[CS M51A W13] Solutions for Final exam

Date: 03/21/13

TA: Gabriel Pan (pansh@cs.ucla.edu)

Problem 1 (5 points)

Find a minimal state table for the following finite state machine:

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

1. **(4 points)** Derive and show P_1, P_2, $\cdots$, until P_i is equal to the previous partition. You do **NOT** need to show the final table.

   Solution

   $P_1 = (A, C, D, H), (B, E, G), (F)$

   For P_2:

	g1				g2			g3
	A	C	D	H	B	E	G	F
a	2	2	2	2	1	1	1	
b	2	2	2	2	2	2	2	
c	2	3	3	3	2	2	2	
d	1	1	1	1	3	3	3	

   $P_2 = (A), (C, D, H), (B, E, G), (F)$

   For P_3:

	g1	g2			g3			g4
	A	C	D	H	B	E	G	F
a		3	3	3	2	2	2	
b		3	3	3	3	3	3	
c		4	4	4	3	3	3	
d		1	2	1	4	4	4	

$P_3 = (A), (C, H), (D), (B, E, G), (F)$

For P_4:

	g1	g2		g3	g4			g5
	A	C	H	D	B	E	G	F
a		4	4		3	2	2	
b		4	4		4	4	4	
c		5	5		4	4	4	
d		1	1		5	5	5	

$P_4 = (A), (C, H), (D), (B), (E, G), (F)$

For P_5:

	g1	g2		g3	g4	g5		g6
	A	C	H	D	B	E	G	F
a		5	5			2	2	
b		5	5			5	5	
c		6	6			5	5	
d		1	1			6	6	

and we can stop here as $P_5 = P_4$.

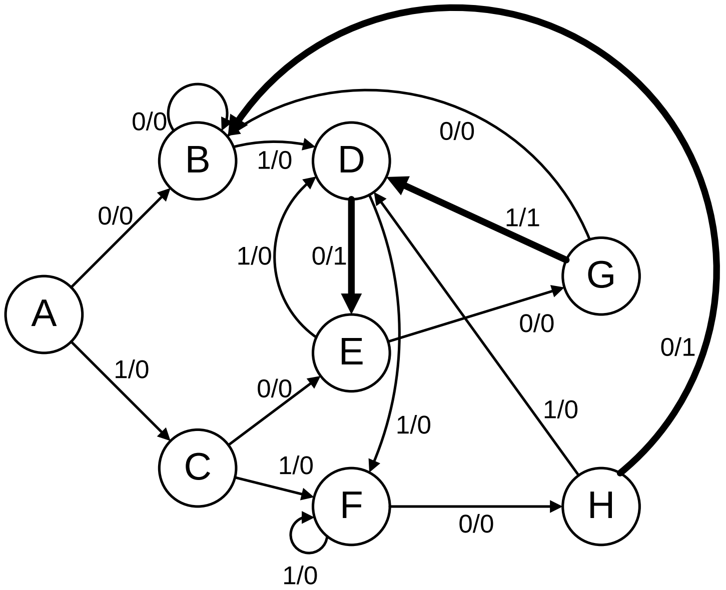
2. **(1 point)** What is the lowest value of integer i that satisfies $P_{i+1} = P_i$?

   Solution Since $P_5 = P_4$, $P_{i+1} = P_i$ for all $i \geq 4$. Therefore, the lowest value of $i = 4$.

Problem 2 (4 points)

The state transition diagram below shows the details of a string detector. The state diagram is minimal, that is, it cannot be reduced any further.

The initial state is A. Each transition is labeled with the triggering input x and output z, as x/z. All transitions that output a 1 are shown in bold.



Which of the following strings will cause the detector to output 1? Choose all that apply.

Solution As the strings are quite long, instead of following the state machine, it is much simpler to identify the substring each state stands for and examine the end of the strings for ones that trigger the output 1.

State A is the initial state S_{init} and it does not stand for any substring. State B is reached through an input 0 from S_{init} so it is S_0. From the same logic, state C is S_1. States D, E and F are reached from S_0 and S_1, they stand for states S_{01}, S_{10}, and S_{11} respectively. Now, states G and H are the length 3 states. Examining the transition arrows, we can set them as states S_{100} and S_{110}.

With the states now named, we can figure out the shortest strings that the detector identifies.

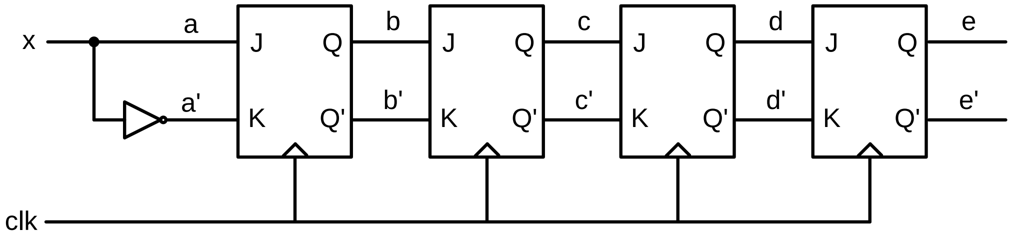
- transition of 0 from D (S_{01}) → string detector outputs 1 for $x(t-2,t) = 010$
- transition of 1 from G (S_{100}) → string detector outputs 1 for $x(t-3,t) = 1001$
- transition of 0 from H (S_{110}) → string detector outputs 1 for $x(t-3,t) = 1100$

Now, by looking at the tail end of the given strings and checking if they match the above strings, we can find the detected strings.

The answers are (b) $\cdots 1100$, (c) $\cdots 1001$, (g) $\cdots 010$ and (i) $\cdots 1001$.

Problem 3 (4 points)

Design a pattern detector that detects the strings 10110, 0110, 0100 and 001 by using a shifting state register. The shifting state register setup is shown below (the JK flip-flop transition function is shown on the last page). Show the two-level AND-OR network that produces the output z. Use only the signals a to e and a' to e' as inputs to the combinational logic. Make sure to simplify the circuit as much as possible.



Solution The shifting state register allows us to keep track of four previous inputs and the current input. To be more specific, $a = x(t)$, $b = x(t-1)$, $c = x(t-2)$, $d = x(t-3)$ and $e = x(t-4)$.

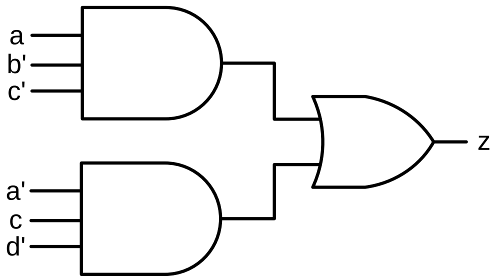
The detector identifies the strings 10010, 0110, 0100 and 001.

- for $x(t-2,t) = 001$ we need $a = 1$, $b = 0$ and $c = 0$ so $z = 1$ when $ab'c' = 1$
- for $x(t-3,t) = 0100$ we need $a = 0$, $b = 0$, $c = 1$ and $d = 0$ so $z = 1$ when $a'b'cd' = 1$
- for $x(t-3,t) = 0110$ we need $a = 0$, $b = 1$, $c = 1$ and $d = 0$ so $z = 1$ when $a'bcd' = 1$
- for $x(t-4,t) = 10110$ we need $a = 0$, $b = 1$, $c = 1$, $d = 0$ and $e = 1$ so $z = 1$ when $a'bcd'e = 1$

Combining these expressions, we get:

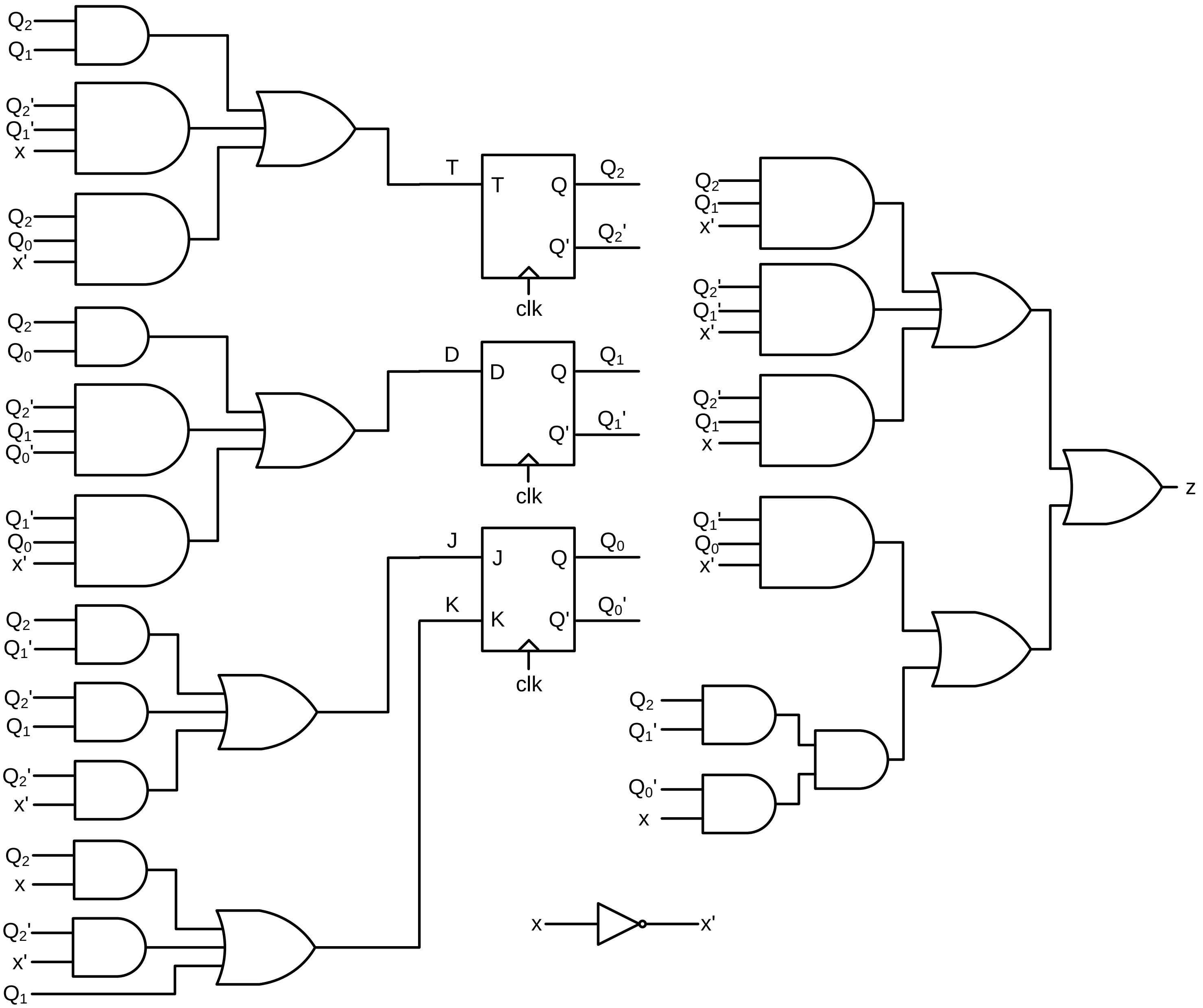
$$\begin{aligned} z &= ab'c' + a'b'cd' + a'bcd' + a'bcd'e \\ &= ab'c' + a'cd'(b'+b) + a'bcd'e \\ &= ab'c' + a'cd' + a'bcd'e \\ &= ab'c' + a'cd'(1+be) \\ &= ab'c' + a'cd' \end{aligned}$$

The final combinational circuit is therefore:



Problem 4 (23 points)

Answer the following questions on the sequential system shown below.



1. **(3 points)** Write the expressions for the flip-flop inputs T, D, J, K and the output z.

   Solution These can be derived directly from the gate network.

$$\begin{aligned}
T &= Q_2Q_1 + Q_2{}'Q_1{}'x + Q_2Q_0x' \\
D &= Q_2Q_0 + Q_2{}'Q_1Q_0{}' + Q_1{}'Q_0x' \\
J &= Q_2Q_1{}' + Q_2{}'Q_1 + Q_2{}'x' \\
K &= Q_2x + Q_2{}'x' + Q_1 \\
z &= (Q_2Q_1x' + Q_2{}'Q_1{}'x' + Q_2{}'Q_1x) + (Q_1{}'Q_0x' + (Q_2Q_1{}')(Q_0{}'x)) \\
&= Q_2Q_1x' + Q_2{}'Q_1{}'x' + Q_2{}'Q_1x + Q_1{}'Q_0x' + Q_2Q_1{}'Q_0{}'x
\end{aligned}$$

2. **(4 points)** Complete the table of flip-flop inputs below.

   Solution From the expressions of T, D, J and K, we can find the 1 terms and fill the corresponding squares in the table.

   This gives us the following table.

PS	input	
$Q_2Q_1Q_0$	$x = 0$	$x = 1$
000	0 0 11	1 0 00
001	0 1 11	1 0 00
010	0 1 11	0 1 11
011	0 0 11	0 0 11
100	0 0 10	0 0 11
101	1 1 10	0 1 11
110	1 0 01	1 0 01
111	1 1 01	1 1 01
	$T\ D\ JK$	

3. **(4 points)** Complete the state transition table below. The JK flip-flop transition function is shown on the last page.

   Solution Using the table from the previous section, we can calculate the next state values. The output values comes from the output expression.

   The completed table is shown.

PS	input		input		input	
$Q_2Q_1Q_0$	$x = 0$	$x = 1$	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	0 0 11	1 0 00	001	100	1	0
001	0 1 11	1 0 00	010	101	1	0
010	0 1 11	0 1 11	011	011	0	1
011	0 0 11	0 0 11	000	000	0	1
100	0 0 10	0 0 11	101	101	0	1
101	1 1 10	0 1 11	011	110	1	0
110	1 0 01	1 0 01	000	000	1	0
111	1 1 01	1 1 01	010	010	1	0
	$T\ D\ JK$		NS		z	

4. **(6 points)** The following timing numbers are given for the system, which we will name System B. It obtains the input signal x from System A, and passes the output signal z to System C. System A and System C are both Moore machines, unlike System B which is a Mealy machine.

Gate type	Fan-in	t_p (ns)
NOT	1	0.05
AND	2	0.15
AND	3	0.20
OR	2	0.20
OR	3	0.35

   All flip-flops share the following characteristics:

t_{su} (ns)	t_p (ns)	t_h (ns)
0.35	0.50	0.18

   The timing values for external systems are $t_{in} = t_p(\text{System A}) = 1.0$ (ns) and $t_{out} = t_{su}(\text{System C}) = 1.5$ (ns). Assume that t_p is the same for both Q and Q' for all flip-flops, but note that the inverted input signal x' goes through an extra NOT gate.

Calculate T_x, T_z and T_y. Other than these three values, are there any signal paths that start from a register and end up at a register input which we need to consider? If any exist, briefly explain the path and obtain T_{other} for that path.

Of T_x, T_y, T_z and T_{other} (if it exists), which has the largest value?

Solution The longest path for $d1^x$ is through NOT → AND3 → OR3.

$$\begin{aligned} d1^x &= t_p(\text{NOT}) + t_p(\text{AND3}) + t_p(\text{OR3}) \\ &= 0.05 + 0.20 + 0.35 \\ &= 0.6(\text{ns}) \end{aligned}$$

With this value obtained, we can write:

$$\begin{aligned} T_x &= t_{in} + d1^x + t_{su}(\text{reg}) \\ &= 1.0 + 0.6 + 0.35 \\ &= 1.95(\text{ns}) \end{aligned}$$

As we have a Mealy machine, we may have different values for $d2^x$ and $d2^y$. The longest path for $d2^y$ is through AND3 → OR3 → OR2.

$$\begin{aligned} d2^y &= t_p(\text{AND3}) + t_p(\text{OR3}) + t_p(\text{OR2}) \\ &= 0.20 + 0.35 + 0.20 \\ &= 0.75(\text{ns}) \end{aligned}$$

With this value obtained, we can write:

$$\begin{aligned} T_z &= t_p(\text{reg}) + d2^y + t_{out} \\ &= 0.5 + 0.75 + 1.5 \\ &= 2.75(\text{ns}) \end{aligned}$$

For T_y, we need the value of $d1_y$ which goes through AND3 → OR3.

$$\begin{aligned} d1^y &= t_p(\text{AND3}) + t_p(\text{OR3}) \\ &= 0.20 + 0.35 \\ &= 0.55(\text{ns}) \end{aligned}$$

With this value obtained, we can write:

$$\begin{aligned} T_y &= t_p(\text{reg}) + d1^y + t_{su}(\text{reg}) \\ &= 0.5 + 0.55 + 0.35 \\ &= 1.4(\text{ns}) \end{aligned}$$

Have we accounted for all possible paths? Unfortunately not, since in the Mealy machine, there is another long path from System A directly to System C via the output combinational circuit in System B. Let us call the timing for this path T_w.

The longest path for $d2^x$ is through NOT → AND3 → OR3 → OR2.

$$\begin{aligned} d2^y &= t_p(\text{NOT}) + t_p(\text{AND3}) + t_p(\text{OR3}) + t_p(\text{OR2}) \\ &= 0.05 + 0.20 + 0.35 + 0.20 \\ &= 0.8(\text{ns}) \end{aligned}$$

With this value obtained, we can write:

$$\begin{aligned} T_w &= t_{in} + d2^x + t_{out} \\ &= 1.0 + 0.8 + 1.5 \\ &= 3.3(\text{ns}) \end{aligned}$$

which is actually the dominant factor for determining the clock period.

The results are:

$$\begin{aligned} T_x &= 1.95(\text{ns}) \\ T_y &= 1.4(\text{ns}) \\ T_z &= 2.75(\text{ns}) \\ T_{\text{other}} &= 3.3(\text{ns}) \end{aligned}$$

The largest value is T_{other}.

5. **(6 points)** Now there is some clock skew affecting the input signal x. System A (which produces the input signal x) has a clock that arrives 0.7 ns earlier than this system (System B). Which of the paths calculated in the previous problem are affected by this? Obtain the new T_{signal} value for all affected paths (if any). Which is now the largest value?

   Solution The timing values affected by this are T_x and T_{other}, as they both are related to the input signal x. The others are not affected.

   And since the clock arrives earlier at System A, it creates extra buffer for the signals to propagate through the system, and the two affected values are reduced by the clock skew amount, which is 0.7 ns.

   Now, the new results are:

$$\begin{aligned} T_x &= 1.25(\text{ns}) \\ T_y &= 1.4(\text{ns}) \\ T_z &= 2.75(\text{ns}) \\ T_{\text{other}} &= 2.6(\text{ns}) \end{aligned}$$

   The largest value is now T_z.

Problem 5 (11 points)

We would like to design an autonomous counter with the following output sequence of cycle length 6:

$$0, 2, 4, 7, 1, 3, 0, 2, 4, 7, 1, 3, 0, 2, ...$$

The initial state is 0. Set the state assignment encoding to be equal to the output, so that the counter is a Moore machine where $z(t) = s(t)$. There is no input signal x to the counter, a single state transition occurs at each clock edge.

1. **(2 points)** Use JK flip-flops for the counter design.

   Complete the table below. The excitation table of the JK flip-flop is:

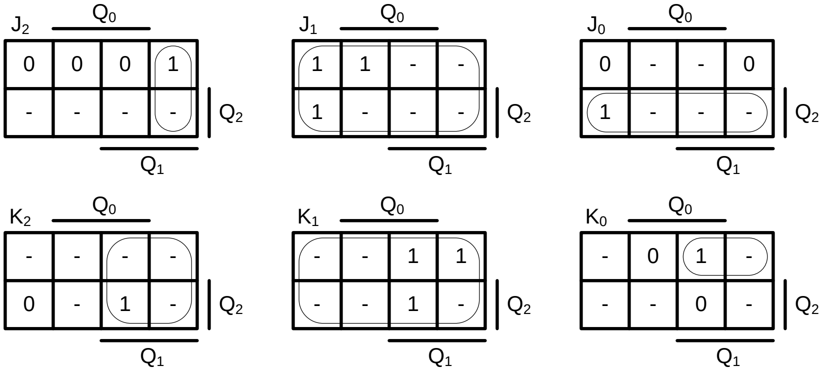
$Q(t)$	$Q(t+1) = 0$	$Q(t+1) = 1$
0	0–	1–
1	–1	–0
	$J(t)K(t)$	

Solution

$PS(Q_2Q_1Q_0)$	$NS(Q_2Q_1Q_0)$	J_2K_2	J_1K_1	J_0K_0
000	010	0–	1–	0–
001	011	0–	1–	–0
010	100	1–	–1	0–
011	000	0–	–1	–1
100	111	–0	1–	1–
101	–	–	–	–
110	–	–	–	–
111	001	–1	–1	–0

2. **(3 points)** Fill in the K-maps for each J_i and K_i, and write the minimal expressions for each.

Solution



J_2

0	0	0	1
-	-	-	-

J_1

1	1	-	-
1	-	-	-

J_0

0	-	-	0
1	-	-	-

K_2

-	-	-	-
0	-	1	-

K_1

-	-	1	1
-	-	1	-

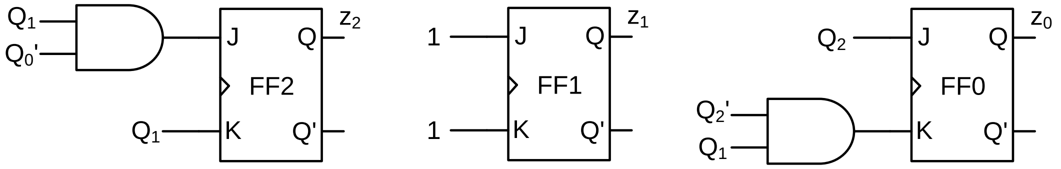
K_0

-	0	1	-
-	-	0	-

$$
\begin{aligned}
J_2 &= Q_1Q_0' \\
J_1 &= 1 \\
J_0 &= Q_2 \\
K_2 &= Q_1 \\
K_1 &= 1 \\
K_0 &= Q_2'Q_1
\end{aligned}
$$

3. **(2 points)** Finish the circuit for the input of each flip-flop for the counter. Assume all flip-flops have both inverted and non-inverted outputs, and use only AND and OR gates in the design.

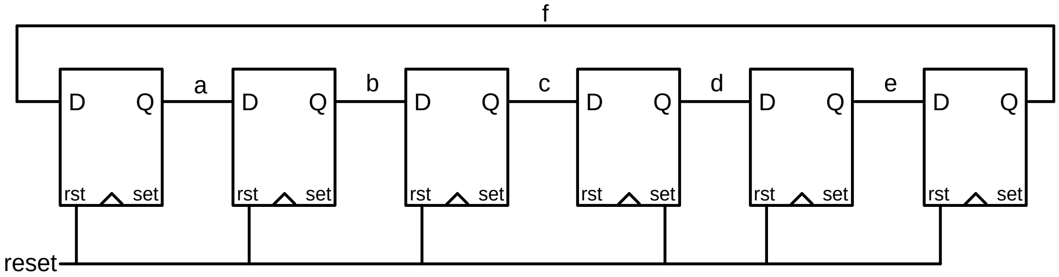
Solution



4. **(4 points)** Design the same autonomous counter with the following output sequence of cycle length 6 using a different component setup:

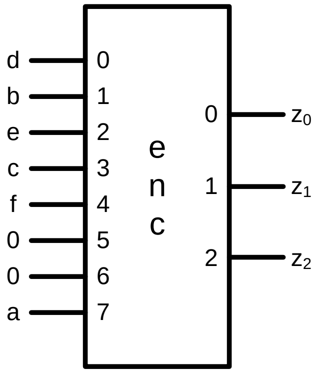
$$0, 2, 4, 7, 1, 3, 0, 2, 4, 7, 1, 3, 0, 2, ...$$

Use the following flip-flop setup and an 8-input encoder. Do not use any additional components. Make sure your design goes to **the initial value of 0** when reset = 1.



Solution When reset = 1, all flip-flop outputs become 0, except $d = 1$. This 1 will shift through the flip-flops every clock cycle with a cycle length of 6. Now we need to connect these signals to an 8-input encoder to produce the output, making sure that d connects to port 0 to match the initial state. Any unused numbers need to be fixed at 0 to provide a valid encoder input.

As the number sequence is 0, 2, 4, 7, 1, 3, we need:



Problem 6 (6 points)

Determine the required cost necessary to design a 20-input decoder network, using 4-input decoders. Use t_{dec} as the delay through a single 4-input decoder, and use t_{AND} as the delay through a single AND gate. There are two methods we can consider.

1. **(3 points)** A tree structure. Answer the following questions (For large numbers, show the correct expression. You do not need to obtain the final number if you do not have a calculator) :

   (a) How many levels are necessary?

   (b) How many 4-input decoder modules are used in total?

   (c) What is the maximum load per network input? Answer in terms of decoder inputs.

   (d) How many AND gates do we need?

   (e) What is the worst case delay through the tree network?

   Solution At each level, we can handle 4 of the 20 inputs we want, as our building block is 4-input decoders. This leads to $\frac{20}{4} = 5$ levels. The number of decoders at each level increases at a ratio of 2^4 which is the number of outputs for a 4-input decoder.

   Therefore, the total number of 5-input decoders is:

$$1 + 2^4 + (2^4)^2 + (2^4)^3 + (2^4)^4 = 1 + 16 + 256 + 4,096 + 65,536 = 69,905$$

The maximum load per network input is for the inputs driving the lowest level decoders. As we saw in the calculation above, there are $(2^4)^4$ decoders at the lowest level, and the input drives one input for each. So the maximum load is 65,536 decoder inputs.

No AND gates are necessary for a tree network. So the number is 0.

The worst case path is through 5 decoders, one at each level. The delay value is $5t_{\text{dec}}$.

To summarize:

(a) $\frac{20}{4} = 5$ levels.

(b)

$$1 + 2^4 + (2^4)^2 + (2^4)^3 + (2^4)^4 = 1 + 16 + 256 + 4,096 + 65,536 = 69,905$$

(c) $(2^4)^4 = 65,536$ decoder inputs

(d) 0

(e) $5t_{\text{dec}}$

2. **(3 points)** A coincident structure using additional AND gates. Answer the following questions (For large numbers, show the correct expression. You do not need to obtain the final number if you do not have a calculator) :

(a) How many 4-input decoders are needed?

(b) We use n-input AND gates to build this network. What is the value of n? Assume that a single AND gate is used for each output.

(c) How many AND gates do we need?

(d) What is the maximum load per decoder output? Answer in terms of AND inputs.

(e) What is the worst case delay through the tree network?

Solution To cover all inputs, we need $\frac{20}{4} = 5$ decoders.

To be able to process one output from each decoder module, we need 5-input AND gates.

The total number of AND gates is equal to the number of outputs, which is $2^{20} = 1,048,576$.

The load per decoder output is equal to the number of AND gates it drives. For one output of a decoder, it is connected to all permutations of output combinations of all the other decoders, so with 5 4-input decoders, the load per decoder output is

$$(2^4)^{(5-1)} = (2^4)^4 = 16^4 = 2^16 = 65,536$$

AND gate inputs.

The worst case path is through one decoder and one AND gate. The delay value is $t_{\text{dec}} + t_{\text{AND}}$.

To summarize:

(a) $\frac{20}{4} = 5$

(b) 5

(c) $2^{20} = 1,048,576$.

(d)

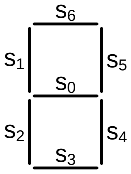
$$(2^4)^{(5-1)} = (2^4)^4 = 16^4 = 65,536$$

AND gate inputs

(e) $t_{\text{dec}} + t_{\text{AND}}$

Problem 7 (9 points)

The 7-segment display is a set of 7 LED lights, where each LED is triggered by the signals shown. Each segment is lit when $s_i = 0$.



Design a code converter that takes a BCD input and produces a 7-segment encoding. The convert table is shown below.

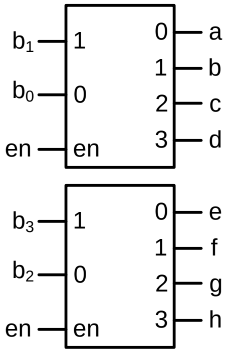
BCD	index	7-segment display						
$b_3b_2b_1b_0$		s_6	s_5	s_4	s_3	s_2	s_1	s_0
0000	0	0	0	0	0	0	0	1
0001	1	1	0	0	1	1	1	1
0010	2	0	0	1	0	0	1	0
0011	3	0	0	0	0	1	1	0
0100	4	1	0	0	1	1	0	0
0101	5	0	1	0	0	1	0	0
0110	6	0	1	0	0	0	0	0
0111	7	0	0	0	1	1	1	1
1000	8	0	0	0	0	0	0	0
1001	9	0	0	0	1	1	0	0

1. **(1 point)** Write s_6 to s_0 in minterm form.

   Solution From the table we can get:

$$\begin{aligned} s_6 &= \sum m(1,4) \\ s_5 &= \sum m(5,6) \\ s_4 &= \sum m(2) \\ s_3 &= \sum m(1,4,7,9) \\ s_2 &= \sum m(1,3,4,5,7,9) \\ s_1 &= \sum m(1,2,3,7) \\ s_0 &= \sum m(0,1,7) \end{aligned}$$

2. **(4 points)**

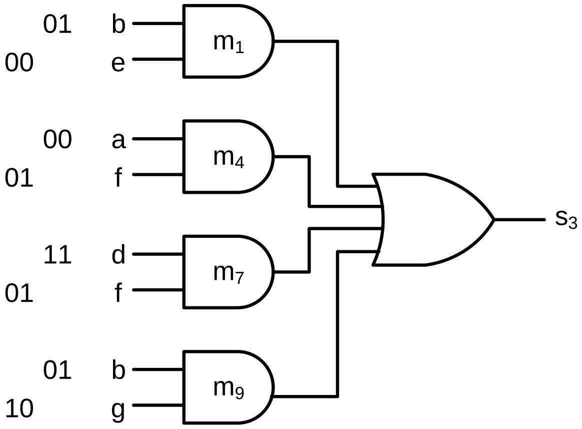


Using the outputs of the above decoders as input to a 2-level AND-OR gate network, design the setup that will produce the signal s_3.

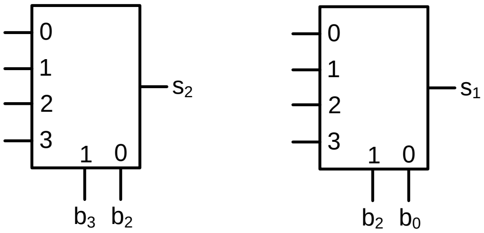
Solution

$$s_3 \quad = \quad \sum m(1,4,7,9)$$

Each minterm can be expressed as the AND of two output signals from the two decoders. The final setup is shown below.



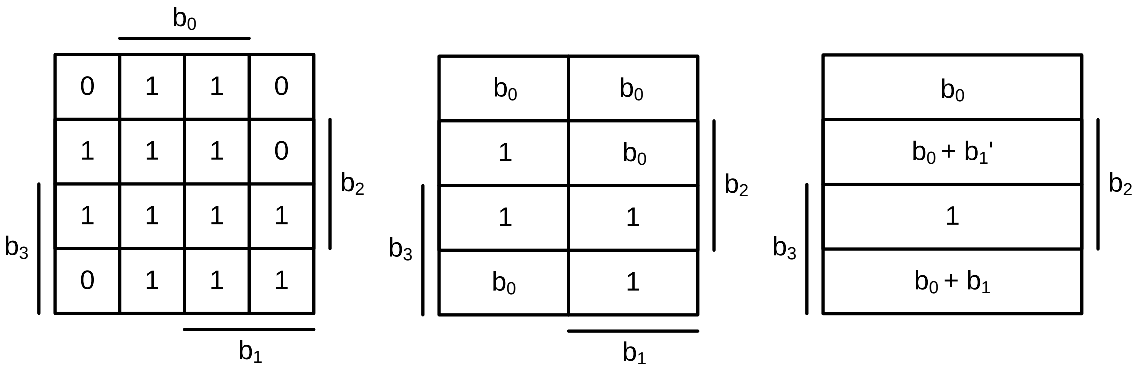
3. **(4 points)**



We wish to obtain the signals s_2 and s_1 using the MUXes and the selection signal setup shown above. Find the expressions that complete the K-map and switching expression. Simplify the expressions as much as possible. For K-maps, do not use any don't-care cases, instead set the values to turn the LED off (to save as much power as possible).

Solution

$$s_2 \quad = \quad \sum m(1,3,4,5,7,9)$$
$$s_1 \quad = \quad \sum m(1,2,3,7)$$

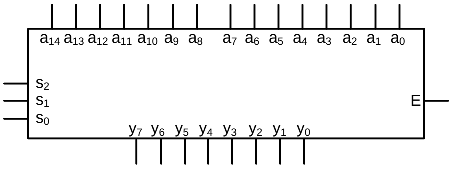


All don't-care cases are set to 1 so that it will turn the LEDs off if those states are ever reached by error.

$$\begin{aligned}
s_1 &= \sum m(1,2,3,7) \\
&= b_3{}'b_2{}'b_1{}'b_0 + b_3{}'b_2{}'b_1b_0{}' + b_3{}'b_2{}'b_1b_0 + b_3{}'b_2b_1b_0 \\
&= b_3{}'b_1{}'m_1(b_2,b_0) + b_3{}'b_1m_0(b_2,b_0) + b_3{}'b_1m_1(b_2,b_0) + b_3{}'b_1m_3(b_2,b_0) \\
&= b_3{}'b_1m_0(b_2,b_0) + b_3{}'m_1(b_2,b_0) + 0\cdot m_2(b_2,b_0) + b_3{}'b_1m_3(b_2,b_0)
\end{aligned}$$

Problem 8 (10 points)

Design a 3-bit decoder using an 8-bit right 7-shifter given below.



Its specification is:

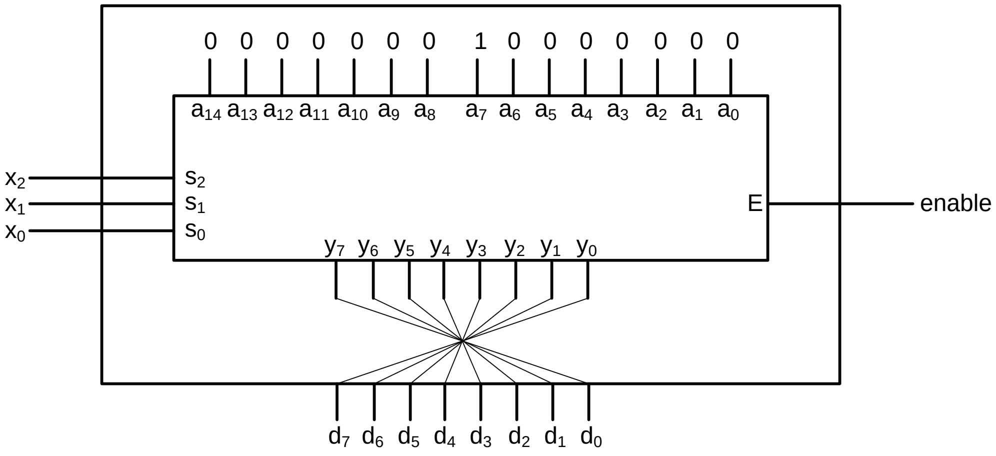
$$\begin{aligned}
\text{inputs:}\quad \underline{a} &= (a_{14}, a_{13}, \cdots, a_8, a_7, a_6, \cdots, a_1, a_0),\ a_i \in \{0,1\} \\
& \quad a_7 \text{ downto } a_0 \text{ are the original input, } a_{14} \text{ downto } a_8 \text{ are bits that get shifted in} \\
\underline{s} &= (s_2, s_1, s_0),\ s_i \in \{0,1\} \\
E &\in \{0,1\} \\
\text{outputs:}\quad \underline{y} &= (y_7, y_6, \cdots, y_1, y_0),\ y_i \in \{0,1\} \\
\text{function:}\quad y_i &= \begin{cases} a_{i+s} & \text{if } E = 1 \\ 0 & \text{otherwise} \end{cases} \\
& \quad \text{where } s = \sum_{j=0}^{2} s_j 2^j \text{ and } i = 0, 1, \cdots, 7
\end{aligned}$$

The 3-bit decoder is specified as follows.

$$\begin{aligned}
\text{inputs:}\quad \underline{x} &= (x_2, x_1, x_0),\ x_j \in \{0,1\} \\
\text{enable} &\in \{0,1\} \\
\text{outputs:}\quad \underline{d} &= (d_7, d_6, \cdots, d_1, d_0),\ d_i \in \{0,1\} \\
\text{function:}\quad d_i &= \begin{cases} 1 & \text{if } (x = i) \text{ and } (E = 1) \\ 0 & \text{otherwise} \end{cases} \\
& \quad \text{where } x = \sum_{j=0}^{2} x_j 2^j \text{ and } i = 0, 1, \cdots, 7
\end{aligned}$$

Clearly show the values of all parallel inputs and how the input/output ports of the shifter is connected to the input/output ports of the decoder. All ports marked **bold** should be accounted for.

Solution With the connection below, we can get a 3-bit decoder. It is an 8-bit right 7-shifter rigged to work as an 8-bit left 7-shifter, and set up with constant values so that the 1 is shifted to the left by the number of bits indicated by the input $\underline{s}$.



Problem 9 (10 points)

1. **(4 points)** Complete the following table for the two's complement number system.

   The 'Representation' column holds the value of the bit-vector representation, and the 'Signed integer' column shows the actual value of the signed integer. For the bit vectors, explicitly show any leading zeroes. Provide a brief explanation if a value cannot be represented using the given bit vector.

   Solution For the first group:

$$\begin{aligned}100101 &= 32+4+1=37\\ 37-2^6 &= 37-64=-27\\ 1100101 &= 64+32+4+1=101\\ 101-2^7 &= 101-128=-27\\ 11100101 &= 128+64+32+4+1=229\\ 229-2^8 &= 229-256=-27\end{aligned}$$

   For the second group:

$$\begin{aligned}42 &= 32+8+2=101010 \text{ (binary)}\\ 42-2^6 &= 42-64=-22\end{aligned}$$

   With a leading 0 in the bit vector, $x_R = x$ for the two remaining lines.

   For the third group, we need at least 7 digits to represent -64 in two's complement as $2^6 = 64$ and with 7 digits we can represent digits between $-64 \le x \le 63$. For the next two lines:

$$\begin{aligned}-64+2^7 &= -64+128=64\\ 64 &= 1000000 \text{ (binary)}\\ -64+2^8 &= -64+256=192\\ 192 &= 128+64=11000000 \text{ (binary)}\end{aligned}$$

Number of bits	Bit-vector $\underline{x}$	Representation x_R (in decimal)	Signed integer x (in decimal)
6	**100101**	37	-27
7	**1100101**	101	-27
8	**11100101**	229	-27
6	101010	**42**	-22
7	0101010	**42**	42
8	00101010	**42**	42
6	–	–	**-64**
7	1000000	64	**-64**
8	11000000	192	**-64**

2. **(1 point)** Two integer numbers are added together using an 8-bit wide two's complement adder. Of the following pairs of numbers, are there any that cause an overflow? Mark **all** pairs that do. If there are none, clearly indicate so.

   Solution For an 8-bit wide two's complement number x, the possible number range is from $-2^7 \leq x \leq 2^7 - 1$. Since $2^7 = 128$, we have $-128 \leq x \leq 127$.

   For the given pairs of numbers, we look for numbers with the same sign that adds up to a number which represents a number with the opposite sign. The pairs that do this is (a) 128 and (h) -129.

3. **(4 points)** Using an 8-bit full adder, we wish to calculate the following sum:

$$72 - (17 + 102) - 23$$

   Assume all numbers are encoded using the two's complement scheme. Show the details of the calculation by completing the following table. Write only the sum bits. Note that the value inside column x may be a negative number.

   Solution The calculation goes as follows:

	x	$\underline{x}$
	17	0001 0001
+	**102**	0110 0110
=	119	0111 0111
	72	0100 1000
+	-119	1000 1001
=	-47	1101 0001
	-47	1101 0001
+	**-23**	1110 1001
=	-70	1011 1010

4. **(1 point)** Is there an overflow during this calculation? Briefly justify your answer.

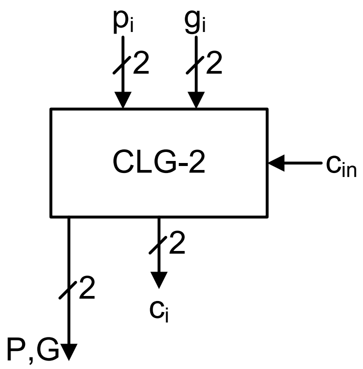
   Solution All addition results are between the range $2^8 - 1 = 127$ to $-2^8 = -128$, so no overflow occurs and the final sum is correct.

Problem 10 (10 points)

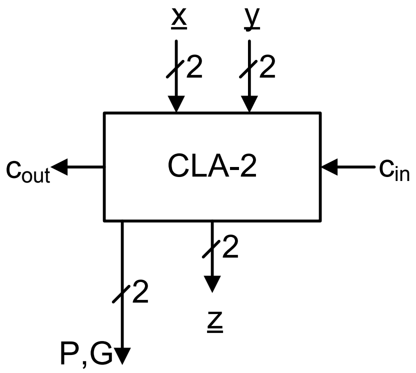
We wish to design and analyze the delay of a 16-bit carry-lookahead adder (CLA-16) network using carry-lookahead generators of size 2 (CLG-2) under various level configurations.

We will only use the following components:

- A 2-input carry-lookahead generator (CLG-2). The inputs are the propagate(p_i) and generate(g_i) signals, and the output is the calculated carries(c_i) of each index, and group propagate(P) and generate(G) signals.



- A 2-input carry-lookahead adder (CLA-2). This is the 2-bit carry-lookahead module. It includes the CLG-2 module above, with additional circuitry for the calculation of p_i and g_i signals that feed the CLG-2 module, and XOR gates for calculating the sum bits z_i.



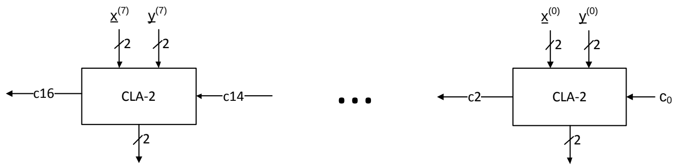
The delay components are listed here.

Term	Module	Description
t_{pg}	CLA-2	delay from inputs $\underline{x}$, $\underline{y}$ to bit propagate p_i and bit generate g_i signals
t_{CLG}	CLG-2	delay from any input of a CLG-2 to any of its outputs
t_s	CLA-2	delay from the carry output (of the CLG-2 inside the CLA-2) to the sum $\underline{z}$

Table 1: Delay components of a carry-lookahead adder.

1. **(3 points)** For a **single-level** implementation of CLA-16 using CLA-2 modules, draw the adder scheme, mark the critical path on the schematics and obtain the delay (in terms provided in Table 1).

   Solution



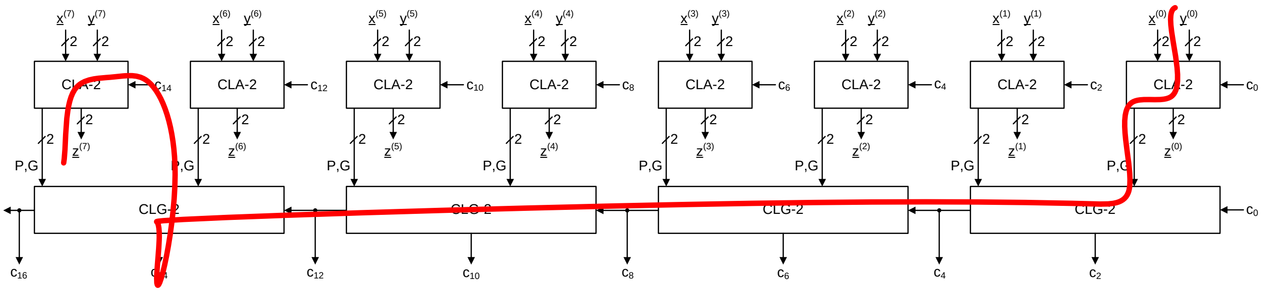
The worst path is from the inputs $\underline{x}^{(0)}$, $\underline{y}^{(0)}$ all the way through the carry chain, and ending at the sum for the last CLA-2 module.

The delay value is:

$$\begin{aligned} t_{1-CLA-16} &= t_{pg} + \frac{16}{2} t_{CLG} + t_s \\ &= t_{pg} + 8t_{CLG} + t_s \end{aligned}$$

2. **(3 points)** For a **two-level** implementation of CLA-16 using CLA-2 and CLG-2 modules, draw the adder scheme, mark the critical path on the schematics and obtain the delay (in terms provided in Table 1).

Solution


x(7) y(7) x(6) y(6) x(5) y(5) x(4) y(4) x(3) y(3) x(2) y(2) x(1) y(1) x(0) y(0)
2
CLA-2
c14 c12 c10 c8 c6 c4 c2 c0
z(7) z(6) z(5) z(4) z(3) z(2) z(1) z(0)
P,G
CLG-2
c16 c14 c12 c10 c8 c6 c4 c2 c0

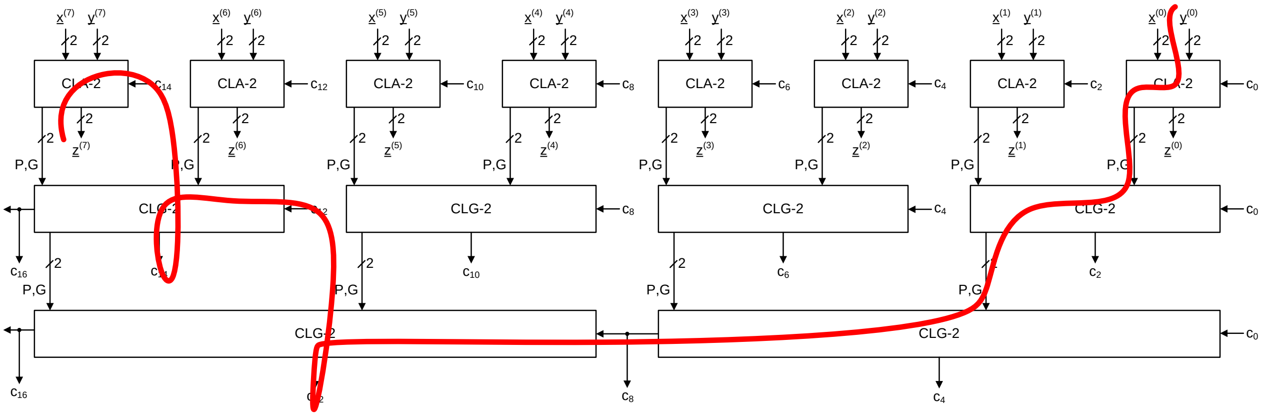
The worst path is from the inputs $\underline{x}^{(0)}$, $\underline{y}^{(0)}$, through the first CLA-2 module, down to the first CLG-2 module via P,G, all the way through the second level, and ending at the sum for the last CLA-2 module.

And the delay is:

$$\begin{aligned} t_{2-CLA-16} &= (t_{pg} + t_{CLG}) + \frac{16}{2 \cdot 2} t_{CLG} + (t_{CLG} + t_s) \\ &= t_{pg} + 6t_{CLG} + t_s \end{aligned}$$

3. **(4 points)** Consider a **three-level** implementation of a CLA-16. Would adding a **third** level provide speed-up over a 2-level implementation? Answer **Yes** or **No** and justify your answer.

Solution The 3-level implementation is as shown.


x(7) y(7) x(6) y(6) x(5) y(5) x(4) y(4) x(3) y(3) x(2) y(2) x(1) y(1) x(0) y(0)
2
CLA-2
c14 c12 c10 c8 c6 c4 c2 c0
z(7) z(6) z(5) z(4) z(3) z(2) z(1) z(0)
P,G
CLG-2
c16 c14 c12 c10 c8 c6 c4 c2 c0
CLG-2
c16 c12 c8 c4 c0

The worst path is from the inputs $\underline{x}^{(0)}$, $\underline{y}^{(0)}$, all the way down to the third level, on to the next CLG-2 module, back up to the last module in the second level, and on to the last CLA-2 module in the first level.

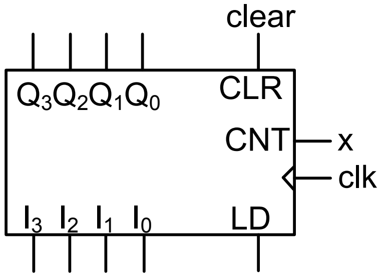
And the delay for this is:

$$\begin{aligned} t_{3-CLA-16} &= (t_{pg} + t_{CLG}) + t_{CLG} + 2t_{CLG} + t_{CLG} + (t_{CLG} + t_s) \\ &= t_{pg} + 6t_{CLG} + t_s \end{aligned}$$

This is exactly the same as $t_{2-CLA-16}$, so the answer is no.

Problem 11 (8 points)

Using a standard modulo-16 binary counter with parallel inputs (shown in the diagram below) along with logic gates, we would like to implement the following counters.

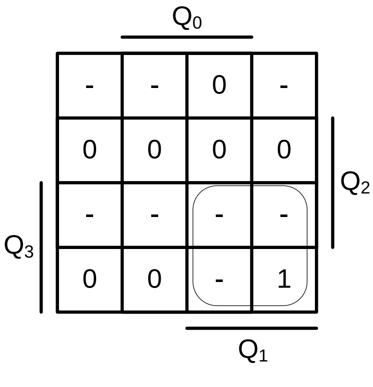


For each design, show the minimization of the gate logic for the load signal LD using K-maps. Clearly show what constant values are given to the parallel input signals I_3 to I_0.

1. **(4 points)** A 3-to-10 counter.

   Solution A 3-to-10 counter starts from 3 and jumps from 10 to 3. Values from 0 to 2 and 11 to 15 are don't-cares as they are never reached.

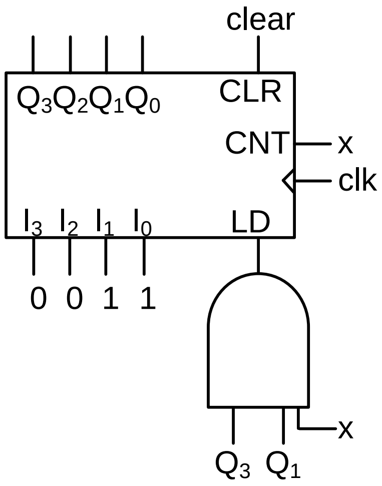
   The K-map for the LD signal is:



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

   where we can get $LD = Q_3Q_1x$.

   The counter implementation is:

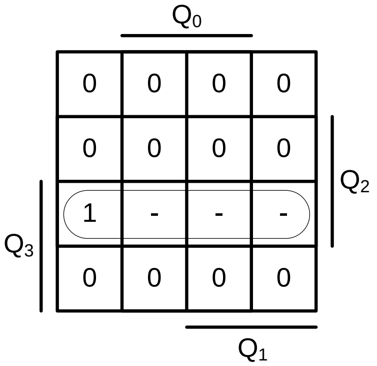


2. **(4 points)** A counter which counts in the following sequence: 0, 1, 11, 12, 13, 4, 5, 6, 7, 8, 9, 14, 15

Use a 4-input binary decoder and a 16-input binary encoder in addition to the standard modulo-16 binary counter. Do not use any variables for the parallel load inputs. Make sure the counter reverts to 0 when the clear signal is given.

Solution We have a decoder and encoder pair which allows us to set up the output sequence at will. Since we have 13 numbers in the sequence, the one remaining thing is to set up the modulo-13 counter from the standard modulo-16 counter. To do this, we need to jump at S_{12} back to S_0.

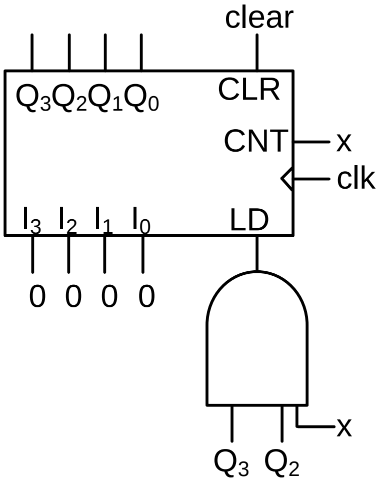
The K-map for the LD signal is:

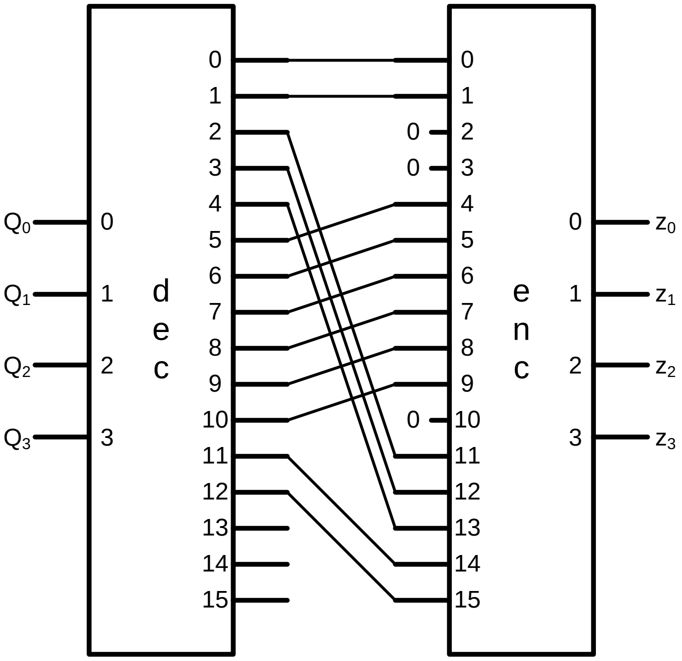


0	0	0	0
0	0	0	0
1	-	-	-
0	0	0	0

where we can get $LD = Q_3Q_2x$.

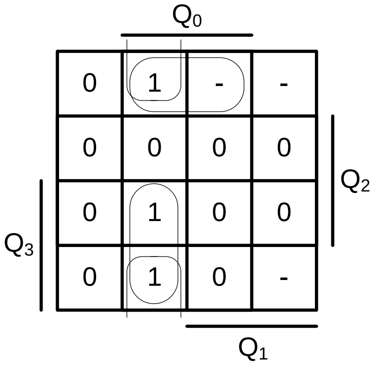
The final counter implementation is:





As a side note, the following shows an implementation without using the decoder plus encoder output. This counter makes three jumps, from 1 to 11, 13 to 4, and 9 to 14. Values 2, 3 and 7 are don't-cares as they are never reached.

The K-map for the LD signal is:

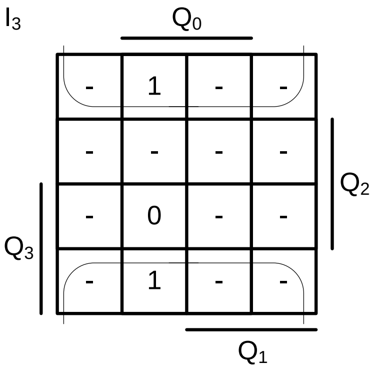


0	1	-	-
0	0	0	0
0	1	0	0
0	1	0	-

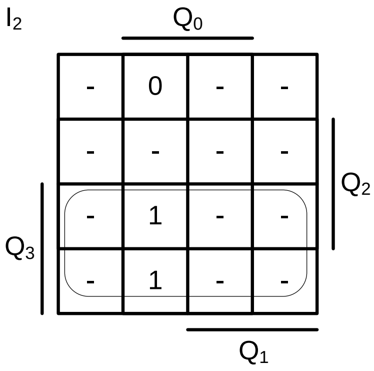
where we can get $LD = (Q_3 Q_1' Q_0 + Q_3' Q_2' Q_0)x$ or $LD = (Q_3 Q_1' Q_0 + Q_2' Q_1' Q_0)x$.

The values of I_0 to I_3 need to be: $I = \begin{cases} 0100 & \text{if } S = 13, \\ 1011 & \text{if } S = 1, \\ 1110 & \text{if } S = 9, \\ \text{don't-care} & \text{otherwise} \end{cases}$

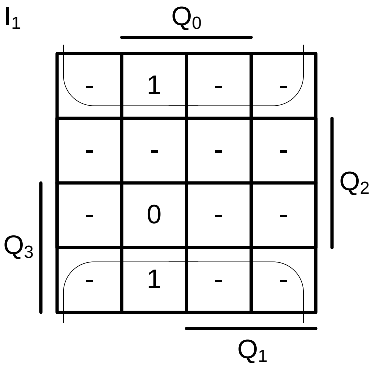
The K-maps are as follows:



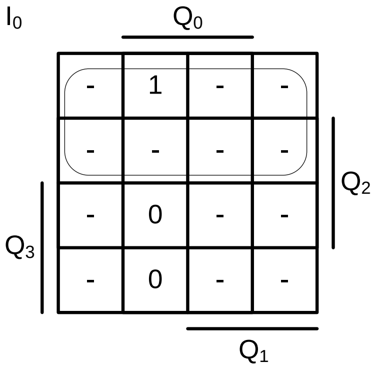
-	1	-	-
-	-	-	-
-	0	-	-
-	1	-	-



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



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

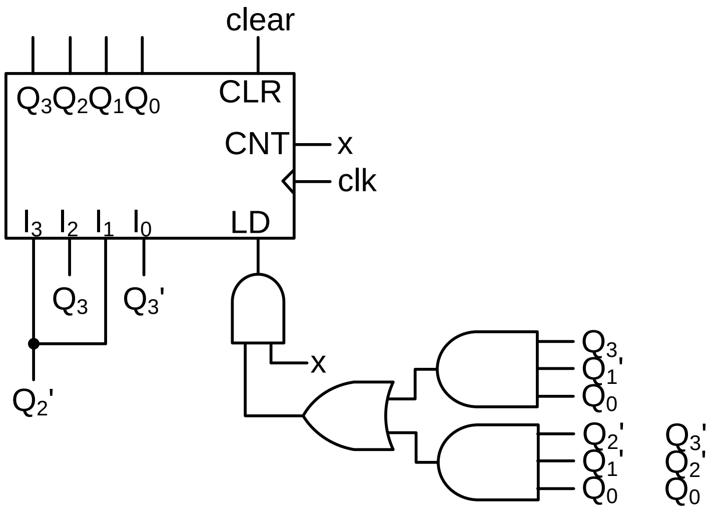


-	1	-	-
-	-	-	-
-	0	-	-
-	0	-	-

and we can get

$$\begin{aligned} I_3 &= Q_2' \\ I_2 &= Q_3 \\ I_1 &= Q_2' \\ I_0 &= Q_3' \end{aligned}$$

The counter implementation is:


clear
Q3Q2Q1Q0
CLR
CNT
x
clk
I3 I2 I1 I0
LD
Q3
Q3'
Q2'
x
Q3'
Q1'
Q0
Q2'
Q1'
Q0
Q3'
Q2'
Q0