Appendix A
Logic Circuits

A.1. The truth table for the COINCIDENCE function is

x_1	x_2	COINCIDENCE
0	0	1
0	1	0
1	0	0
1	1	1

$\text{COINCIDENCE} = \overline{x}_1\overline{x}_2 + x_1x_2 = \overline{(x_1 \oplus x_2)}$

A.2. Proof for identity (a):

$$\begin{aligned}\overline{(a \oplus b)} \oplus c &= \overline{(a \oplus b)}\overline{c} + (a \oplus b)c \\ &= \overline{a}\overline{b}\overline{c} + ab\overline{c} + \overline{a}bc + a\overline{b}c\end{aligned}$$

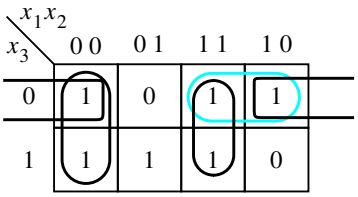
Proof for identity (b):

$$\begin{aligned}x + w\overline{x} &= (x + w)(x + \overline{x}) \\ &= x + w\end{aligned}$$

Proof for identity (c):

$$\begin{aligned}x_1\overline{x}_2 + \overline{x}_2x_3 + x_3\overline{x}_1 &= x_1\overline{x}_2 + \overline{x}_2x_3(x_1 + \overline{x}_1) + x_3\overline{x}_1 \\ &= x_1\overline{x}_2 + x_1\overline{x}_2x_3 + x_3\overline{x}_1\overline{x}_2 + x_3\overline{x}_1 \\ &= x_1\overline{x}_2 + x_3\overline{x}_1\end{aligned}$$

A.3. Using Karnaugh maps get:

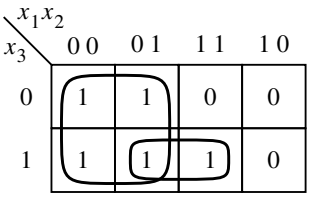


A minimum cost expression for f_1 is

$$f_1 = \overline{x}_1\overline{x}_2 + x_1x_2 + x_1\overline{x}_3$$

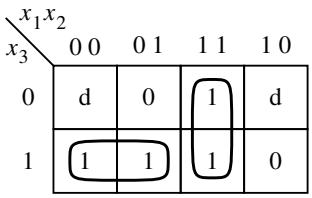
Another expression that has the same cost is

$$f_1 = \overline{x}_1\overline{x}_2 + x_1x_2 + \overline{x}_2\overline{x}_3$$



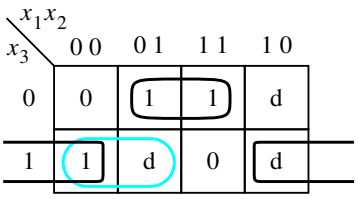
The minimum cost expression for f_2 is

$$f_2 = \overline{x}_1 + x_2x_3$$



The minimum cost expression for f_3 is

$$f_3 = x_1x_2 + \overline{x}_1x_3$$



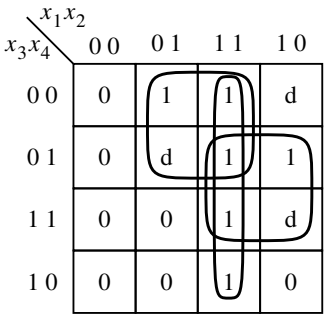
A minimum cost expression for f_4 is

$$f_4 = x_2\overline{x}_3 + \overline{x}_2\overline{x}_3$$

Another expression that has the same cost is

$$f_4 = x_2\overline{x}_3 + \overline{x}_1x_3$$

A.4. The corresponding Karnaugh map is



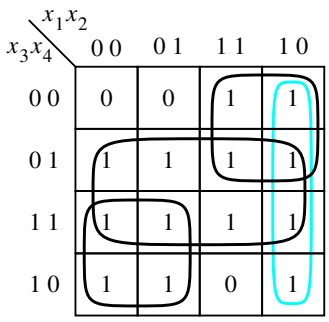
A minimum-cost SOP expression is

$$f = x_2\overline{x}_3 + x_1x_2 + x_1\overline{x}_3$$

Another expression that has the same cost is

$$f = x_2\overline{x}_3 + x_1x_2 + x_1x_4$$

A.5. The Karnaugh map is



(a) The minimum-cost SOP expression is

$$f = x_4 + \overline{x}_1 x_3 + x_1 \overline{x}_3 + x_1 \overline{x}_2$$

(b) The minimum-cost SOP expression for the complement of f is

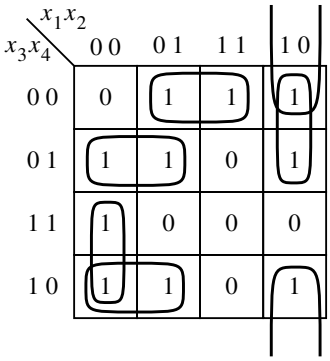
$$\overline{f} = \overline{x}_1 \overline{x}_3 \overline{x}_4 + x_1 x_2 x_3 \overline{x}_4$$

Complementing this expression using de Morgan's rule gives

$$f = (x_1 + x_3 + x_4)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_4)$$

This expression requires 2 OR gates, one AND gate and 9 inputs to gates, for a total cost of 12. The SOP expression requires 3 AND gates, one OR gate and 10 inputs to gates, for a total cost of 14. The apparent conclusion is that for some functions the POS implementation is less expensive than the SOP implementation, and vice versa.

A.6. The corresponding Karnaugh map is



A minimum-cost SOP expression is

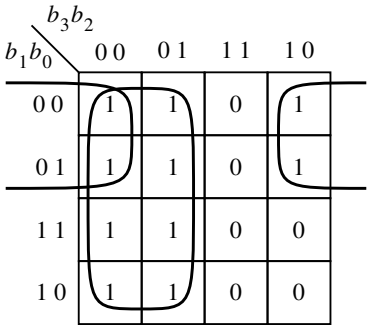
$$f = x_1 \overline{x}_2 \overline{x}_3 + x_2 \overline{x}_3 \overline{x}_4 + \overline{x}_1 \overline{x}_3 x_4 + \overline{x}_1 \overline{x}_2 x_3 + \overline{x}_1 x_3 \overline{x}_4 + x_1 \overline{x}_2 \overline{x}_4$$

The minimum-cost POS expression is

$$f = (x_1 + x_2 + x_3 + x_4)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_4)(\overline{x}_1 + \overline{x}_3 + \overline{x}_4)(\overline{x}_2 + \overline{x}_3 + \overline{x}_4)$$

The cost of the SOP expression is 31, comprising 7 gates and 24 inputs to gates. The cost of the POS expression is 27, comprising 6 gates and 21 inputs. Therefore, the POS expression leads to the minimum-cost implementation.

A.7. The desired function, f, has the value 1 for the first ten rows of the truth table in Figure A.6. The value of f for the remaining six rows is 0. The corresponding Karnaugh map is

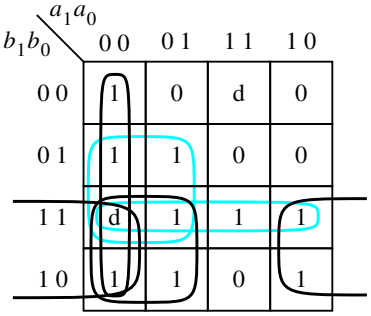


Hence, the expression

$$f = \overline{b}_3 + \overline{b}_1\overline{b}_2$$

describes the required circuit.

A.8. The comparison function, f, is defined by the map



The minimum-cost SOP expression is

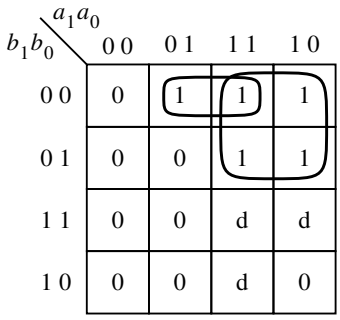
$$f = \overline{a}_1\overline{a}_0 + b_1b_0 + \overline{a}_0b_1 + \overline{a}_1b_0 + \overline{a}_1b_1$$

The minimum-cost POS expression is

$$f = (\overline{a}_1 + b_1)(\overline{a}_0 + b_1 + b_0)(\overline{a}_1 + \overline{a}_0 + b_0)$$

The cost of the SOP expression is 21, comprising 6 gates and 15 inputs to gates. The cost of the POS expression is 15, comprising 4 gates and 11 inputs. This describes the lowest-cost circuit.

A.9. The comparison function, f, is defined by the map



The minimum-cost circuit is specified by the expression

$$f = a_1\overline{b}_1 + a_0\overline{b}_1\overline{b}_0$$

A.10. The associative rule would require that

$$(w \uparrow y) \uparrow z = w \uparrow (y \uparrow z)$$

The left-hand side of this expression gives

$$\begin{aligned}(w \uparrow y) \uparrow z &= \overline{\overline{wy}z} \\ &= wy + \overline{z}\end{aligned}$$

The right-hand side gives

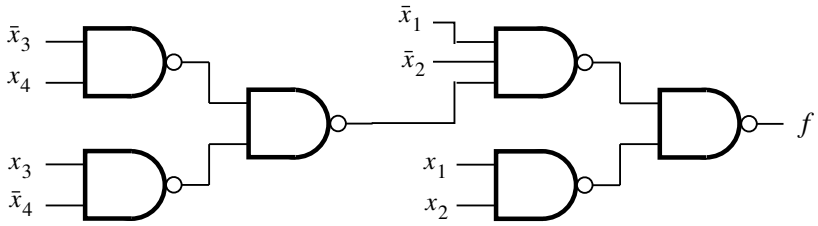
$$\begin{aligned}w \uparrow (y \uparrow z) &= \overline{w\overline{\overline{yz}}} \\ &= \overline{w} + yz\end{aligned}$$

These expressions do not represent the same function. For example, when $w = y = 1$ and $z = 0$, the left hand side is equal to 1 while the right hand side is equal to 0.

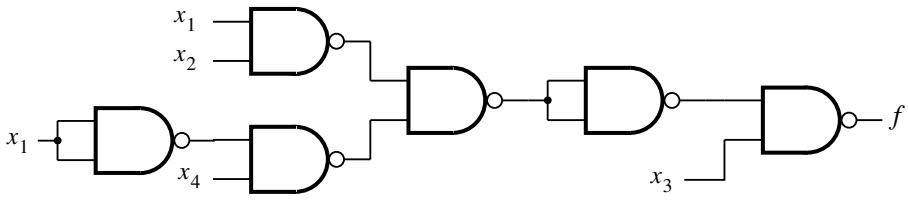
A.11 Simplifying the expression for f into

$$f = x_1x_2 + \overline{x}_1\overline{x}_2(\overline{x}_3x_4 + x_3\overline{x}_4)$$

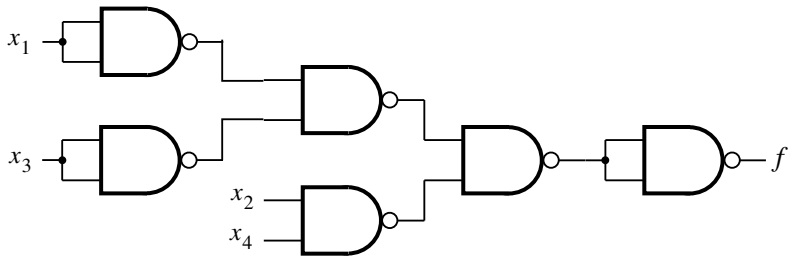
and manipulating it using de Morgan's rule, we can get the following circuit:



A.12. A possible circuit is



A.13. A possible circuit is

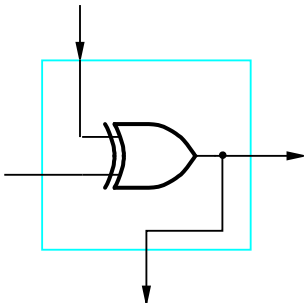


A.14. (a) The sum-of-products expressions are

$$\begin{aligned} f_1 &= a \\ f_2 &= a \oplus b \\ &= \overline{a}b + a\overline{b} \\ f_3 &= a \oplus b \oplus c \\ &= a\overline{b}\overline{c} + \overline{a}b\overline{c} + \overline{a}\overline{b}c + abc \end{aligned}$$

These AND-OR circuits can be implemented using only NAND gates by a direct transformation as explained in Figure A.8. The expressions for f_1, f_2 and f_3 require 0, 3 and 5 gates, respectively, plus 3 gates to invert the input variables a, b and c.

(b) The general block in Figure PA.2b can be implemented as

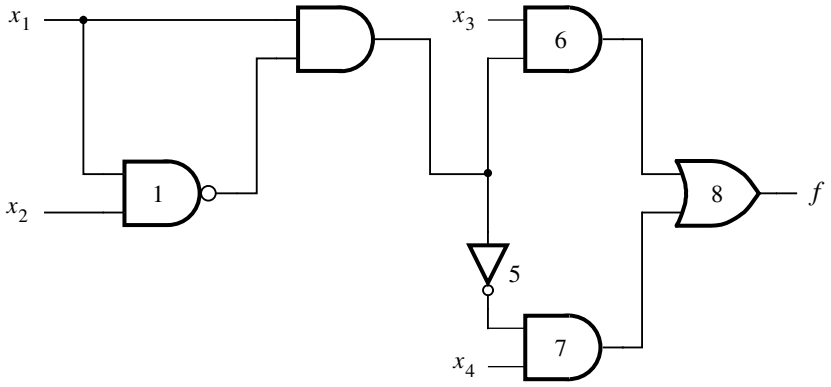


Since the leftmost block need not have an input from the left side, the XOR gate is not needed in that block, and input a is wired directly to the output of the block. Thus, only two XOR gates are needed. They can be implemented with a total of 8 NAND gates (see Problem A.15).

A.16. Consider the a function only. The implementation given in Figure A.37 can be seen to be correct by the following argument. The input to the inverter must be $\overline{a}$. The 2-level NAND network implements a sum-of-products expression for the 0s of the truth table column for a. (See Figure A.8). The 4-input NAND gate accounts for the input valuation $(x_1, x_2, x_3, x_4) = (0, 0, 0, 1)$, and the 3-input NAND gate accounts for the input valuation $(x_1, x_2, x_3, x_4) = (0, 1, 0, 0)$ coupled with the "don't care" entry at $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$. The remaining functions in the given implementation can be verified in the same way.

The AND, OR, NOT implementation follows directly via replacement of the individual NAND gate networks by AND and OR gates as shown in Figure A.8.

A.17. The stuck-at-1 fault at F1 reduces the network to



The stuck-at-0 fault at F2 reduces the network to just a wire that implements

$$f = x_4$$

A.18. As explained in Section A.5.1, in a CMOS circuit the pull-up network implements the function f and the pull-down network implements its complement $\overline{f}$. In our case

$$f = \overline{x}_1\overline{x}_2 + \overline{x}_3\overline{x}_4$$

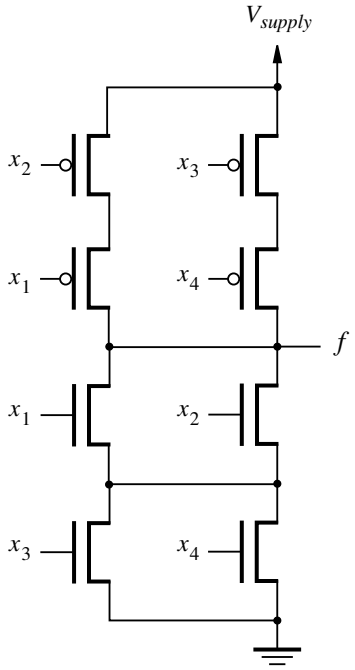
and

$$\overline{f} = (x_1 + x_2)(x_3 + x_4)$$

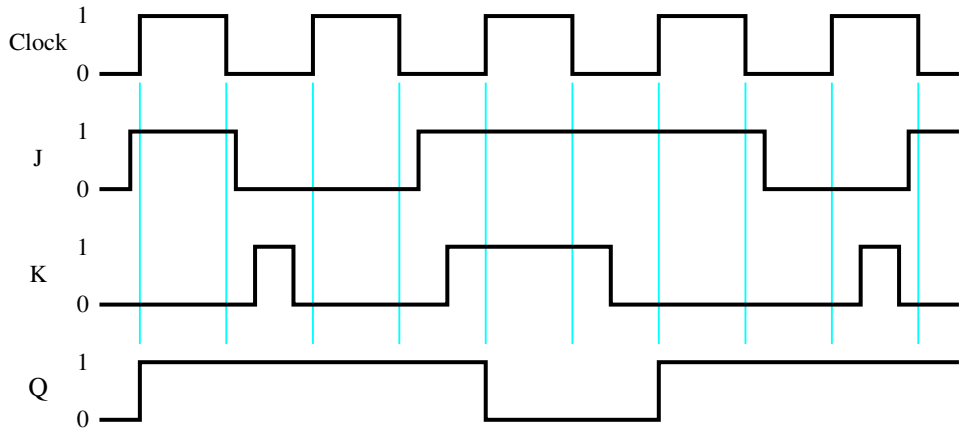
The term $(x_1 + x_2)$ is realized as a parallel connection of NMOS transistors driven by inputs x_1 and x_2. Similarly, the term $(x_3 + x_4)$ is realized as a parallel connection of NMOS transistors driven by inputs x_3 and x_4. These two parallel subcircuits have to be connected in series to realize the product of the two terms.

In the pull-up network, the term $\overline{x}_1\overline{x}_2$ is realized as a series connection of PMOS transistors driven by inputs x_1 and x_2. Similarly, the term $\overline{x}_3\overline{x}_4$ is realized as a series connection of PMOS transistors driven by inputs x_3 and x_4. A parallel connection of these subcircuits realizes $\overline{x}_1\overline{x}_2 + \overline{x}_3\overline{x}_4$.

Therefore, the desired circuit is



A.19. The waveforms are



A.20. The truth table for the NAND latch is

A	B	Q	$\overline{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0/1	1/0

This truth table describes the same behavior as the truth table in Figure A.24b if we let A = $\overline{S}$ and B = $\overline{R}$. The only difference is when A = B = 0 causing Q = $\overline{Q}$ = 1, but this input valuation should not be used in an SR latch. The two NAND gates at the input of the circuit in Figure A.26 provide the required inversion of signals S and R when Clk = 1.

A.21. Point P3 follows changes at D with 1 gate delay, and point P4 follows changes at D with 2 gate delays. If we assume that both P3 and P4 are to be stable at their correct values no later than when the clock goes to 0, then the minimum setup time is 2 gate delays.

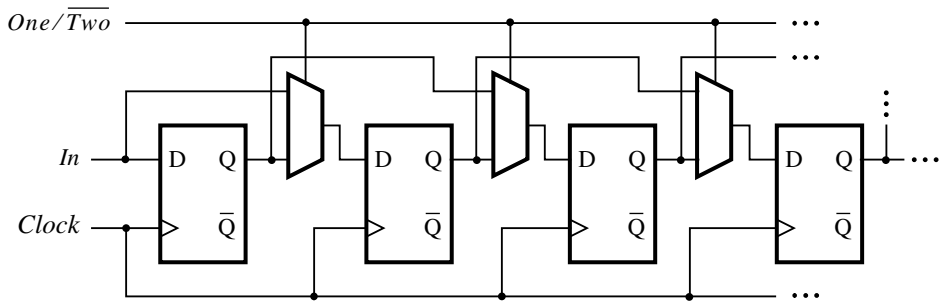
For calculating hold time, the critical case is when P1 is set to 1 as a result of the clock going to 0. This is the case when D = 1 at the clock edge and the flip-flop is to be set into the 1 state. The D line must hold for at least 1 gate delay after the trailing edge of the clock so that the output of gate 2 can get to 1 and maintain the output of gate 1 at 0 for proper operation. Therefore, the hold time is 1 gate delay.

A.22. Using all NOR gates gives the truth table

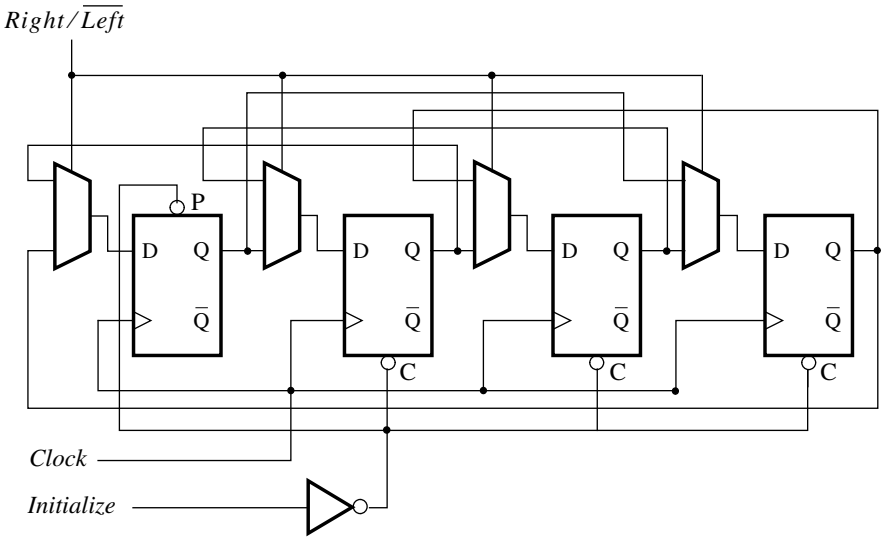
Clk	D	Q($t+1$)
0	0	0
0	1	1
1	x	Q(t)

Therefore, the circuit is a gated D latch which is set to the value of D input when Clk = 0.

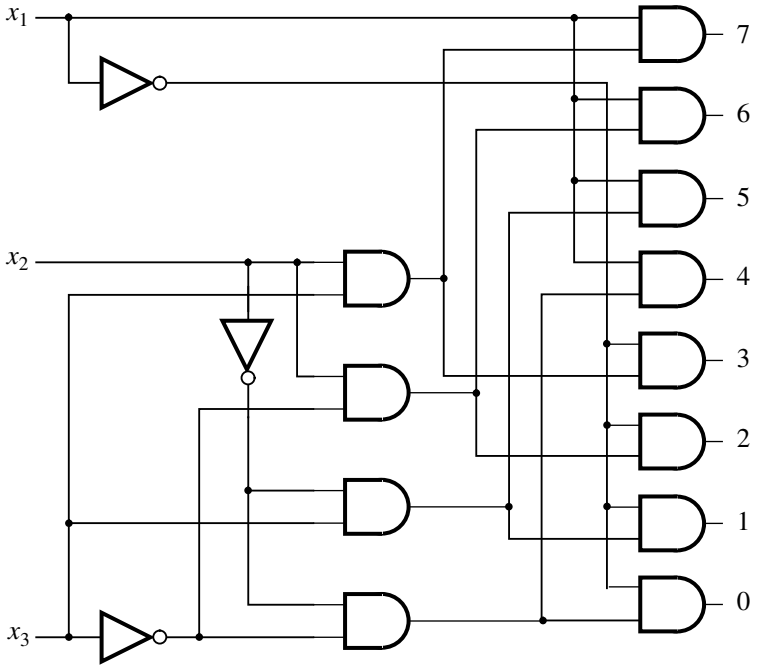
A.23. The modified circuit is



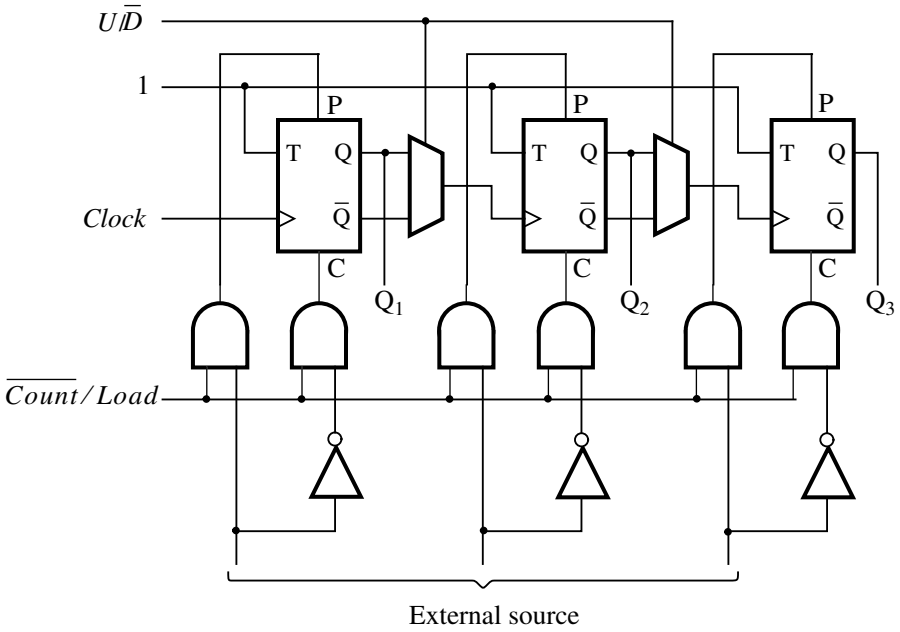
A.24. A possible circuit is



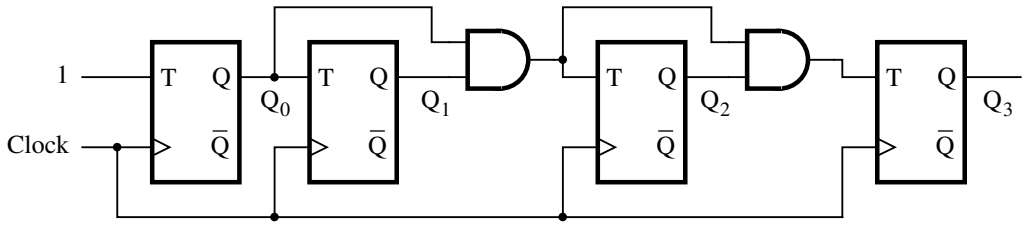
A.25. The 3-input decoder circuit is



A.26. The up/down counter can be implemented as follows:



A.27. A 4-bit synchronous counter can be implemented as



A.28. (a) The truth table for f implemented using an 8-input multiplexer is

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	x_4
0	1	1	0
1	0	0	$\overline{x}_4$
1	0	1	$\overline{x}_4$
1	1	0	0
1	1	1	$\overline{x}_4$

(b) The truth table using a 4-input multiplexer is

x_3	x_4	f
0	0	$\overline{x}_2$
0	1	$\overline{x}_1$
1	0	x_1
1	1	0

A.29. (a) The truth table for f implemented using an 8-input multiplexer is

x_1	x_2	x_3	f
0	0	0	$\overline{x}_4$
0	0	1	$\overline{x}_4$
0	1	0	$\overline{x}_4$
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	x_4

(b) It cannot be done with a single 4-input multiplexer.

A.30. In this case the state-assigned table is

Present state	Next state		Output z
	x = 0	x = 1	
y_2 y_1	Y_2 Y_1	Y_2 Y_1	
1 0	1 1	0 0	0
1 1	0 1	1 0	0
0 1	0 0	1 1	1
0 0	1 0	0 1	0

The next-state and output equations are

$$\begin{aligned} Y_2 &= xy_1 + \overline{y}_1\overline{x} \\ Y_1 &= x \oplus y_2 \\ z &= \overline{y}_2 y_1 \end{aligned}$$

A.31. The state table is

Present state	Next state
S0	S3
S1	S2
S2	S0
S3	S1

The state-assigned table is

Present state	Next state
$y_2 y_1$	$Y_2\ Y_1$
0 0	1 1
0 1	1 0
1 0	0 0
1 1	0 1

The next-state and output equations are

$$\begin{aligned} Y_2 &= \overline{y}_2 \\ Y_1 &= \overline{y}_1\overline{y}_2 + y_1 y_2 \\ z_2 &= y_2 \\ z_1 &= y_1 \end{aligned}$$

A.32. The state table is

Present state	Next state
S0	S1
S1	S2
S2	S3
S3	S4
S4	S5
S5	S0

The state-assigned table is

Present state			Next state		
y_3	y_2	y_1	Y_3	Y_2	Y_1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	d	d	d
1	1	1	d	d	d

The next-state and output equations are

$$
\begin{aligned}
Y_3 &= y_1 y_2 + \overline{y}_1 y_3 \\
Y_2 &= \overline{y}_1 y_2 + y_1 \overline{y}_2 \overline{y}_3 \\
Y_1 &= \overline{y}_1 \\
z_3 &= y_3 \qquad z_2 = y_2 \qquad z_1 = y_1
\end{aligned}
$$

A.33. The state table is

Present state	Next state		Output z
	$x = 0$	$x = 1$	
S0	S1	S0	0
S1	S1	S2	0
S2	S1	S3	0
S3	S1	S0	1

The state-assigned table is

Present state	Next state		Output z
	$x = 0$	$x = 1$	
$y_2\ y_1$	$Y_2\ Y_1$	$Y_2\ Y_1$	
0 0	0 1	0 0	0
0 1	0 1	1 0	0
1 0	0 1	1 1	0
1 1	0 1	0 0	1

The next-state and output equations are

$$\begin{aligned} Y_2 &= x(y_1 \oplus y_2) \\ Y_1 &= \overline{x} + \overline{y}_1 y_2 \\ z &= y_1 y_2 \end{aligned}$$

A.34. The state table is

Present state	Next state		Output z
	$x = 0$	$x = 1$	
S0	S1	S0	0
S1	S1	S2	0
S2	S4	S3	0
S3	S1	S0	1
S4	S1	S2	1

The state-assigned table is

Present state	Next state		Output z
	$x = 0$	$x = 1$	
$y_3\ y_2\ y_1$	$Y_3\ Y_2\ Y_1$	$Y_3\ Y_2\ Y_1$	
0 0 0	0 0 1	0 0 0	0
0 0 1	0 0 1	0 1 0	0
0 1 0	1 0 0	0 1 1	0
0 1 1	0 0 1	0 0 0	1
1 0 0	0 0 1	0 1 0	1

The next-state and output equations are

$$\begin{aligned} Y_3 &= \overline{x}y_2\overline{y}_1 \\ Y_2 &= xy_3 + x\overline{y}_2y_1 + x\overline{y}_1y_2 \\ Y_1 &= \overline{x}y_1 + \overline{y}_2\overline{x} + x\overline{y}_1y_2 \\ z &= y_3 + y_1y_2 \end{aligned}$$