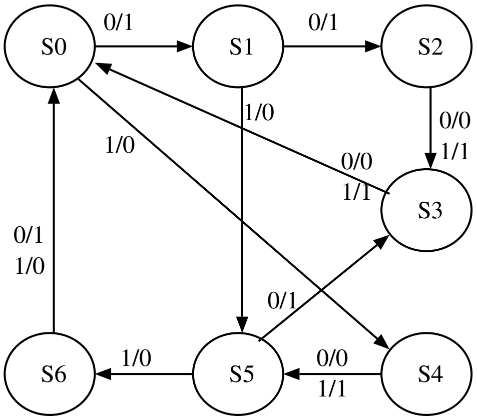


Figure 8.14: State diagram for Exercise 8.21

This corresponds to the desired converter. A timing diagram is

x	0	0	0	0	0	0	1	0	1	0	0	1	1	1	0	0
State	S_0	S_1	S_2	S_3	S_0	S_1	S_2	S_3	S_0	S_4	S_5	S_3	S_0	S_4	S_5	S_3
z	1	1	0	0	1	1	1	0	0	0	1	1	0	1	1	0

Exercise 8.23

To recognize a sequence with two consecutive 1's followed by one 0 we need to distinguish between the following cases:

S_2 :	$x(t-2) = x(t-1) = 1$
S_1 :	Not S_2 and $x(t-1) = 1$
S_0 :	None of the above

The corresponding state table is

PS	Input	
	$x = 0$	$x = 1$
S_0	$S_0, 0$	$S_1, 0$
S_1	$S_0, 0$	$S_2, 0$
S_2	$S_0, 1$	$S_2, 0$
	NS,z	

To implement these states we need at least two flip-flops. Let us define the following state assignment:

	Q_2Q_1
S_0	00
S_1	01
S_2	10

The resulting state table is

PS	Input	
Q_2Q_1	$x = 0$	$x = 1$
00	00,0	01,0
01	00,0	10,0
10	00,1	10,0
	NS,z	

Since the excitation function of a JK flip-flop is

PS	NS	
	0	1
0	0-	1-
1	-1	-0
	JK	

the inputs J_1, K_1, J_2, and K_2 to the JK flip-flops are

PS	Input	
Q_2Q_1	$x = 0$	$x = 1$
00	0-,0-	0-,1-
01	0-,-1	1-,-1
10	-1,0-	-0,0-
	J_2K_2, J_1K_1	

Switching expressions for these flip-flop inputs are

$$J_2 = Q_1 x \qquad J_1 = Q_2' x$$

$$K_2 = x' \qquad K_1 = 1$$

An expression for the output is

$$z = Q_2 x'$$

The sequential network is shown in Figure 8.15.

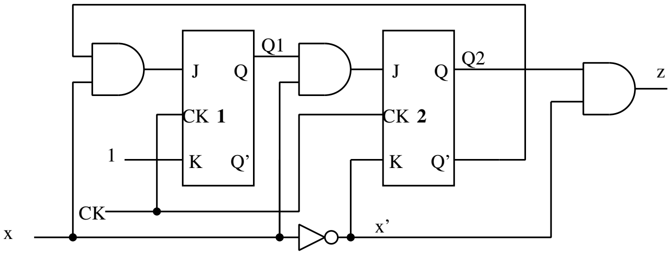


Figure 8.15: Network for Exercise 8.23

- **Exercise 8.25**

We have to store the last input to be able to recognize the sequence 11; therefore, we need:

$$\begin{aligned} S_0: \quad x(t-1) &= 0 \\ S_1: \quad x(t-1) &= 1 \end{aligned}$$

The state table is

PS	Input	
	$x=0$	$x=1$
S_0	$S_0, 0$	$S_1, 0$
S_1	$S_0, 0$	$S_1, 1$
	NS,z	

Coding S_0 as 0 and S_1 as 1, the resulting state table is

PS	Input	
Q	$x=0$	$x=1$
0	0,0	1,0
1	0,0	1,1
	NS,z	

We only need one JK flip-flop. Since its excitation function is

PS	NS	
	0	1
1	0-	1-
1	-1	-0
	JK	

the inputs are

$$\begin{aligned} J &= x \\ K &= x' \end{aligned}$$

The output is described by

$$z = Qx$$

The sequential network is shown in Figure 8.16.

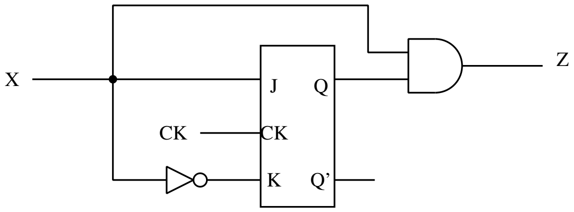


Figure 8.16: Network for Exercise 8.25

Exercise 8.27

a) To design a cyclic counter with the output sequence $0, 1, 3, 7, 6, 4, 0, 1, \ldots$ we need six states. In this first part, we select the coding so that the output corresponds to the state. The state table is

PS	Input	
	$x = 0$	$x = 1$
000	000	001
001	001	011
011	011	111
100	100	000
110	110	100
111	111	110
	$NS = z$	

Since the excitation function of a JK flip-flop is

PS	NS	
	0	1
0	0-	1-
1	-1	-0
	JK	

we get the following flip-flop inputs

PS	Input					
$Q_2Q_1Q_0$	$x = 0$			$x = 1$		
000	0-	0-	0-	0-	0-	1-
001	0-	0-	-0	0-	1-	-0
011	0-	-0	-0	1-	-0	-0
100	-0	0-	0-	-1	0-	0-
110	-0	-0	0-	-0	-1	0-
111	-0	-0	-0	-0	-0	-1
	J_2K_2 J_1K_1 J_0K_0					

From K-maps we obtain

$$J_2 = xQ_1 \qquad K_2 = xQ_1'$$

$$J_1 = xQ_0 \qquad K_1 = xQ_0'$$

$$J_0 = xQ_2' \qquad K_0 = xQ_2$$

The sequential network is shown in Figure 8.17. Recall that the output $\underline{z} = (z_2, z_1, z_0)$ corresponds to the state vector (Q_2, Q_1, Q_0).

b) In this second case, the state table is

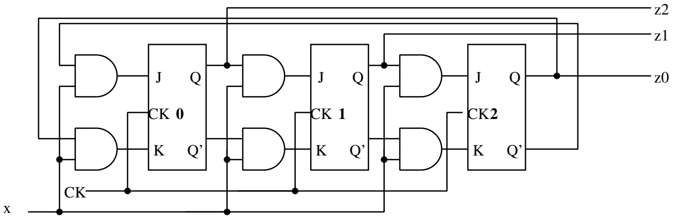


Figure 8.17: Network for Exercise 8.27

PS	Input		
	$x=0$	$x=1$	
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_3	3
S_3	S_3	S_4	7
S_4	S_4	S_5	6
S_5	S_5	S_0	4
	NS		Output

with the following encoding for the state:

	$Q_2Q_1Q_0$
S_0	000
S_1	001
S_2	010
S_3	011
S_4	100
S_5	101

Using the previously given excitation function for a JK flip-flop, the flip-flops inputs are

PS	Input						Output
$Q_2Q_1Q_0$	$x=0$			$x=1$			$z(t)$
000	0-	0-	0-	0-	0-	1-	000
001	0-	0-	-0	0-	1-	-1	001
010	0-	-0	0-	0-	-0	1-	011
011	0-	-0	-0	1-	-1	-1	111
100	-0	0-	0-	-0	0-	1-	110
101	-0	0-	-0	-1	0-	-1	100
	J_2K_2	J_1K_1	J_0K_0				

From K-maps we obtain

$$J_2 = xQ_1Q_0 \qquad K_2 = xQ_0$$

$$J_1 = xQ_2'Q_0 \qquad K_1 = xQ_0$$

$$J_0 = x \qquad K_0 = x$$

For the output,

$$\begin{aligned} z_2 = Q_2 &\quad + \quad Q_1Q_0 \\ z_1 = Q_1 &\quad + \quad Q_2Q_0' \\ z_0 = Q_1 &\quad + \quad Q_2'Q_0 \end{aligned}$$

The sequential network is shown in Figure 8.18.

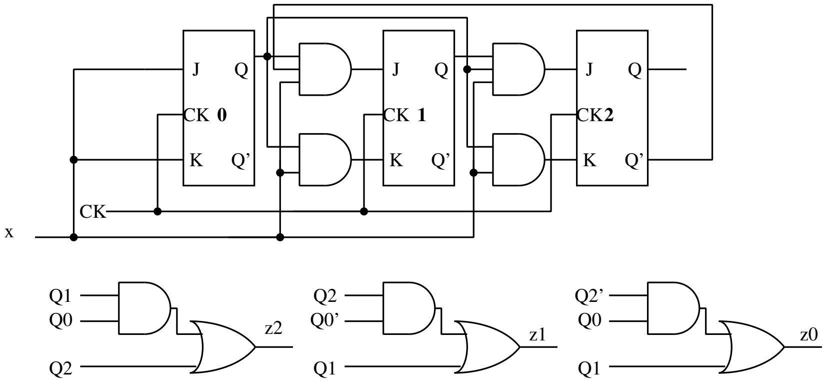


Figure 8.18: Network for Exercise 8.27 (b)

A comparison of the networks of Figures 8.17 and 8.18 indicates that solution (a) results in a simpler network.

Exercise 8.29

To recognize the sequence $x(t-3,t) = 0101$ or 0110 we need to distinguish among the following cases:

S_4 :	$x(t-3,t-1) = 011$
S_3 :	$x(t-3,t-1) = 010$
S_2 :	$x(t-2,t-1) = 01$
S_1 :	Not S_3 and $x(t-1) = 0$
S_0 :	None of the above

The corresponding state table is

PS	Input	
	$x=0$	$x=1$
S_0	$S_1,0$	$S_0,0$
S_1	$S_1,0$	$S_2,0$
S_2	$S_3,0$	$S_4,0$
S_3	$S_1,0$	$S_2,1$
S_4	$S_1,1$	$S_0,0$
	NS,z	

Assigning to each state the binary value of its subindex the resulting state table is

PS	Input	
$Q_2Q_1Q_0$	$x=0$	$x=1$
000	001,0	000,0
001	001,0	010,0
010	011,0	100,0
011	001,0	010,0
100	001,1	000,0
	NS,z	

Since the excitation function of a JK flip-flop is

PS	NS	
	0	1
0	0-	1-
1	-1	-0
	JK	

we determine the inputs J_2, K_2, J_1, K_1, J_1, and K_1 to be

PS	Input					
$Q_2Q_1Q_0$	$x=0$			$x=1$		
000	0-	0-	1-	0-	0-	0-
001	0-	0-	-0	0-	1-	-1
010	0-	-0	1-	1-	-1	0-
011	0-	-1	-0	0-	-0	-1
100	-1	0-	1-	-1	0-	0-
	J_2K_2	J_1K_1	J_0K_0			

Using K-maps we get the following switching expressions:

$$
\begin{aligned}
J_2 &= xQ_1Q_0' \\
K_2 &= 1 \\
J_1 &= xQ_0 \\
K_1 &= xQ_0' + x'Q_0 \\
J_0 &= x' \\
K_0 &= x
\end{aligned}
$$

The output expression is:

$$z = x'Q_2 + xQ_1Q_0$$

The sequential network is shown in Figure 8.19

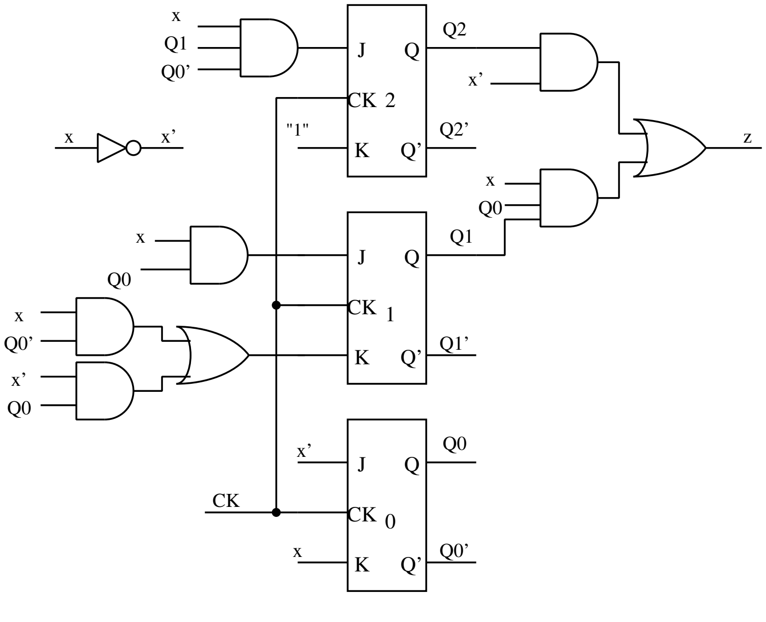
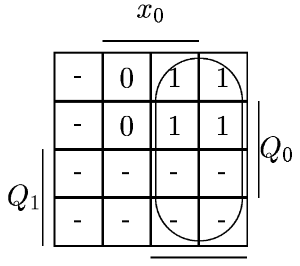


Figure 8.19: Network for Exercise 8.29

Exercise 8.31 To implement Example 8.9 using JK flip-flops we use the following state transition table and output, that already includes the appropriate input functions for the flip-flops:

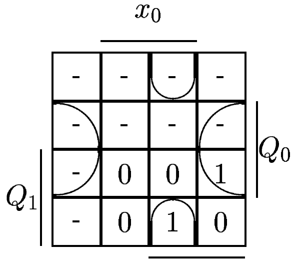
PS	Input			Input		
Q_1Q_0	01	10	11	01	10	11
00	01,0	10,1	10,0	0-1-	1-0-	1-0-
01	00,0	11,1	11,0	0–1	1–0	1–0
10	11,0	10,0	00,1	-01-	-00-	-10-
11	10,0	00,0	11,1	-0-1	-1-1	-0-0
	NS, z			$J_1K_1J_0K_0$		

The Kmaps for the combinational circuit that activates the JK inputs of the flip-flops are:



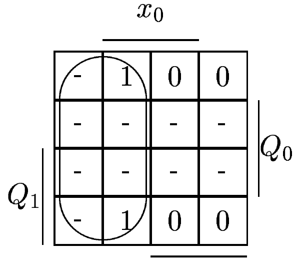
-	0	1	1
-	0	1	1
-	-	-	-
-	-	-	-

J_1:



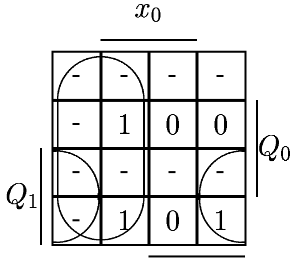
-	-	-	-
-	-	-	-
-	0	0	1
-	0	1	0

K_1:



-	1	0	0
-	-	-	-
-	-	-	-
-	1	0	0

J_0:



-	-	-	-
-	1	0	0
-	-	-	-
-	1	0	1

K_0:

which result in the following expressions:

$$\begin{aligned} J_1 &= x_1 \\ k_1 &= x_1x_0Q_0' + x_0'Q_0 \\ J_0 &= x_1' \\ K_0 &= x_1' + x_0'Q_1 \end{aligned}$$

The gate network for this exercise is shown in Figure 8.20. This network has fewer gates than that of Figure 8.29 of the text.

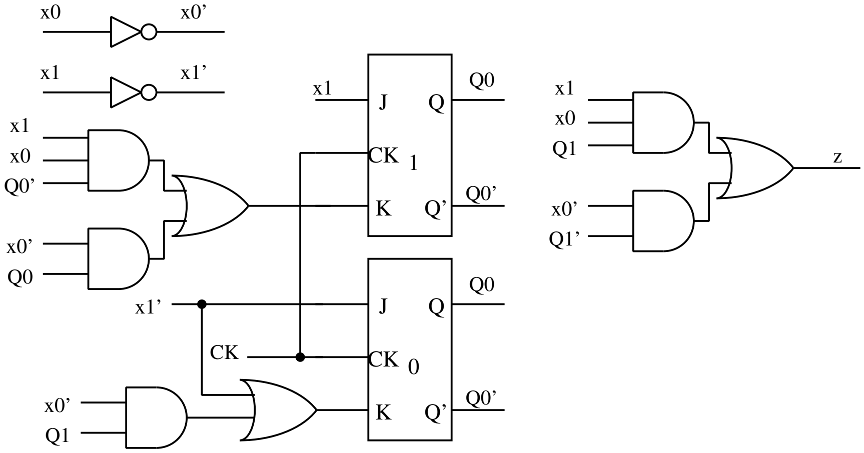


Figure 8.20: Network for Exercise 8.31

Exercise 8.33 Each state is represented by one flip-flop. Consider that the input of these FFs are represented as $NA, NB, NC, ND, NE,$ and NF respectively. The switching expressions for these variables are:

$$\begin{aligned}
NA &= Fx' + Bx \\
NB &= Ax' + Cx \\
NC &= Bx' + Cx \\
ND &= Cx' + Ex \\
NE &= Dx' + Fx \\
NF &= Ex' + Ax
\end{aligned}$$

For the following output encoding

	$z_1 z_0$
a	00
b	01
c	10

the switching expressions for the output are:

$$\begin{aligned}
z_0 &= Dx' + Ax + Cx + E \\
z_1 &= Bx + Dx + F
\end{aligned}$$

The network that corresponds to these expressions is shown in Figure **??**.

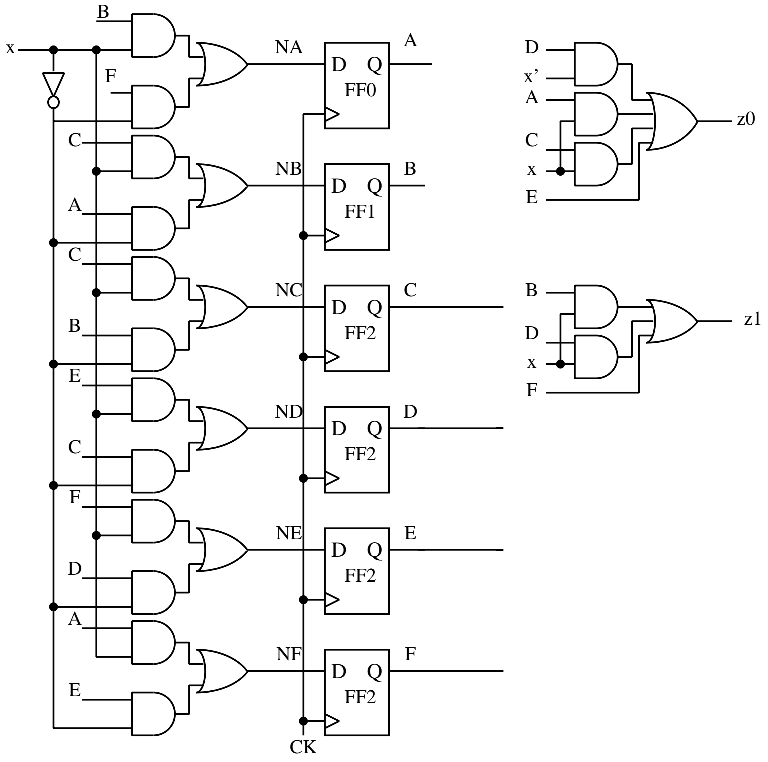


Figure 8.21: Network for Exercise 8.33