Tutorial 1:

1. **Convert 10101101.10110 into Octal, Hexadecimal, and Decimal.**

   a. **Octal**

$(10101101.10110)_2 = (010\ 101\ 101.101\ 100)_2$
$(10101101.10110)_2 = (2\ 5\ 5\ .5\ 4)_8$

   b. **Hexadecimal**

$(10101101.10110)_2 = (1010\ 1101.1011\ 0000)_{16}$
$(10101101.10110)_2 = (10\ 13.11\ 00)_{16}$
$(10101101.10110)_2 = (A\ D.B\ 0)_{16}$

   c. **Decimal**

$(10101101.10110)_2 = (2^7 + 0^6 + 2^5 + 0^4 + 2^3 + 2^2 + 0^1 + 2^0.2^{-1} + 0^{-2} + 2^{-3} + 2^{-4})_{10}$
$(10101101.10110)_2 = (128 + 0 + 32 + 0 + 8 + 4 + 0 + 1.(0.5) + 0 + (0.125) + (0.0625))_{10}$
$(10101101.10110)_2 = (173.6875)_{10}$

2. **Convert $(FD8.C2B)_{16}$ into:**

   a. **Binary**

$(FD8.C2B)_{16} = (F\ D\ 8.C\ 2\ B)_{16}$
$(FD8.C2B)_{16} = (15\ 13\ 8.12\ 2\ 11)_{16}$
$(FD8.C2B)_{16} = (1111\ 1101\ 1000.1100\ 0010\ 1011)_2$

   b. **Octal**

$(FD8.C2B)_{16} = (111\ 111\ 011\ 000.110\ 000\ 101\ 011)_2$
$(FD8.C2B)_{16} = (7\ 7\ 3\ 0.6\ 0\ 5\ 3)_8$

   c. **Decimal**

$(FD8.C2B)_{16} = 2^{11} + 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 0^5 + 2^4 + 2^3 + 0^2 + 0^1 + 0^0.2^{-1} + 2^{-2} + 0^{-3} + 0^{-4} + 0^{-5} + 0^{-6} + 2^{-7} + 0^{-8} + 2^{-9} + 0^{-10} + 2^{-11} + 2^{-12}$
$(FD8.C2B)_{16} = 3840 + 208 + 128 + 0.75 + 0.0078125 + 0.003906$
$(FD8.C2B)_{16} = (4176.76171875)_2$

3. **Perform the following subtraction using 1's CF:**
   $10110.01 - 11010.10$

$$\begin{array}{r} 10110.01 \\ -11010.10 \\ \hline ? \end{array}$$

1's CF of 11010.10=00101.01

$$\begin{array}{r} 10110.01 \\ +00101.01 \\ \hline 11011.10 \end{array}$$

4. **Assume 7-bit word length for a signed binary integers representation. After finding the representations of (+42), (-42), (+13), (-13) in 2's CF, perform the following:**

Positive Number
2's Complement = 1's Complement + 1 bit

$(+42)_{10} = (010\ 1010)_2$
$(-42)_{10} = (101\ 0101 + 1)_2 = (101\ 0110)_2$
$(+13)_{10} = (000\ 1101)_2$
$(-13)_{10} = (111\ 0010 + 1)_2 = (111\ 0011)_2$

a. $(+42) + (-13)$ using addition in 2's CF

Decimal calculation:

$$\begin{array}{r} 42 \\ -13 \\ \hline +29 \end{array}$$

Binary calculation with original magnitude:

$$\begin{array}{r} 010\ 1010 \\ -000\ 1101 \\ \hline ? \end{array}$$

Binary calculation with 2's complement:

$$\begin{array}{r} 010\ 1010 \\ +111\ 0011 \\ \hline 001\ 1101 \end{array}$$

b. $(-42) - (+13)$ using subtraction in 2's CF

Decimal calculation:

$$\begin{array}{r} -42 \\ -\ 13 \\ \hline -55 \end{array}$$

Binary calculation with original magnitude:

$$\begin{array}{r} -010\ 1010 \\ -000\ 1101 \\ \hline ? \end{array}$$

Binary calculation with 2's complement:

$$\begin{array}{r} 101\ 0110 \\ +111\ 0011 \\ \hline 100\ 1001 \end{array}$$

5. **Convert $A = 16.25$ and $B = 8.25$ into binary. Use 7-bits to represent the integral part and 3 bits to represent the fractional part, then perform the following operations in binary:**

$A = (16.25)_{10} = (001\ 0000.010)_2$
$B = (8.25)_{10} = (000\ 1000.010)_2$
$-B = (111\ 0111.101 + 000\ 0000.001)_2 = (111\ 0111.110)_2$

a. **$C = A + B$**

$$\begin{array}{r} 001\ 0000.010 \\ +000\ 1000.010 \\ \hline 001\ 1000.100 \end{array}$$

b. **$D = A - B$ using unsigned binary**

$$\begin{array}{r} 001\ 0000.010 \\ +000\ 1000.010 \\ \hline 001\ 1000.100 \end{array}$$

c. **$F = A - B$ using signed 2's complement**

$$\begin{array}{r} 001\ 0000.010 \\ +\quad 111\ 0111.110 \\ \hline (1)000\ 1000.000 \end{array}$$

Discard the carry (1):
$(1)000\ 1000.000 = 000\ 1000.000$

6. **In a signed binary representation 6-bit wide is there an overflow for the following operations:**

   a. $(+\mathbf{17}) + (+\mathbf{16})$

$(17)_{10} = (01\ 0001)_2$
$(16)_{10} = (01\ 0000)_2$

$$\begin{array}{r} 01\ 0001 \\ +01\ 0000 \\ \hline 10\ 0001 \end{array}$$

The first digit indicates that it is a negative number due to overflow. Therefore, we convert the overflow to a negative sign, and then take the 1's complement of the binary representation:
$(1)0\ 0001 = -\ 1\ 1110 = -(16 + 8 + 4 + 2 + 0) = (-31)_{10}$

   b. $(+\mathbf{17}) + (-\mathbf{17})$

$(17)_{10} = (01\ 0001)_2$
$(-17)_{10} = (10\ \ 1110 + 00\ 0001)_2 = (10\ \ 1111)_2$

$$\begin{array}{r} 01\ 0001 \\ +10\ 1111 \\ \hline (1)00\ 0000 \end{array}$$

When we discard the carry-out (1), we have:
00 0000, which has no overflow. Therefore:
$(00\ 0000)_2 = (0)_{10}$

Tutorial 2:

1. **Perform the binary equivalent of $27 - 61$ using the signed 1's complement representation and enough digits to accommodate the numbers.**

$(+27)_{10} = (001\ 1011)_2$
$(+61)_{10} = (011\ 1101)2$
$(-61)_{10} = (100\ 0010 + 000\ 0001)2 = (100\ 0011)_2$

2. **Simplify algebraically:**

   a. $(AB)'(A'B' + A'B + B)$

$(AB)'(A'B' + A'B + B) = (A' + B')\big(A'B' + B(A' + 1)\big)$
$(AB)'(A'B' + A'B + B) = (A' + B')\big(A'B' + B(1)\big)$
$(AB)'(A'B' + A'B + B) = (A' + B')(A'B' + B)$
$(AB)'(A'B' + A'B + B) = (A'A'B' + A'B + B'A'B' + B'B)$
$(AB)'(A'B' + A'B + B) = (A'B' + A'B + A'B' + 0)$
$(AB)'(A'B' + A'B + B) = (A'B' + A'B + A'B')$
$(AB)'(A'B' + A'B + B) = A'(B' + B + B')$
$(AB)'(A'B' + A'B + B) = A'(B' + B)$
$(AB)'(A'B' + A'B + B) = A'(1)$
$(AB)'(A'B' + A'B + B) = A'$

   b. $(A + C)(AD + AD') + AC + C$

$(A + C)(AD + AD') + AC + C = (A + C)\big(A(D + D')\big) + C(A + 1)$
$(A + C)(AD + AD') + AC + C = (A + C)\big(A(1)\big) + C(1)$
$(A + C)(AD + AD') + AC + C = (A + C)A + C$
$(A + C)(AD + AD') + AC + C = AA + AC + C$
$(A + C)(AD + AD') + AC + C = A + AC + C$
$(A + C)(AD + AD') + AC + C = A(1 + C) + C$
$(A + C)(AD + AD') + AC + C = A(1) + C$
$(A + C)(AD + AD') + AC + C = A + C$

   c. $A'(A + B) + (B + A)(A + B')$

$A'(A + B) + (B + A)(A + B') = AA' + A'B + AB + BB' + AA + AB'$
$A'(A + B) + (B + A)(A + B') = 0 + A'B + AB + 0 + A + AB'$
$A'(A + B) + (B + A)(A + B') = B(A' + A) + A(1 + B')$
$A'(A + B) + (B + A)(A + B') = B(1) + A(1$
$A'(A + B) + (B + A)(A + B') = A + B$

   d. $(xy + z')' + z + xyz' + wz$

$(xy + z')' + z + xyz' + xz = z(x' + y') + z + xyz + wz$
$(xy + z')' + z + xyz' + xz = x'z + y'z + z + xyz' + wz$
$(xy + z')' + z + xyz' + xz = z(x' + y' + 1 + w) + xyz'$
$(xy + z')' + z + xyz' + xz = z(1) + xyz'$
$(xy + z')' + z + xyz' + xz = z + xyz'$
$(xy + z')' + z + xyz' + xz = (z + z')(z + xy)$
$(xy + z')' + z + xyz' + xz = (1)(z + xy)$
$(xy + z')' + z + xyz' + xz = z + xy$

3. Let $Y(A,B,C,D) = A'B'CD + ABC + AB'D$

   a. Draw the circuit.

   b. $Y = \sum m(?)$

$$Y = \sum m(0011 + 1110 + 1111 + 1001 + 1011)$$

   c. Derive its truth table.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

   d. Simplify Y using Boolean algebra.

$$Y(A,B,C,D) = A'B'CD + ABC + AB'D$$
$$Y(A,B,C,D) = B'D(A'C + A) + ABC$$
$$Y(A,B,C,D) = B'D\big(A'C + A(1 + C)\big) + ABC$$
$$Y(A,B,C,D) = B'D(A'C + A + AC) + ABC$$
$$Y(A,B,C,D) = B'D\big(A + C(A' + A)\big) + ABC$$
$$Y(A,B,C,D) = B'D\big(A + C(1)\big) + ABC$$
$$Y(A,B,C,D) = B'D(A + C) + ABC$$
$$Y(A,B,C,D) = AB'D + B'CD + ABC$$
$$Y(A,B,C,D) = AB'D + B'CD + ABC$$

4. Let $f(A, B, C) = \prod M(0, 3, 6, 7)$.

a. Derive its truth table.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

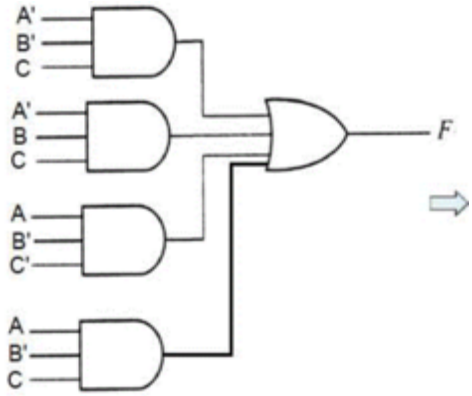
b. Derive its SOP and POS.

$$f(A, B, C) = \sum m(1, 2, 4, 5) = A'B'C + AB'C + AB'C' + ABC'$$

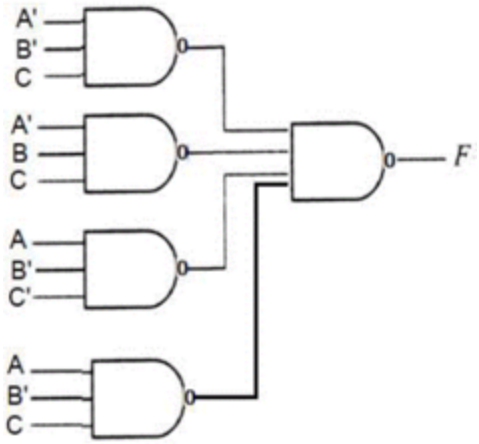
$$f'(A, B, C) = \prod M(0, 3, 6, 7) = (A + B + C)(A + B' + C')(A' + B' + C)(A' + B' + C')$$

c. Draw a NAND implementation of the circuit.

To implement the NAND implementation of the circuit, begin with the SOP implementation:



Convert every gate from the SOP into a NAND gate to derive the NAND implementation of the circuit:



5. **A majority circuit is a combinational circuit whose output is equal to 1 if the input variables have more 1's than 0's. The output is 0 otherwise. We want to design a 3-input majority circuit.**

   a. **Derive the truth table.**

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

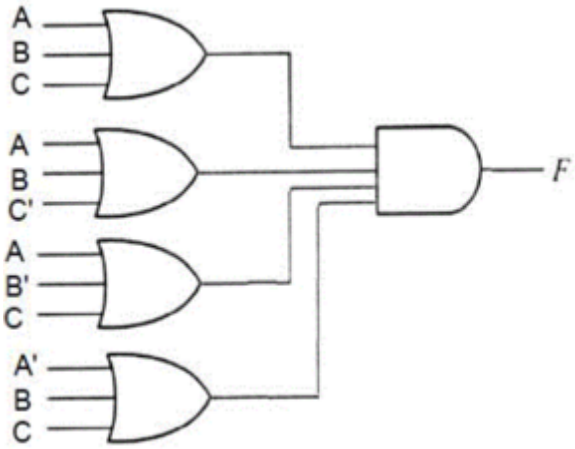
   b. **Derive its SOP and POS.**

$$f(A,B,C) = \sum m(3,5,6,7) = A'BC + AB'C + ABC' + ABC$$

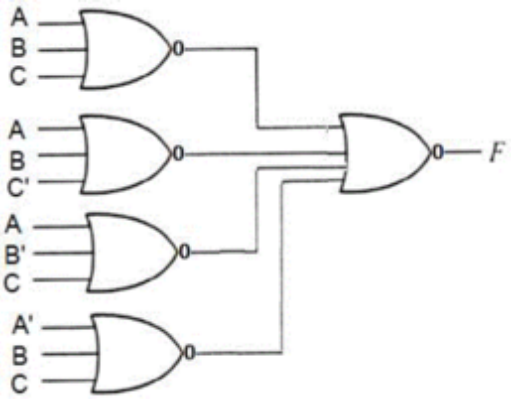
$$f(A,B,C) = \prod M(0,1,2,4) = (A' + B' + C')(A' + B' + C)(A' + B + C')(A + B' + C')$$

   c. **Draw a NOR implementation of the circuit.**

To derive the NOR implementation of the circuit, start with the POS implementation:



Convert every gate from the POS into a NOR gate to derive the NOR implementation of the circuit:



Tutorial 3:

1. **Design a half-subtractor (the basic subtraction of 2 single digits: x-y).**

Derive the truth table for the half-subtractor:

Notes:

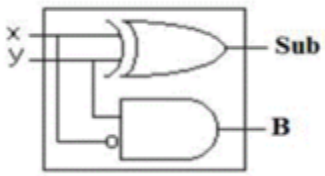
Borrow-Out (B_{out}) is equal to 1 when $x - y < 0$.

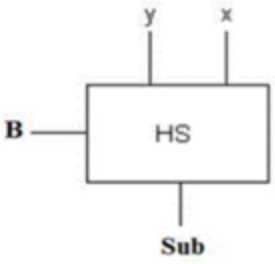
If $B_{out} = 1$, then the Difference ($D_{ifference}$) is $2 + (x - y)$.

x	y	B_{out}	$D_{ifference}$	Comment:
0	0	0	0	$0 - 0 = 0, no\ borrow$
0	1	1	1	$0 - 1 = -1; borrow\ 2; 2 - 1 = 1$
1	0	0	1	$1 - 0 = 1; no\ borrow$
1	1	0	0	$1 - 1 = 0; no\ borrow$

$B_{out} = x'y$

$D = x \oplus y$





2. Design a full-subtractor (FS) using 2 half-subtractors and some gates.

Derive the truth table for the full-subtractor:

Notes:

Borrow-Out (B_{out}) is equal to 1 when $x - y - B_{in} < 0$.

If $B_{out} = 1$, then the Difference ($D_{ifference}$) is $2 + (x - y - B_{in})$.

x	y	B_{in}	B_{out}	$D_{ifference}$	Comment:
0	0	0	0	0	$0-0-0=0; no\ borrow$
0	0	1	1	1	$0-0-1=-1; borrow\ 2; 2-1=1$
0	1	0	1	1	$0-1-0=-1; borrow\ 2; 2-1=1$
0	1	1	1	0	$0-1-1=-2; borrow\ 2; 2-2=0$
1	0	0	0	1	$1-0-0=1; no\ borrow$
1	0	1	0	0	$1-0-1=0; no\ borrow$
1	1	0	0	0	$1-1-0=0; no\ borrow$
1	1	1	1	1	$1-1-1=-1; borrow\ 2; 2-1=1$

Use K-Maps with the terms needed to build a half-subtractor to simplify B_{out}, $D_{ifference}$.

For Half-Subtractor:

$B_{out} = x'y$

$D = x \oplus y$

$$B_{out} = \sum m(1,2,3,7)$$

x/yB_{in}	00	01	11	10
0	0 (0)	1 (1)	1 (3)	1 (2)
1	0 (4)	0 (5)	1 (7)	0 (6)

$B_{out} = x'y'B_{in} + x'y + xyB_{in}$

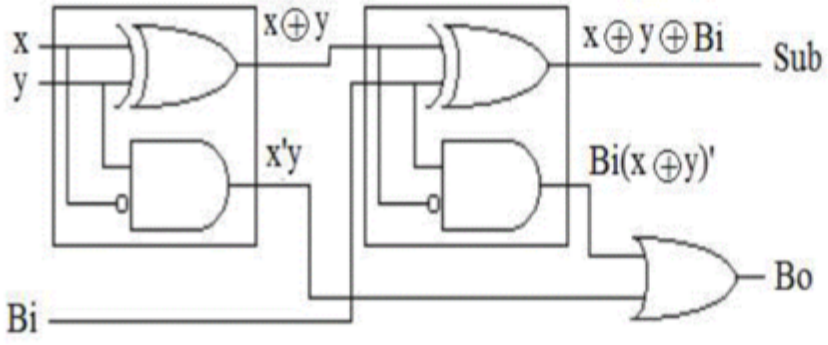
$B_{out} = x'y + B_{in}(x \odot y)$

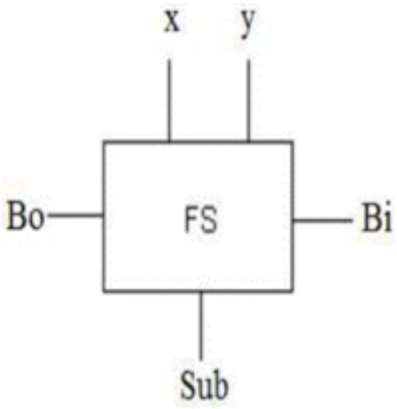
$B_{out} = x'y + B_{in}(x \oplus y)'$

$$D = \sum m(1,2,4,7)$$

v	00	01	11	10
0	0 (0)	1 (1)	0 (3)	1 (2)
1	1 (4)	0 (5)	1 (7)	0 (6)

$D = x'y'B_{in} + x'yB'_{in} + xy'B'_{in} + xyB_{in}$
$D = x'(y \oplus B_{in}) + x(y \odot B_{in})$
$D = x'(y \oplus B_{in}) + x(y \oplus B_{in})'$
$D = x \oplus y \oplus B_{in}$





3. From the following truth table of a function Y:

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

a. Derive the canonical SOP expression.

$$Y = \sum m(3, 9, 11, 14, 15)$$

$$Y = A'B'CD + AB'C'D + AB'CD + ABCD' + ABCD$$

b. Using K-map, derive the optimized SOP.

AB/CD	00	01	11	10
00	0 (0)	0 (1)	1 (3)	0 (2)
01	0 (4)	0 (5)	0 (7)	0 (6)
11	0 (12)	0 (13)	1 (15)	1 (14)
10	0 (8)	1 (9)	1 (11)	0 (10)

SOP: $Y = AB'D + ABC + A'B'CD$

c. Using K-map, derive the optimized POS.

SOP of Y':

$Y' = \prod M(0, 1, 2, 4, 5, 6, 7, 8, 10, 12, 13)$

AB/CD	00	01	11	10
00	0 (0)	0 (1)	1 (3)	0 (2)
01	0 (4)	0 (5)	0 (7)	0 (6)
11	0 (12)	0 (13)	1 (15)	1 (14)
10	0 (8)	1 (9)	1 (11)	0 (10)

SOP of Y':

$Y' = B'D' + BC' + A'C' + A'B$

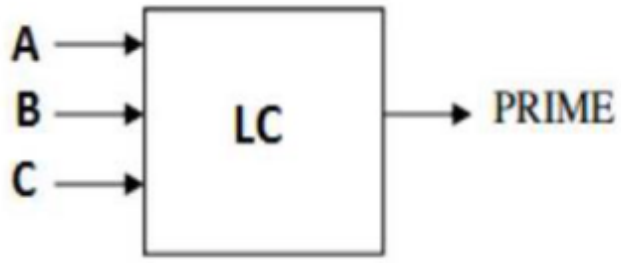
$Y' = A'B + A'C' + BC' + B'D'$

POS of $(Y')'$:

$Y = (Y')' = (A'B)' + (A'C')' + (BC')' + (B'D')'$

$Y = (Y')' = (A + B')(A + C)(B' + C)(B + D)$

4. A system is described by the following bloc diagram:



The output, PRIME, is set to 1 when the number introduced by ABC is prime, otherwise it is 0:

a. Derive the truth table of the PRIME function.

A	B	C	PRIME
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

b. Derive the canonical SOP PRIME.

$$Y = \sum m(1,3,5,7)$$
$$Y = A'B'C + A'BC + AB'C + ABC$$

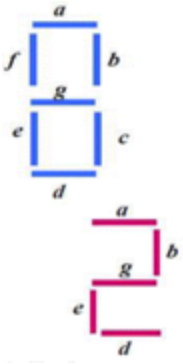
c. Using K-map to derive the optimized SOP of PRIME.

A/BC	00	01	11	10
0	0 (0)	1 (1)	1 (3)	0 (2)
1	0 (4)	1 (5)	1 (7)	0 (6)

$$Y = C$$

5. The BCD to 7-segment system is described by the following truth table:

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	0	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X



a. Derive the 7-segment outputs' SOP using K-map.

$$a = \sum m(0,2,3,5,6,7,8,9);\ a' = \prod M(1,4)$$

$$b = \sum m(0,1,2,3,6,7,8,9);\ b' = \prod M(4,5)$$

$$c = \sum m(0,1,3,4,5,6,7,8,9);\ c' = \prod M(2)$$

$$d = \sum m(0,2,3,5,6,8,9);\ d' = \prod M(1,4,7)$$

$$e = \sum m(0,2,6,8);\ e' = \prod M(1,3,4,5,9)$$

$$f = \sum m(0,4,5,6,8,9);\ f' = \prod M(1,2,3,7)$$

$$g = \sum m(2,3,4,5,6,8,9);\ g' = \prod M(0,1,7)$$

$a = \sum m(0,2,3,5,6,7,8,9);\ a' = \prod M(1,4)$

AB/CD	00	01	11	10
00	1 0	0 1	1 3	1 2
01	0 4	1 5	1 7	1 6
11	X 12	X 13	X 15	X 14
10	1 8	1 9	X 11	X 10

$a = A + BD + B'D' + C$
$a = A + B \odot D + C$

$b = \sum m(0,1,2,3,4,7,8,9);\ b' = \prod M(5,6)$

AB/CD	00	01	11	10
00	1 0	1 1	1 3	1 2
01	1 4	0 5	1 7	0 6
11	X 12	X 13	X 15	X 14
10	1 8	1 9	X 11	X 10

$b = B' + CD + C'D'$
$b = C \odot D + B'$

$$c = \sum m(0,1,3,4,5,6,7,8,9);\ c' = \prod M(2)$$

AB/CD	00	01	11	10
00	1 0	1 1	1 3	0 2
01	1 4	1 5	1 7	1 6
11	X 12	X 13	X 15	X 14
10	1 8	1 9	X 11	X 10

$$c = C' + B + D$$

$d = \sum m(0,2,3,5,6,8,9);\ d' = \prod M(1,4,7)$

AB/CD	00	01	11	10
00	1 (0)	0 (1)	1 (3)	1 (2)
01	0 (4)	1 (5)	0 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

$d = B'D' + B'C + BC'D + A + CD'$
$d = A + B'C + B'D' + BC'D + CD'$
$d = A + B'(C + D') + BC'D + CD'$
$d = A + D'(B' + C) + B'C + BC'D$
$d = A + D'(BC')' + B'C + BC'D$
$d = A + D \odot (BC) + B'C$

$e = \sum m(0,2,6,8);\ e' = \prod M(1,3,4,5,7,9)$

AB/CD	00	01	11	10
00	1 (0)	0 (1)	0 (3)	1 (2)
01	0 (4)	0 (5)	0 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	0 (9)	X (11)	X (10)

$e = B'D' + CD'$
$e = D'(B' + C)$

$f = \sum m(0,4,5,6,8,9);\ f' = \prod M(1,2,3,7)$

AB/CD	00	01	11	10
00	1 (0)	0 (1)	0 (3)	0 (2)
01	1 (4)	1 (5)	0 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

$f = C'D' + BD' + BC' + A$
$f = A + C'D' + BD' + BC'$

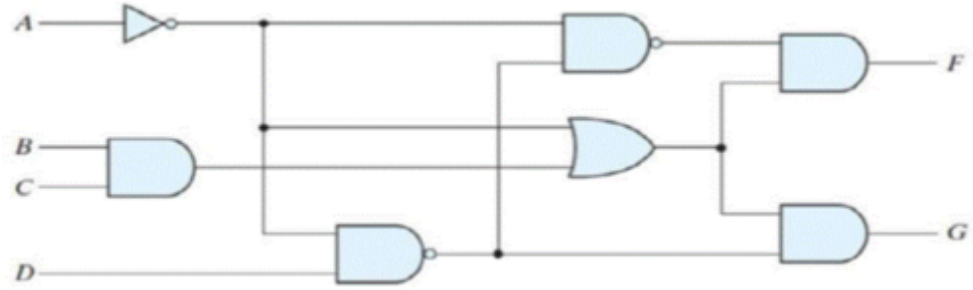
$g = \sum m(2,3,4,5,6,8,9);\ g' = \prod M(0,1,7)$

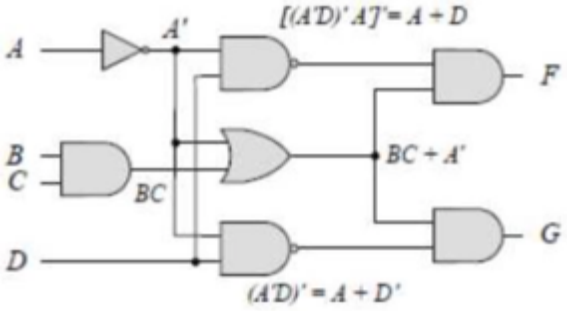
AB/CD	00	01	11	10
00	0 (0)	0 (1)	1 (3)	1 (2)
01	1 (4)	1 (5)	0 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

$g = B'C + CD' + BC' + A$
$g = A + B \oplus C + CD'$

Tutorial 4:

1. Using K-map, derive the SOP of the 2 outputs given by the circuit below:





A	B	C	D	A'	BC	$(A'D)' = A + D'$	$\left(A'(A'D)\right)' = A + D$	$A' + BC$	F	G
0	0	0	0	1	0	1	0	1	0	1
0	0	0	1	1	0	0	1	1	1	0
0	0	1	0	1	0	1	0	1	0	1
0	0	1	1	1	0	0	1	1	1	0
0	1	0	0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1	1	1	0
0	1	1	0	1	1	1	0	1	0	1
0	1	1	1	1	1	0	1	1	1	0
1	0	0	0	0	0	1	1	0	0	0
1	0	0	1	0	0	1	1	0	0	0
1	0	1	0	0	0	1	1	0	0	0
1	0	1	1	0	0	1	1	0	0	0
1	1	0	0	0	0	1	1	0	0	0
1	1	0	1	0	0	1	1	0	0	0
1	1	1	0	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1

$F = \sum m(1,3,5,7,14,15);\ F' = \prod M(0,2,4,6,8,9,10,11,12,13)$

AB/CD	00	01	11	10
00	0 (0)	1 (1)	1 (3)	0 (2)
01	0 (4)	1 (5)	1 (7)	0 (6)
11	0 (12)	0 (13)	1 (15)	1 (14)
10	0 (8)	0 (9)	0 (11)	0 (10)

$F = A'D + ABC$

$G = \sum m(0,2,4,6,14,15);\ G' = \prod M(1,3,5,7,8,9,10,11,12,13)$

AB/CD	00	01	11	10
00	1 (0)	0 (1)	0 (3)	1 (2)
01	1 (4)	0 (5)	0 (7)	1 (6)
11	0 (12)	0 (13)	1 (15)	1 (14)
10	0 (8)	0 (9)	0 (11)	0 (10)

$G = A'D' + ABC$

2. **Design a combinational circuit with 3 inputs and one output described as follows:**

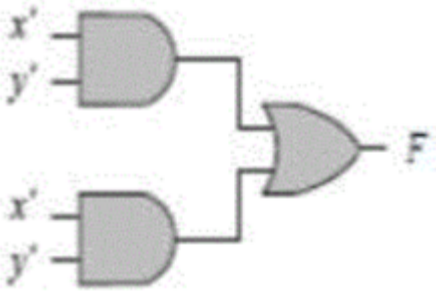
   a. **The output is TRUE when the input binary value is less than 3, otherwise it is FALSE.**

n	x	y	z	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

$$F = \sum m(0,1,2);\ F' = \prod M(3,4,5,6,7)$$

x/yz	00	01	11	10
0	1 (0)	1 (1)	0 (3)	1 (2)
1	0 (4)	0 (5)	0 (7)	0 (6)

$$F = x'y' + x'z' = x'(y' + z')$$



b. The output is TRUE when the input binary value is an even number, otherwise it is FALSE.

n	x	y	z	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

$$F = \sum m(0,2,4,6);\ F' = \prod M(1,3,5,7)$$

x/yz	00	01	11	10
0	1 (0)	0 (1)	0 (3)	1 (2)
1	1 (4)	0 (5)	0 (7)	1 (6)

$$F = z'$$

3. **Implement the following Boolean functions using an appropriate active-high decoder:**
   $F(x, y, z) = x'y'z' + xz$
   $G(x, y, z) = xy'z' + x'y$
   $H(x, y, z) = x'y'z + xy$

Active-high DECODER means the functions will be expressed as a sum of the minterms, (when each function is "1"), so use OR gates of the minterms.
Active-low DECODER means the functions will be expressed as the complement product of the minterms, so use NAND gates of the minterms

Here, the question asks to use an active-high decoder.

The procedure for implementing a combinational circuit by means of a decoder and OR gates requires that the Boolean function for the circuit be expressed as a sum of minterms. A decoder is then chosen that generates all the minterms of the input variables. The inputs to each OR gate are selected from the decoder outputs according to the list of minterms of each function.

Since the inputs are $n = 3$, then the outputs of 2^n are $2^3 = 8$. So $0 \leq D \leq 7$.

First, derive the truth table for the active-high decoder based on the functions for F, G, H:

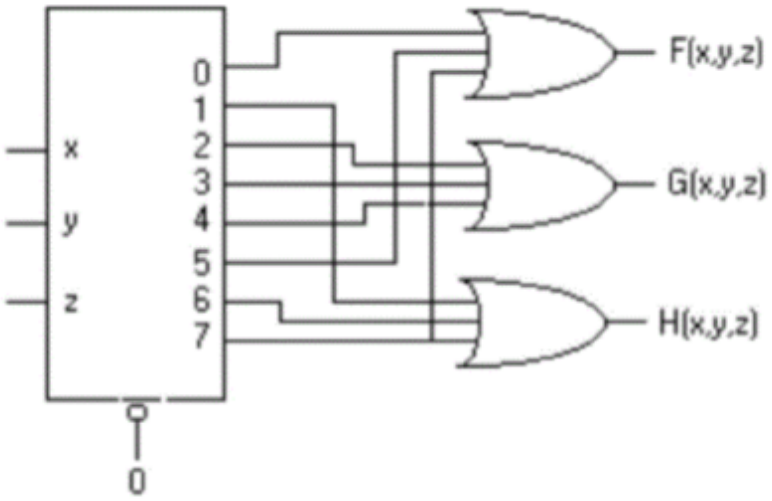
En	x	y	z	F	G	H
1	0	0	0	1	0	0
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1
1	1	1	1	1	0	1

Since there are three inputs and a total of eight minterms, we need a 3-to-8-line decoder.
The decoder generates the eight minterms for x, y, z.

$F = \sum m(0, 5, 7);\ F' = \prod M(1, 2, 3, 4, 6)$
$F = x'y'z' + xy'z + xyz$
The OR gate for output F forms the logical sum of minterms 0, 5, and 7.

$G = \sum m(2, 3, 4);\ G' = \prod M(0, 1, 5, 6, 7)$
$G = x'yz' + x'yz + xy'z'$
The OR gate for the output G forms the logical sum of minterms 2, 3, and 4.

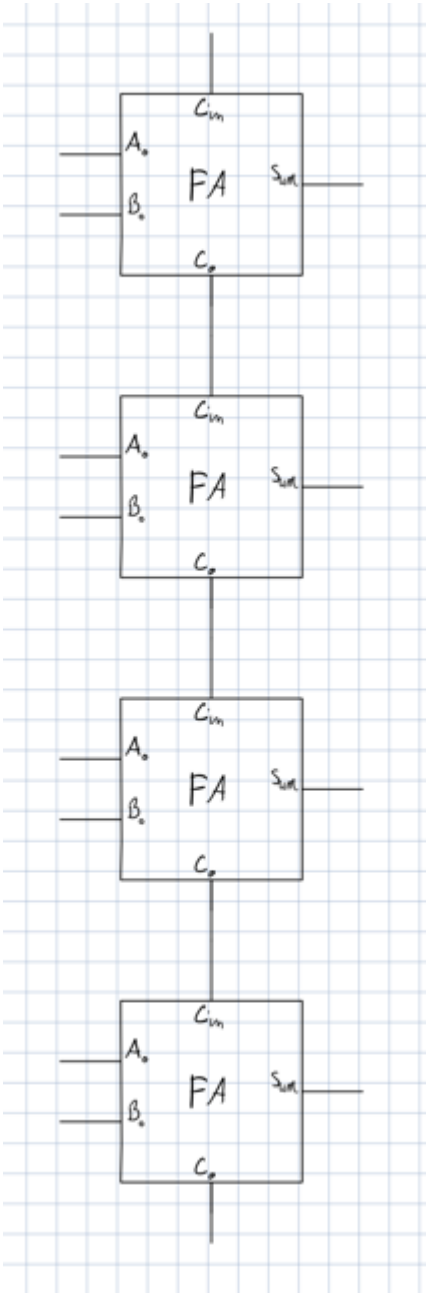
$H = \sum m(1, 6, 7);\ H' = \prod M(0, 2, 3, 4, 5)$
$H = x'y'z + xyz' + xyz$
The OR gate for the output H forms the logical sum of minterms 1, 6, 7.



If the decoder asked for active low, the enable input would be equal to 0, (not inverted), and the OR gates would be changed to NAND gates.

4. **Design a BCD Adder (addition of 2 BCD numbers) using FAs.**

Since we are adding 2 BCD numbers, and there are 4 bits in a BCD number, then there must be 8 inputs, and 4 outputs, and a Carry-Out output.



This design suffers from the carry propagation and can be solved by designing a more complex and expensive carry look-ahead adder.
https://www.geeksforgeeks.org/carry-look-ahead-adder/

Here is the more formal BCD adder design:
https://www.youtube.com/watch?v=wLHFJEQWEng

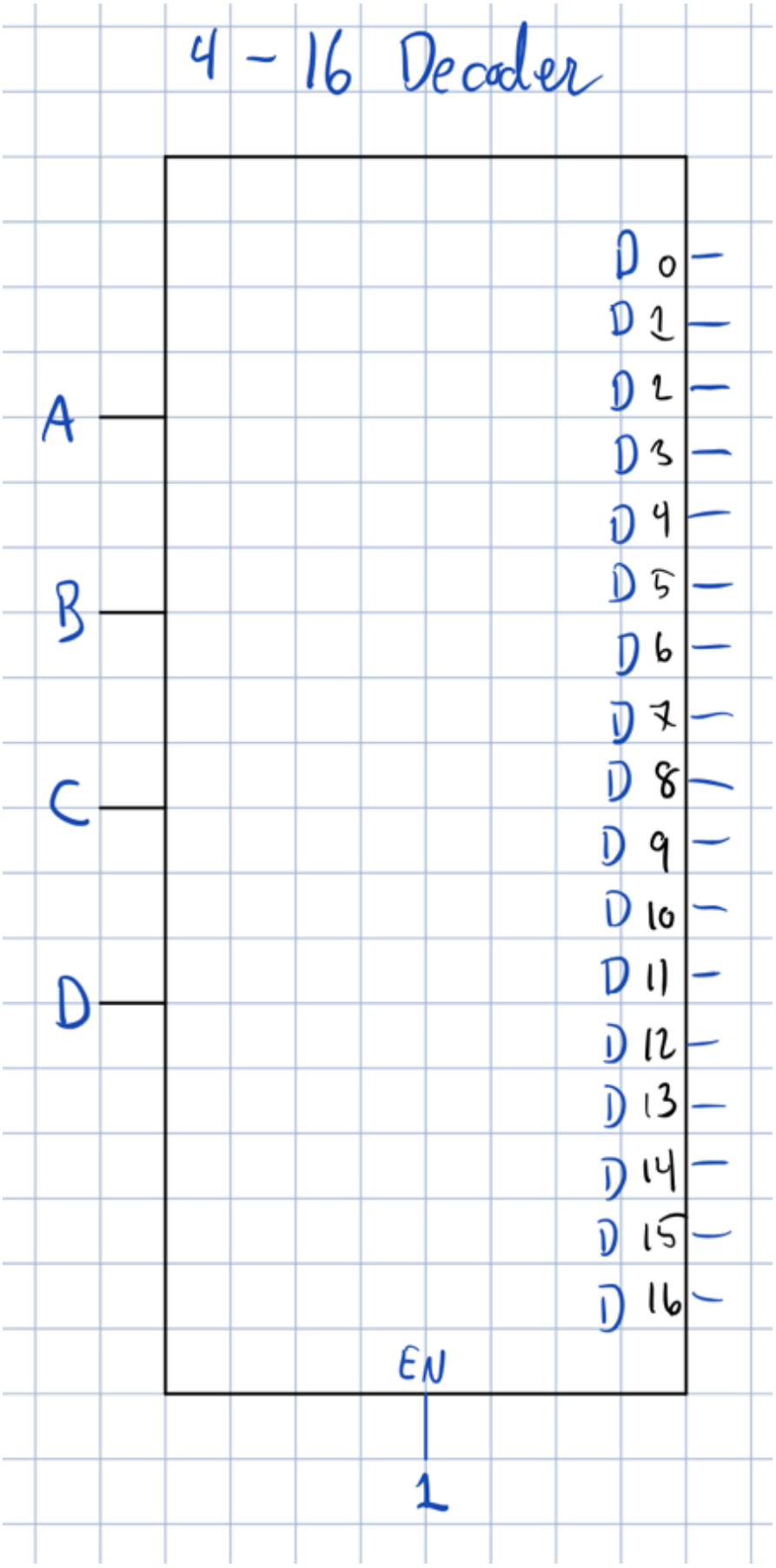
5. **Construct a 4-to-16 line decoder with five 2-to-4 decoders with enable.**

A 4-to-16 line decoder has 4 inputs and 16 outputs.

EN	W	X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Let's create the circuit of a 4-16 line decoder, just for fun. Note: when the enable is high, there is exclusively one minterm for each decoder output:

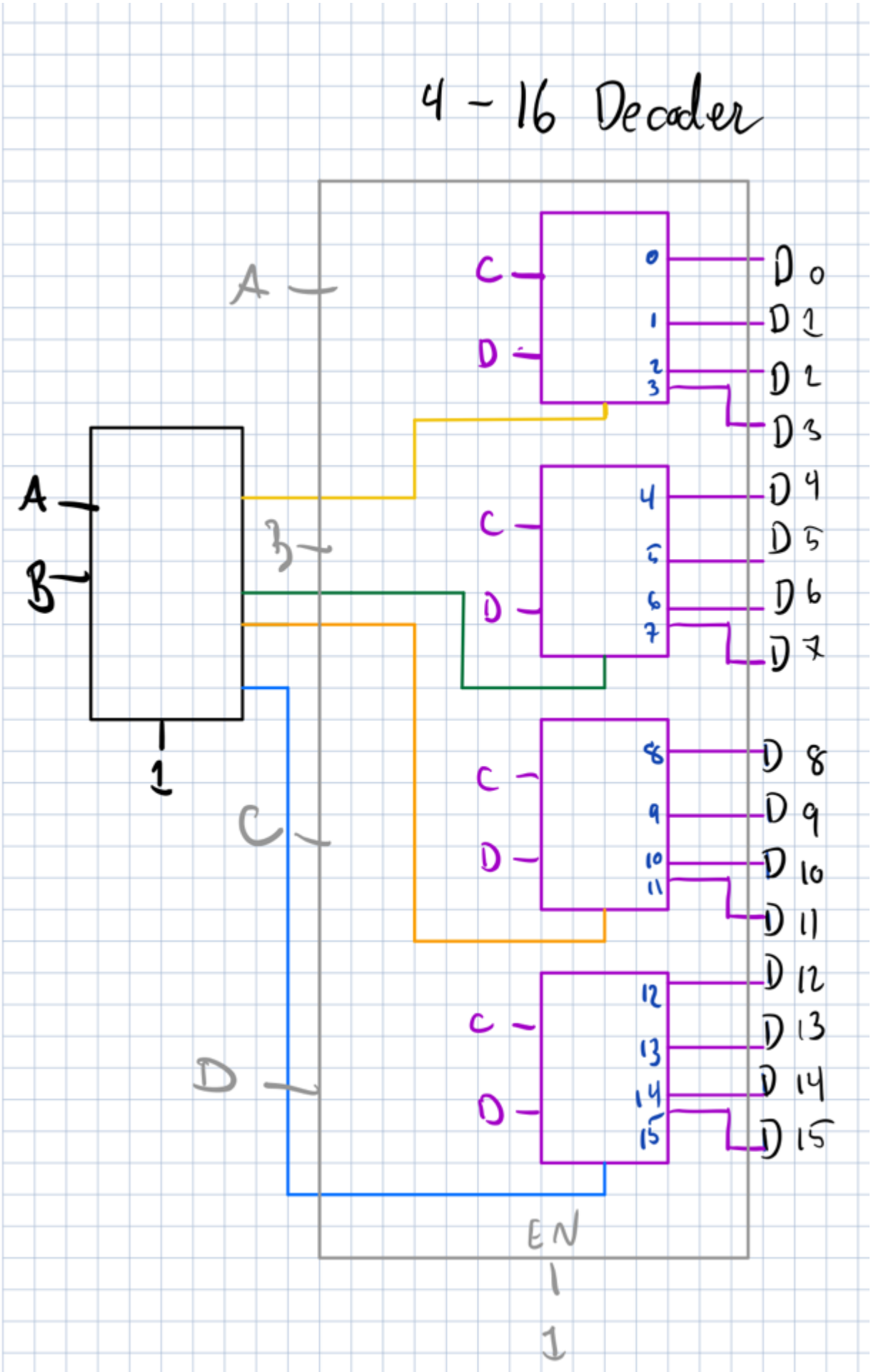
4 - 16 Decoder



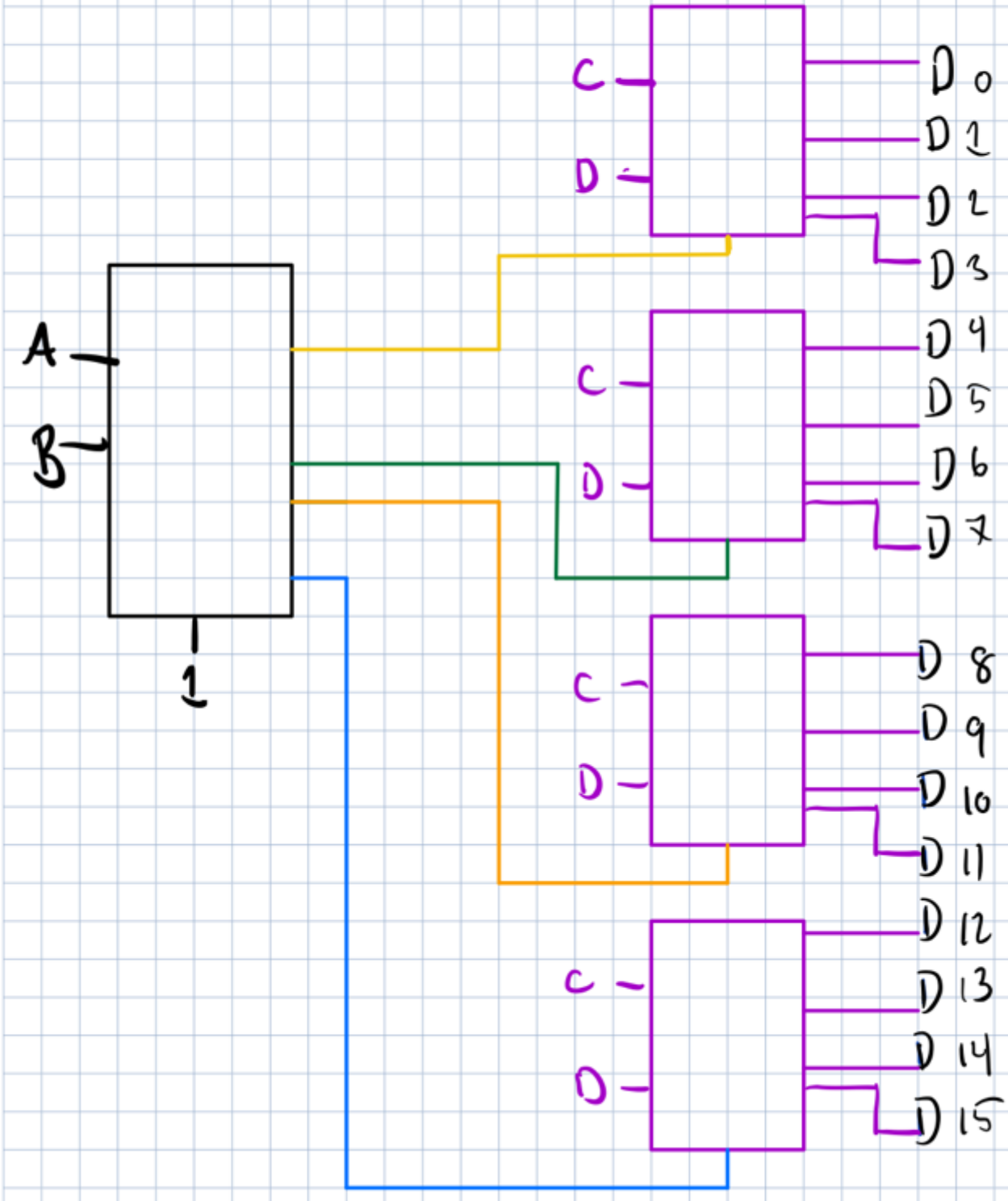
A 4-to-16 line decoder will use $16 \div 4 = 4$, 2-to-4 decoders + 1.
Let's divide the truth table into four sections.

EN	W	X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

4 - 16 Decoder


A
B
1
A
B
C
D
EN
1
C
D
0
1
2
3
D 0
D 1
D 2
D 3
C
D
4
5
6
7
D 4
D 5
D 6
D 7
C
D
8
9
10
11
D 8
D 9
D 10
D 11
C
D
12
13
14
15
D 12
D 13
D 14
D 15

4 - 16 Decoder



Tutorial 5:

1. **Using ROM, implement an Excess-3 adder. Consider two numbers represented in Excess-3, this adder will produce the sum (in Excess-3) and the carry.**
   Example: $1010 + 1001$with a carry of 1 $(7 + 6 = 13)$.

   a. **Derive the logic bloc diagram of the required ROM as well as its inputs and outputs (label them). Note that the inside of the ROM is not to be included. Specify the size of the ROM.**

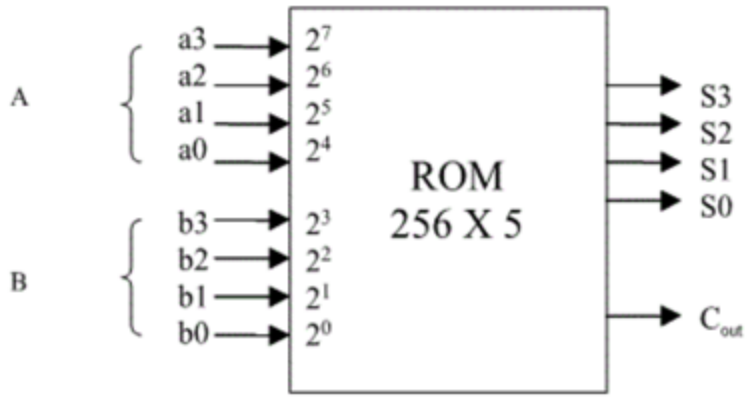
[add steps]
https://computationstructures.org/lectures/combinational/combinational.html#25

First, write a decimal to excess-3 conversion table:
Note: Count for an excess-3 system *starts* at the BCD representation for "3".

Decimal Digit	Excess-3 Code
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

To find the size of the ROM, declare the amount of inputs there are:
One decimal digit is represented with 4 bits in the excess-3 system.
Since we are adding two excess-3 numbers, then there are $n = 8$ inputs.
The outputs for the ROM are the sum of each bit, and the Carry-Out.

Therefore, the size of the ROM is: $2^n \times Output = 2^8 \times 5 = 256 \times 5$



b. Provide the partial truth table of the ROM (first 8 combinations and last 8 combinations + some in between).

a3 a2 a1 a0 b3 b2 b1 b0	S3 S2 S1 S0 C_{out}	Interpretation
0 0 0 0 0 0 0 0	X X X X X	(Inputs A and B are invalid.)
0 0 0 0 0 0 0 1	X X X X X	(Inputs A and B are invalid.)
0 0 0 0 0 0 1 0	X X X X X	(Inputs A and B are invalid.)
0 0 0 0 0 0 1 1	X X X X X	(Input A is invalid.)
0 0 0 0 0 1 0 0	X X X X X	(Input A is invalid.)
0 0 0 0 0 1 0 1	X X X X X	(Input A is invalid.)
0 0 0 0 0 1 1 0	X X X X X	(Input A is invalid.)
0 0 0 0 0 1 1 1	X X X X X	(Input A is invalid.)
…	…	…
…	…	…
0 0 1 1 0 0 1 1	0 0 1 1 0	(0 + 0 = 0)
0 0 1 1 0 1 0 0	0 1 0 0 0	(0 + 1 = 1)
…	…	…
…	…	…
1 1 0 0 1 1 0 0	1 0 1 1 1	(9 + 9 = 18)
…	…	…
…	…	…
1 1 1 1 1 0 0 0	X X X X X	(Les entrées A et B sont invalides.)
1 1 1 1 1 0 0 1	X X X X X	(Les entrées A et B sont invalides.)
1 1 1 1 1 0 1 0	X X X X X	(Les entrées A et B sont invalides.)
1 1 1 1 1 0 1 1	X X X X X	(Inputs A and B are invalid.)
1 1 1 1 1 1 0 0	X X X X X	(Inputs A and B are invalid.)
1 1 1 1 1 1 0 1	X X X X X	(Inputs A and B are invalid.)
1 1 1 1 1 1 1 0	X X X X X	(Inputs A and B are invalid.)
1 1 1 1 1 1 1 1	X X X X X	(Inputs A and B are invalid.)

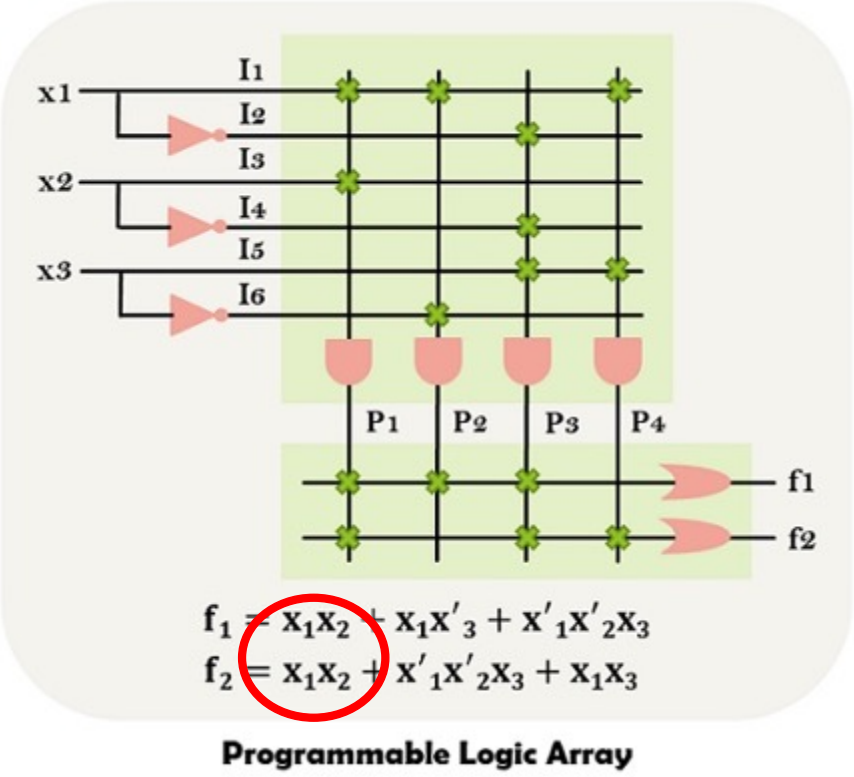
2. Braille is a tactile writing system for people with a visual impairment. It uses a pattern of dots to represent alphanumeric characters. The table below shows the correlation between BCD numbers and dots (a dot is associated to "1").

A	B	C	D	W Z	X Y
0	0	0	0	⠁	
0	0	0	1	⠂	
0	0	1	0	⠄	
0	0	1	1	⠉	
0	1	0	0	⠅	
0	1	0	1	⠊	
0	1	1	0	⠆	
0	1	1	1	⠋	
1	0	0	0	⠇	
1	0	0	1	⠌	

a. **What is the difference between PLA and PAL?**

PLA and PAL are types of Programmable Logic Devices (PLD) which are used to design combination logic together with sequential logic. The significant difference between the PLA and PAL is that the PLA consists of the programmable array of AND and OR gates, while PAL has the programmable array of AND but a fixed array of OR gate.

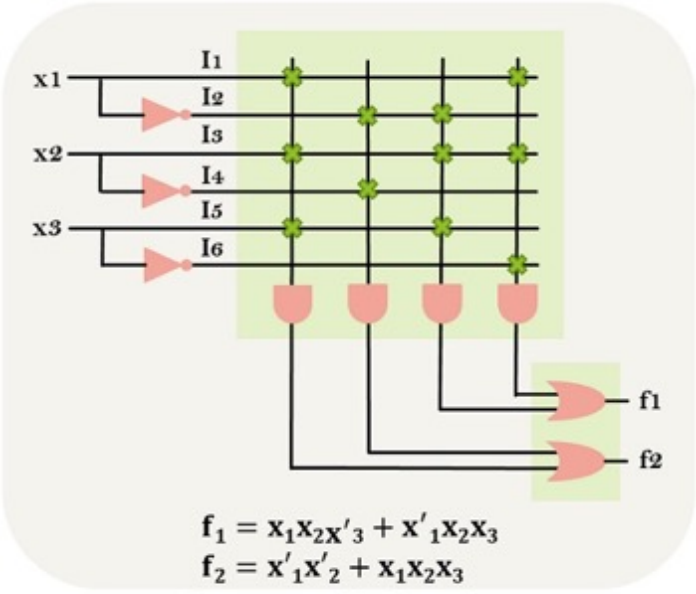
PLA:
https://www.youtube.com/watch?v=CC7DHECXcLE
https://www.youtube.com/watch?v=jrQ1YYgiOTo
PLA stands for the Programmable Logic Array which presents the Boolean function in the SOP (Sum of Products) form. The PLA contains NOT, AND and OR gates fabricated on the chip. It passes every input by a NOT gate which makes each input and its complement available to every AND gate. The output of each AND gate is given to the each OR gate. At last, the OR gate output produces chip output. So, this is how suitable connections are made to employ SOP expressions.
When simplifying each function with K-MAPS, it is important to find common product terms among each function, even if not completely optimized.



Programmable Logic Array

In PLA the connections to both AND and OR arrays are programmable. PLA is considered more expensive and complex as compared to the PAL. The two different manufacturing techniques can be used for PLA to increase the ease of programming. In this technique, each connection is built through a fuse at every intersection point where the unwanted connections can be removed by blowing the fuses. The latter technique involves the connection making at the time of the fabrication process with the help of the proper mask provided for the specific interconnection pattern.

PAL:
https://www.youtube.com/watch?v=1nNb_hBMGu8
https://www.youtube.com/watch?v=qlq4NHk5Y_w
PAL (Programmable Array Logic) is also a PLD (Programmable Logic Device) circuit which works similar to the PLA. PAL employs the programmable AND gates but fixed OR gates, unlike PLA. It implements two simple functions where the number of linked AND gates to each OR gate specifies the maximum number of product terms that can be generated in a sum-of-products representation of the particular function. While the AND gates are perpetually connected to the OR gates, **which signifies that the produced product term is not shareable with the output functions.**



Programmable Array Logic

b. **Implement an encoding BCD-to-Braille using a PLA of appropriate size. Derive its programming table and the circuit.**

First create the truth table for the BCD to Braille circuit:

A	B	C	D	W	X	Y	Z
0	0	0	0	0	1	1	1
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	1	1	0	0
0	1	0	0	1	1	1	0
0	1	0	1	1	0	1	0
0	1	1	0	1	1	0	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	1	1
1	0	0	1	0	1	0	1
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Use K-Maps to find the simplified SOP with the functions' minterms and maxterms. If you can find a product term similar to another function (but not necessarily completely optimized) simplify to that common term:

$$W = \sum m(1,2,3,4,5,6,7,8);\ W' = \prod M(0,9)$$

AB/CD	00	01	11	10
00	0 (0)	1 (1)	1 (3)	1 (2)
01	1 (4)	1 (5)	1 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	0 (9)	X (11)	X (10)

$W = AD' + B + A'D + C$
$W = B + C + A'D + AD'$
$W' = A'B'C'D' + AD$

$$X = \sum m(0,3,4,6,7,9);\ X' = \prod M(1,2,5,8)$$

AB/CD	00	01	11	10
00	1 (0)	0 (1)	1 (3)	0 (2)
01	1 (4)	0 (5)	1 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	0 (8)	1 (9)	X (11)	X (10)

$X = A'C'D' + AD + CD + \{BC\ or\ BD'\}$
$X = AD + A'C'D' + CD + \{BC\ or\ BD'\}$
$X' = B'C'D' + A'C'D + AD'$

$$Y = \sum m(0,4,5,7,8);\ Y' = \prod M(1,2,3,6,9)$$

AB/CD	00	01	11	10
00	1 (0)	0 (1)	0 (3)	0 (2)
01	1 (4)	1 (5)	1 (7)	0 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	0 (9)	X (11)	X (10)

$Y = C'D' + BD$
$Y = BD + C'D'$
$Y' = B'D + CD'$

$$Z = \sum m(0,2,6,7,8,9);\ Z' = \prod M(1,3,4,5)$$

AB/CD	00	01	11	10
00	1 (0)	0 (1)	0 (3)	1 (2)
01	0 (4)	0 (5)	1 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

$Z = B'D' + BC + \{A\ or\ AD\}$
$Z = A + B'D' + BC$
$Z' = BC' + A'B'D$

List the unique terms between the function or complement function, which ever has the least amount of terms, (the lowest cost).

Summary of all simplified SOPs:
W=B+C+A'D+AD'
X= AD +A'C'D'+CD+{BC or BD'}
Y=BD+C'D'
Z=BC+B'D'+{A or AD}

W'=A'B'C'D'+AD
X'=B'C'D'+A'C'D+AD'
Y'=B'D+CD'
Z'=BC'+A'B'D

Since AD is the most common term among the functions, we select all of the functions with AD, and repeat for all other functions that have the most repeating terms. When there is a function that does not share any terms, select the function that has the lowest cost. If the cost is equal, any of the functions may be used.

Chosen Functions:
W'=A'B'C'D'+AD
X= AD +A'C'D'+CD+BC
Y=BD+C'D'
Z=BC+B'D'+AD

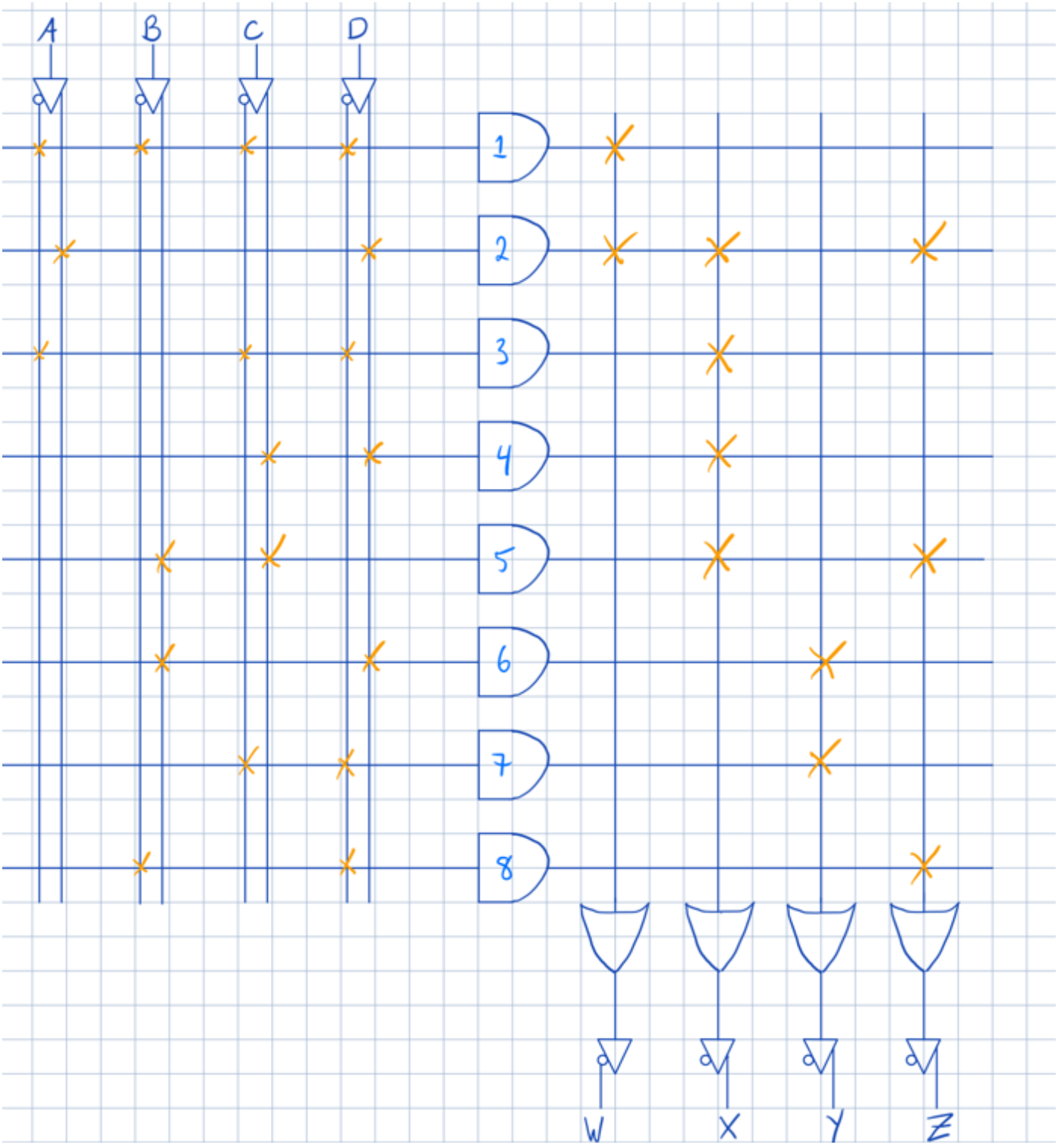
Label all the distinct terms into the PLA Programming Table:

Notes:

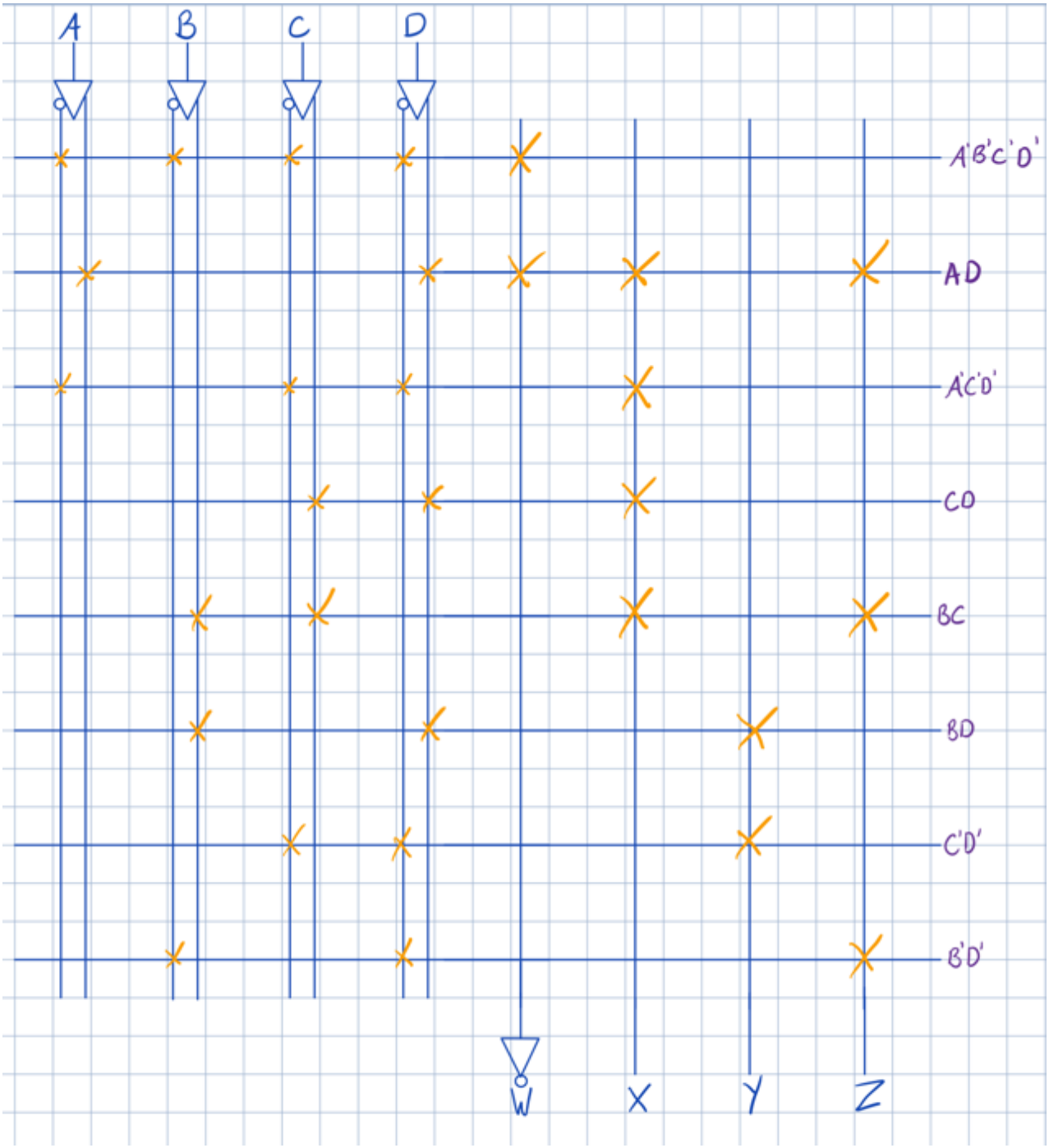
- # is label that will be used later. It should be numerical down the table.
- "Distinct Terms" are all the distinct terms in the functions that have been derived using K-Maps.
- Label the terms "A, B, C, D" with their binary values for each variable.
- Label each function at the bottom with a "T" for "TRUE" if you chose the "True" SOP, or a "C" for "COMPLEMENT" if you chose the "complement" SOP.
    - If the function is "C", remember to use an inverter in the logic circuit.

#	Product Terms	A	B	C	D	W	X	Y	Z
1	A'B'C'D'	0	0	0	0	1			
2	AD	1				1	1		1
3	A'C'D'	0		0	0		1		
4	CD			1	1		1		
5	BC		1	1			1		1
6	BD		1		1			1	
7	C'D'			0	0			1	
8	B'D'		0		0				1
						C	T	T	T

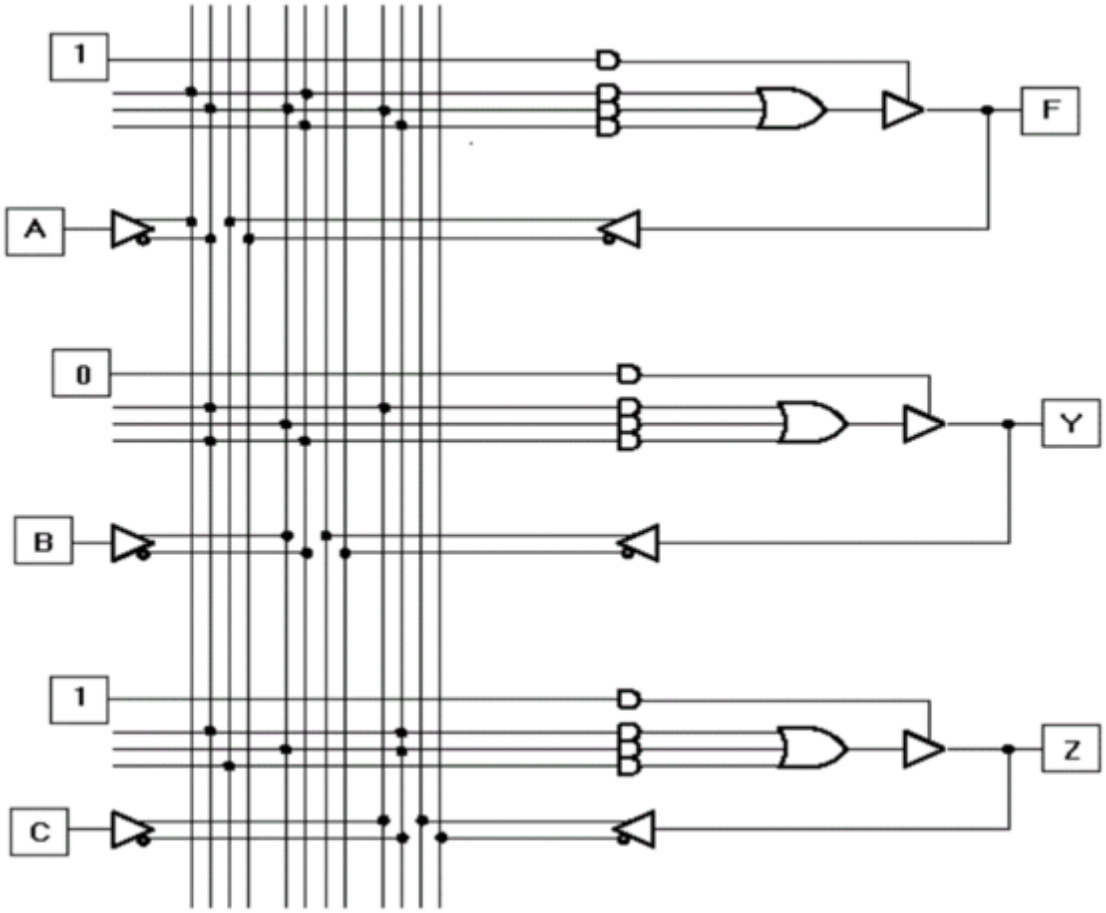
Unprogrammed PLA Logic Circuit:



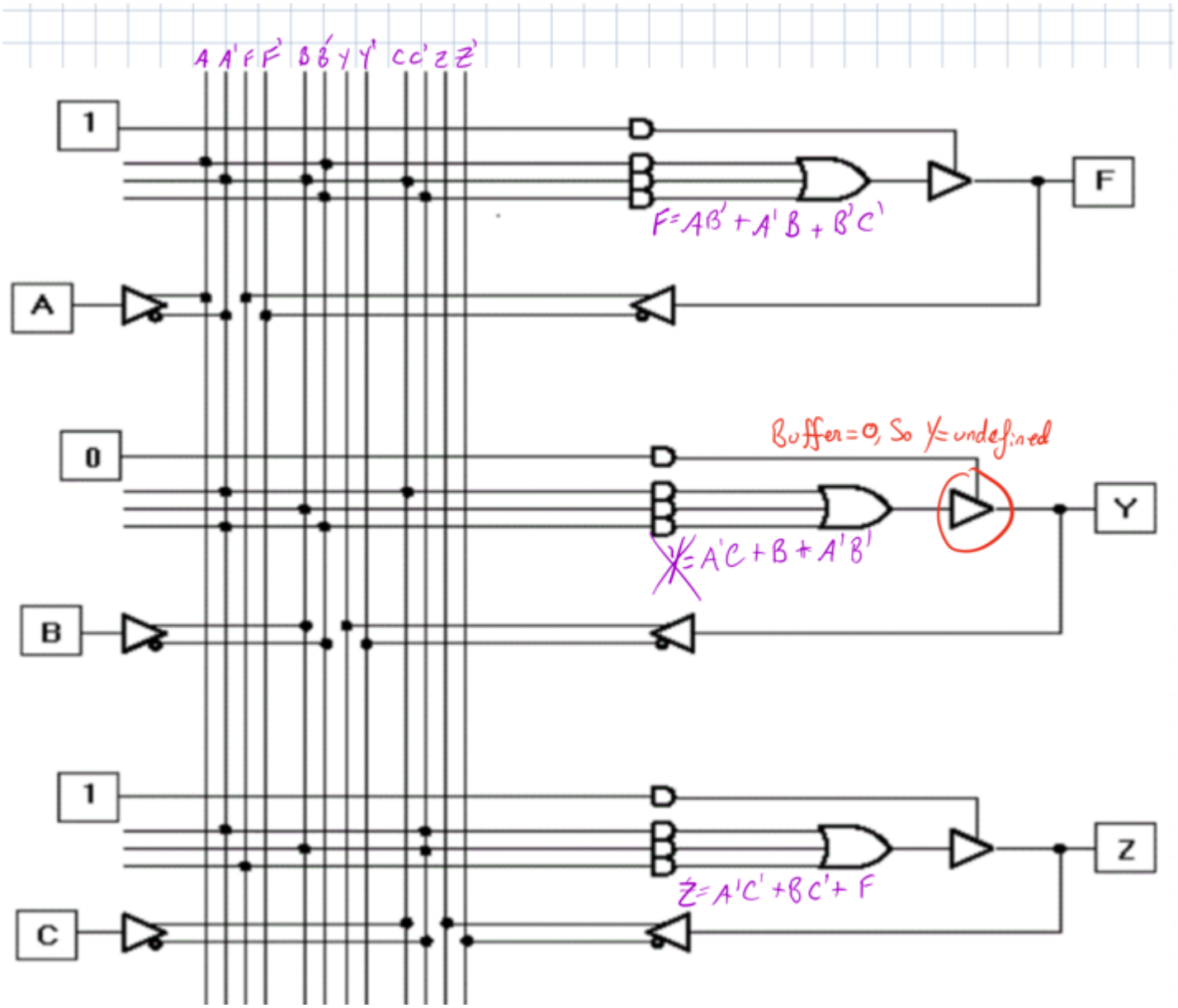
Programmed Logic Circuit:



3. **Derive the algebraic expressions of $F(A, B, C), Y(A, B, C)$ *and* $Z(A, B, C)$, that are realized by the following circuit:**



Notes:



$F(A,B,C) = AB' + A'B + B'C'$
$Y(A,B,C) = [undefined]$ due to buffer $= 0$
$Z(A,B,C) = A'C' + B'C' + F(A,B,C)$

4. **Assume that the inputs X_i and Y_i of a full-adder of an arithmetic unit are defined as follows:**
   $X_i = A_i'S + A_iC_{in}'$ and $Y_i = B_i$
   where:
   S is a selector input
   C_{in} is the inside carry
   A_i and B_i are the data inputs in a 1 bit step i $(0 \leq i \leq 3)$

   a. **Draw the logic diagram of a 4-bit arithmetic unit.**

Similarly, since the domain of i is $0 \leq i \leq 3$, then:

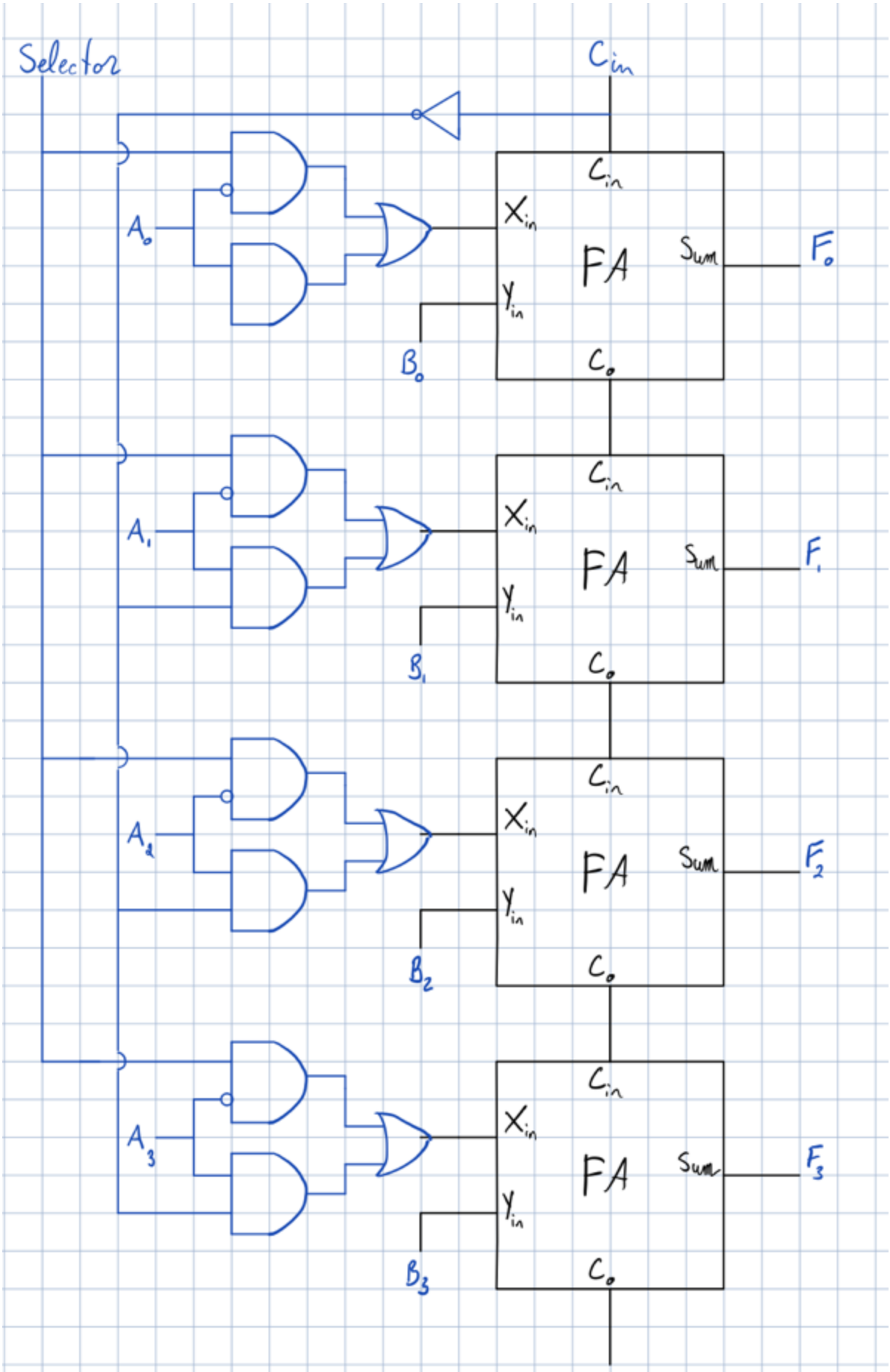
$X_0 = A_0'S + A_0C_{in}'$
$Y_0 = B_0$

$X_1 = A_1'S + A_1C_{in}'$
$Y_1 = B_1$

$X_2 = A_2'S + A_2C_{in}'$
$Y_2 = B_2$

$X_3 = A_3'S + A_3C_{in}'$
$Y_3 = B_3$

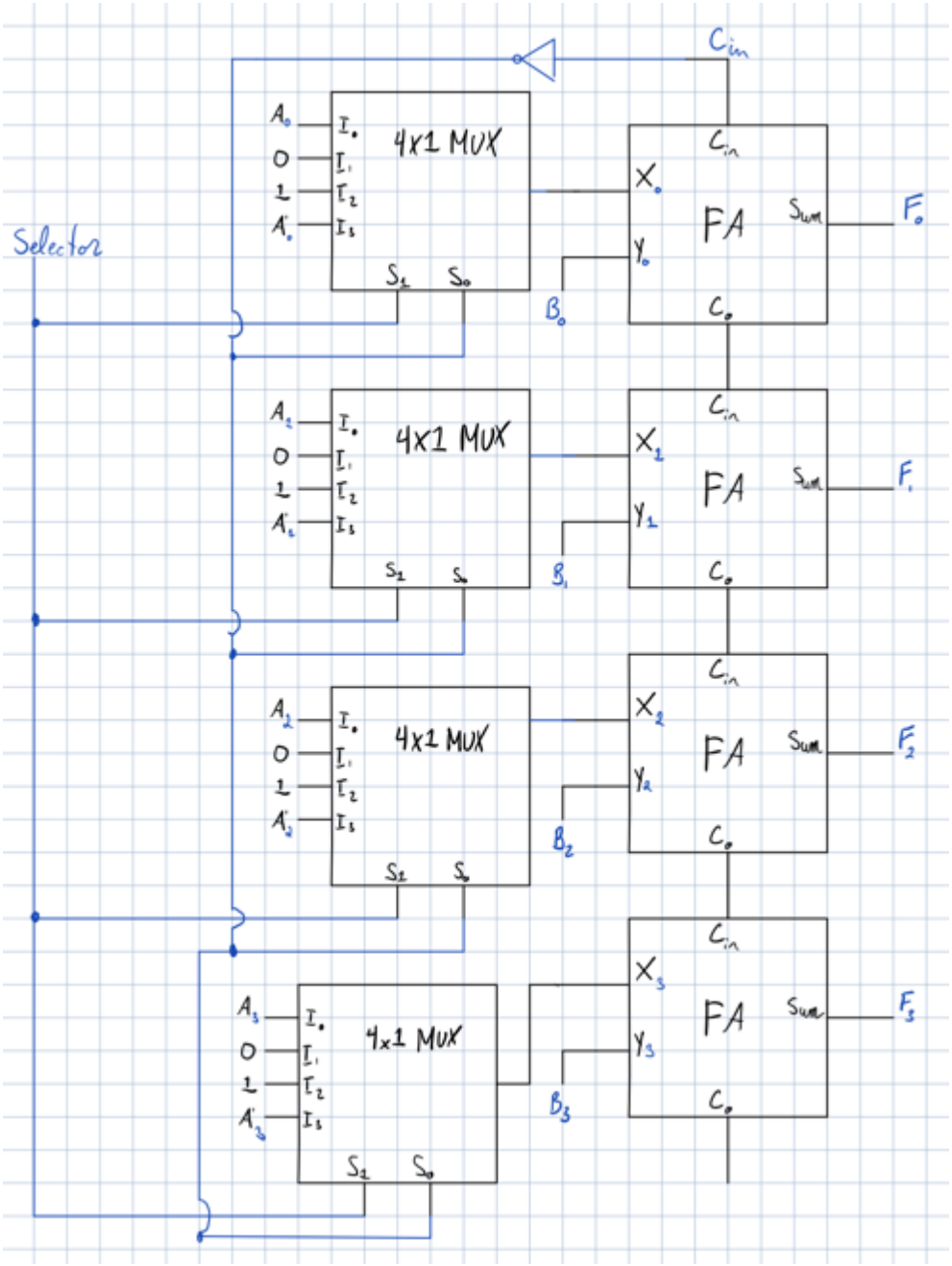
Therefore, X_i shares S and C_{in}'.


Selector
C_{in}
A_0
B_0
C_{in}
X_{in}
y_{in}
FA
Sum
C_o
F_0
A_1
B_1
C_{in}
X_{in}
y_{in}
FA
Sum
C_o
F_1
A_2
B_2
C_{in}
X_{in}
y_{in}
FA
Sum
C_o
F_2
A_3
B_3
C_{in}
X_{in}
y_{in}
FA
Sum
C_o
F_3

b. **Derive the arithmetic operations (formula and description by filling up the following table (do not forget to take into account C_{in}):**

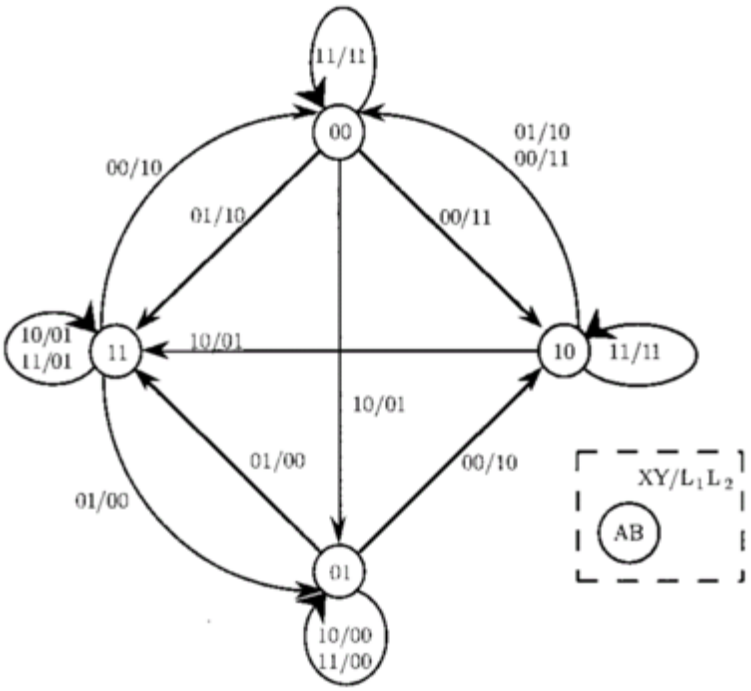
$S_{elector}$	C_{arryIn}	Calculation	Output Function	Operation
X	X	$X_i = A'_i S + A_i C'_{in}$	$F = X_i + B_i$	
0	0	$X_i = A'_i(0) + A_i(0)' = A_i$	$F = A_i + B_i$	Addition
0	1	$X_i = A'_i(0) + A_i(1)' = 0$	$F = B_i + 1$	Increment
1	0	$X_i = A'_i(1) + A_i(0)' = A'_i + A_i = 1$	$F = 1 + B_i + 1 = B_i + overflow = B_i - 1$	Decrement
1	1	$X_i = A'_i(1) + A_i(1)' = A'_i$	$F = A'_i + B_i = B_i - A_i$	Subtraction

We could also use four 4x1 MUX to provide the input for X. If the selectors of MUX i were S and C_{in} *+(in that order), then the data inputs would be $I_0 = A_i, I_1 = 0, I_3 = 1; I_4 = A'_i$



Tutorial 6:

1. **A finite state machine is described by the following transition diagram:**



   a. **What kind of state machine is represented in the state diagram, Moore or Mealy?**

Mealy Machine:
A Mealy machine is defined as a machine whose output values are determined by both its current state and current inputs. In this machine, at most one transition is possible

Mealy Machine Quick Facts:

- Output depends on present state as well as input.
- If input changes, output also changes.
- Less number of states required.
- There is less hardware requirement.
- They react faster to inputs.
- Asynchronous output generation.
- Output is placed on transitions.
- It is difficult to design.
- Input directly goes to the output logic gates.

Mealy Machine Example:

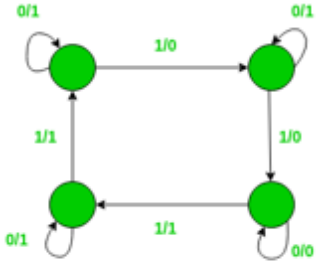


Figure - Mealy machine

Moore Machine:
A Moore machine is defined as a machine whose output values are determined only by its current state.

Moore Machine Quick Facts:

- Output depends only upon present state.
- If input changes, output does not change.
- More number of states are required.
- There is more hardware requirement.
- They react slower to inputs (one clock cycle later).
- Synchronous output and state generation.
- Output is placed on states.
- Easy to design.
- Input **does not** directly goes to the output logic gates.

Moore Machine Example:

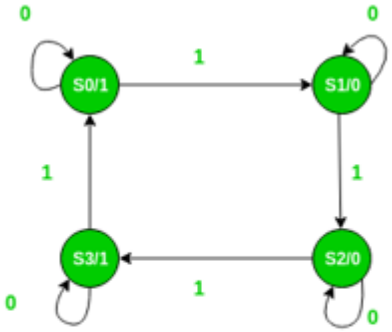


Figure - Moore machine

Side Note: [How to Convert from Mealy to Moore Machine]

Answer:
Since the circuit in the state diagram is asynchronous, (among other traits), we can deduce that it is of Mealy design.

b. Derive the machine excitation table.

Notes based on state diagram:
Present State Variables: A, B
Inputs: X, Y
Outputs: L_1, L_2
Next State: A^+, B^+

First derive the transition table:

Present State		Inputs		Next State		Outputs	
A	B	X	Y	A^+	B^+	L_1	L_2
0	0	0	0	1	0	1	1
0	0	0	1	1	1	1	0
0	0	1	0	0	1	0	1
0	0	1	1	0	0	1	1
0	1	0	0	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	0	0	1	1
1	0	0	1	0	0	1	0
1	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1
1	1	0	0	0	0	1	0
1	1	0	1	0	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	0	1

Notes about the PAL Schema in question c):
The configuration mostly used in a Sequential (or simple) programmable logic device (SPLD)is the combinational PAL together with D flip-flops. A PAL that includes flip-flops is referred to as a registered PAL, to signify that the device contains flip-flops in addition to the AND-OR array. Each section of an SPLD is called a microcell, which is a circuit that contains a SOP combinational logic function and an optional flip-flop. However, with negative logic using an inverter in the circuit, you could also optimize the inverse of the next states and outputs of the systems on the K-Map.

The output is driven by an edge-triggered D flip-flop, connected to a common clock input positive edge clock (since CLK = 1 and not inverted before being inputted into the flip-flop).

The output of the of the D flip-flop is connected to a three-state buffer (or in this case an inverter) controlled by an output-enable signal marked in the diagram as OE. The OE is inverted prior to be inputted into the three-state inverter, thereby creating an active low tri-state buffer. Therefore, the output is inverted if the OE is NOT equal to "1".
See more here: https://www.electronics-tutorials.ws/logic/logic_9.html

In summary, because the output of the D flip-flop is controlled by an active-low inverter, (ie. the output is inverted if the output-enable is not equal to "1"), then we must optimize the SOP of the maxterms for the next-states and outputs in terms present-states and inputs.

Optimize the inverse of the next states and outputs using K-Maps, in terms of the present state and inputs:

$$A^+ = \sum m(0,1,4,5,19,11,14,15); A^{+'} = \prod M(2,3,6,7,8,9,12,13)$$
$$B^+ = \sum m(1,2,5,6,7,10,13,14,15); B^{+'} = \prod M(0,3,4,8,9,11,12)$$
$$L_1 = \sum m(0,1,3,4,8,9,11,12); L_1 = \prod M(2,5,6,7,10,13,14,15)$$
$$L_2 = \sum m(0,2,3,8,10,11,14,15); L_2 = \prod M(1,4,5,6,7,9,12,13)$$

$$A^+ = \sum m(0,1,4,5,19,11,14,15); A^{+\prime} = \prod M(2,3,6,7,8,9,12,13)$$

AB/XY	00	01	11	10
00	1 0	1 1	0 3	0 2
01	1 4	1 5	0 7	0 6
11	0 12	0 13	1 15	1 14
10	0 8	0 9	1 11	1 10

$$A^{+\prime} = AX' + A'X$$

$$B^+ = \sum m(1,2,5,6,7,10,13,14,15); B^{+\prime} = \prod M(0,3,4,8,9,11,12)$$

AB/XY	00	01	11	10
00	0 0	1 1	0 3	1 2
01	0 4	1 5	1 7	1 6
11	0 12	1 13	1 15	1 14
10	0 8	0 9	0 11	1 10

$$B^{+\prime} = X'Y' + B'XY + \{AB'Y \text{ or } AB'X'\}$$

$$L_1 = \sum m(0,1,3,4,8,9,11,12); L_1' = \prod M(2,5,6,7,10,13,14,15)$$

AB/XY	00	01	11	10
00	1 (0)	1 (1)	1 (3)	0 (2)
01	1 (4)	0 (5)	0 (7)	0 (6)
11	1 (12)	0 (13)	0 (15)	0 (14)
10	1 (8)	1 (9)	1 (11)	0 (10)

$$L_1' = BY + XY$$

$$L_2 = \sum m(0,2,3,8,10,11,14,15); L_2' = \prod M(1,4,5,6,7,9,12,13)$$

AB/XY	00	01	11	10
00	1 (0)	0 (1)	1 (3)	1 (2)
01	0 (4)	0 (5)	0 (7)	0 (6)
11	0 (12)	0 (13)	1 (15)	1 (14)
10	1 (8)	0 (9)	1 (11)	1 (10)

$$L_2' = A'B + BX' + X'Y$$

Declare the characteristic and excitation table for a D Flip-Flop:

D Flip-Flop Characteristic Table		
D_Q	Q^+	Description
0	0	Reset
1	1	Set

D Flip-Flop Excitation Table		
Q	Q^+	D_Q
0	0	0
0	1	1
1	0	0
1	1	1
$Q^+ = D$		
$D = Q^+$		
$Q = Q^+D + Q^{+'}D' = Q^+ \odot D$		

Use the characteristic equation for the D Flip-Flop to find the D input equations:

$D = Q^+$

$D_A = A^+$

$D_A = AX^{'} + A'X$

$D_B = X^{'}Y^{'} + B^{'}XY + \{AB^{'}Y \text{ or } AB^{'}X^{'}\}$

$D_B = X^{'}Y^{'} + B^{'}XY + AB'Y$

[optional] Fill in the state table with the values of the D flip-flop inputs:

Present State		Inputs		Next State		Outputs		D Flip-Flop Inputs	
A	B	X	Y	A^+	B^+	L_1	L_2	D_A	D_B
0	0	0	0	1	0	1	1	1	0
0	0	0	1	1	1	1	0	1	1
0	0	1	0	0	1	0	1	0	1
0	0	1	1	0	0	1	1	0	0
0	1	0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	0	1	1
0	1	1	0	0	1	0	0	0	1
0	1	1	1	0	1	0	0	0	1
1	0	0	0	0	0	1	1	0	0
1	0	0	1	0	0	1	0	0	0
1	0	1	0	1	1	0	1	1	1
1	0	1	1	1	0	1	1	1	0
1	1	0	0	0	0	1	0	0	0
1	1	0	1	0	1	0	0	0	1
1	1	1	0	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1

Fill in the PAL Programming Table:

Summary of Equations:

$A^{+'} = AX' + A'X$

$B^{+'} = X'Y' + B'XY + AB'Y$

$L_1' = BY + XY$

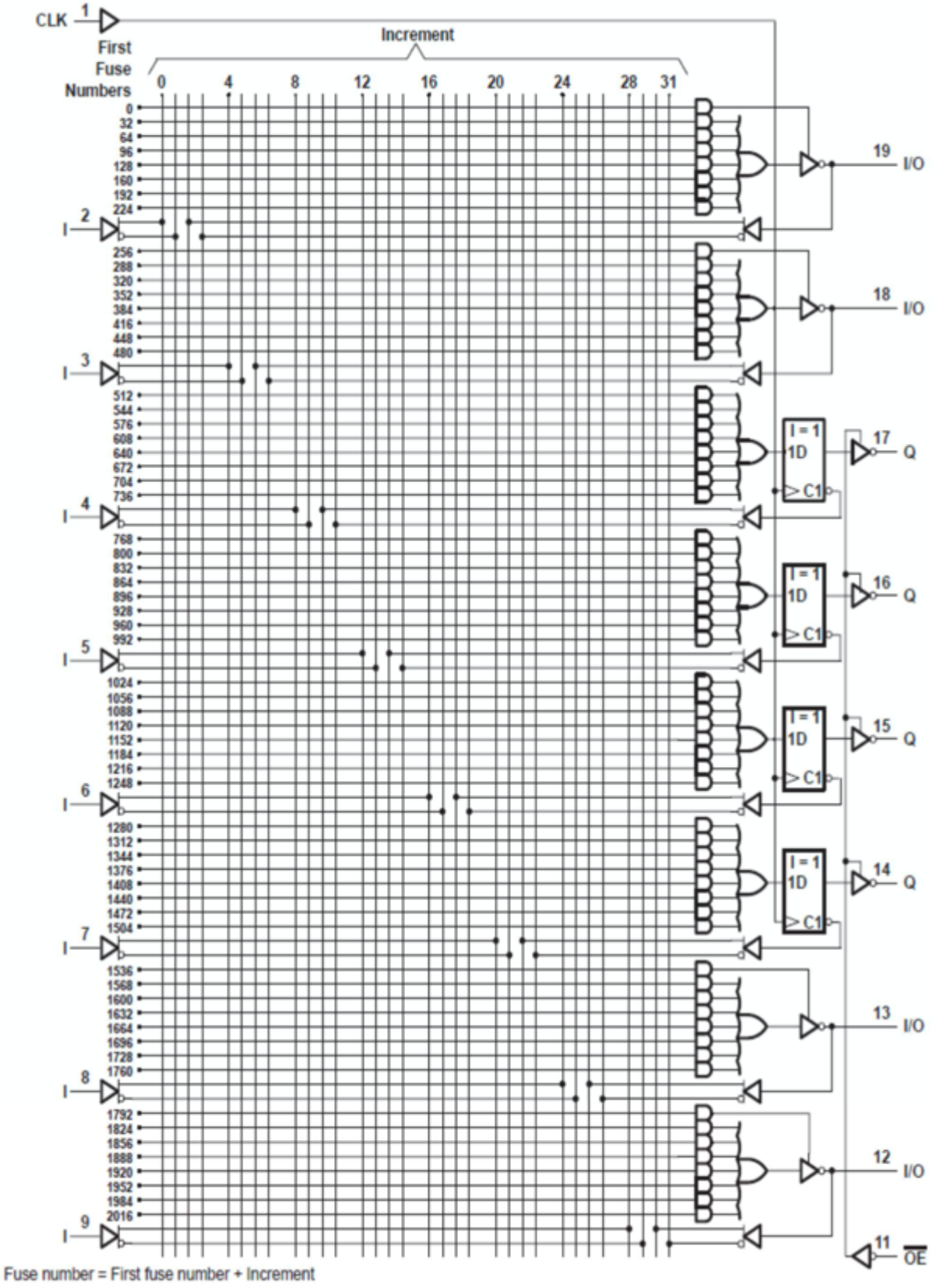
$L_2' = A'B + BX' + X'Y$

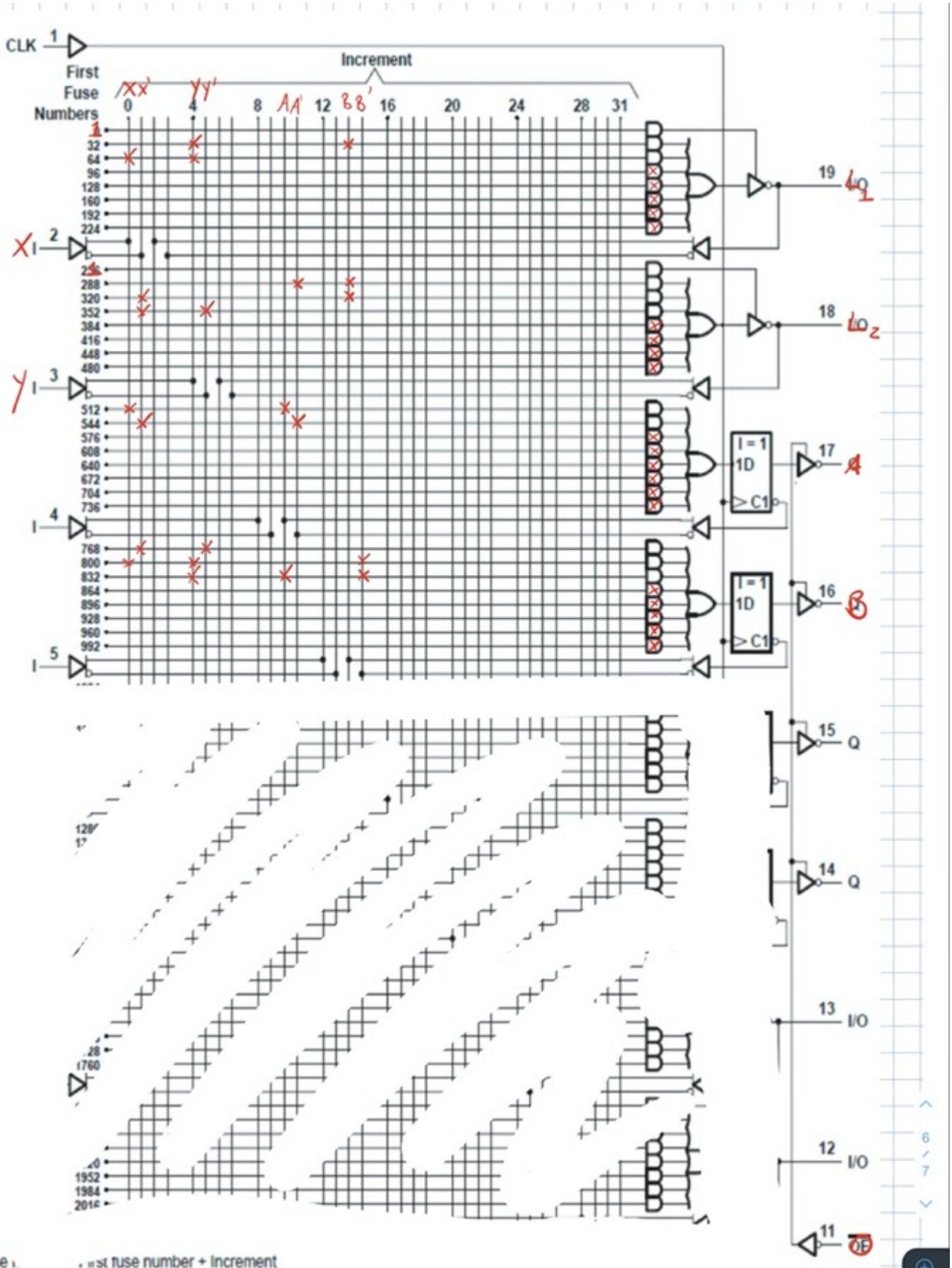
$D_A = AX^{'} + A'X$

$D_B = X^{'}Y^{'} + B^{'}XY + AB^{'}Y$

Product Term		Inputs				Outputs			
		A	B	X	Y	A^+	B^+	L_1	L_2
1	AX’	1		0		1			
2	A’X	0		1		1			
3	X’Y’	0			0		1		
4	B’XY		0	1	1		1		
5	AB’Y	1	0		1		1		
6	BY		1		1			1	
7	XY			1	1			1	
8	A’B	0	1						1
9	BX’		1	0					1
10	X’Y		0	1					1

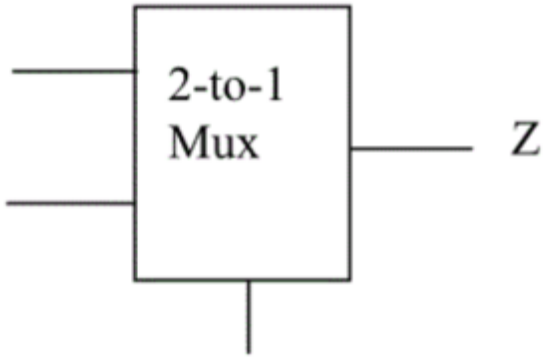
c. Implement the machine state using the following PAL schema. Clearly indicate the enabled components versus the disabled ones.





Fuse ... first fuse number + Increment

2. **Show (by appropriately labelling the inputs) how a 2-to-1 multiplexer can be used as a 2-input OR gate that implements $Z = A + B$:**



Describe the function characteristics based on the function equation:

$$Z = A + B$$

Use a truth table to show all of the possible outputs for Z:

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

So the function is "1" when A or B is "1".

Choose the Selection Line:

The selection line can be A or B.

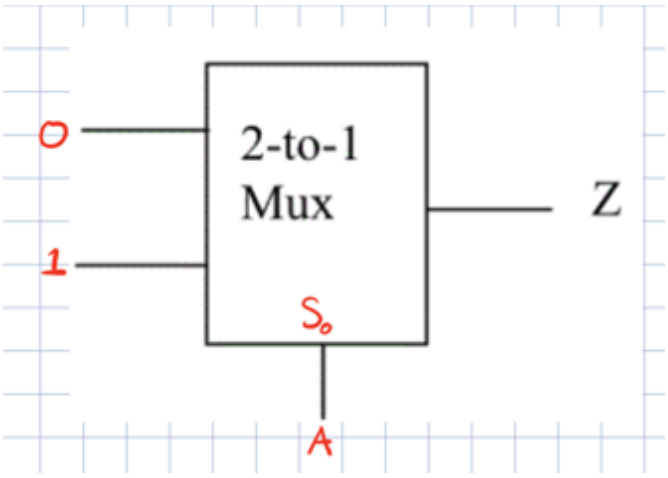
Selection Line = A

Based on the Selection Line, what are the inputs available for the MUX to select:

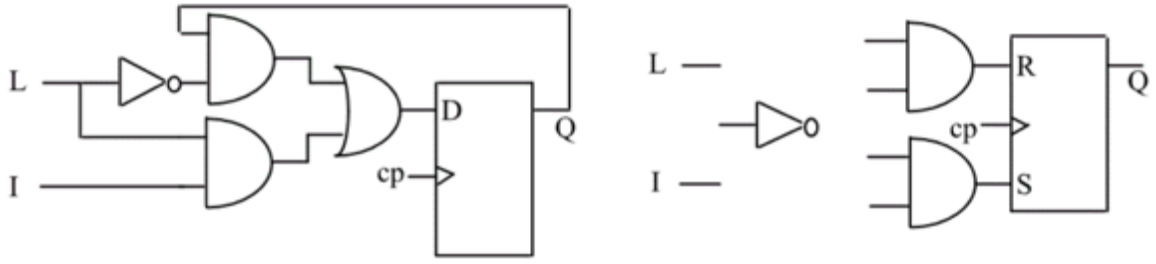
The MUX inputs are all of the values of when the function Z is "1" regardless of the value of A.

$I_0 = B$ *or* 1

$I_1 = B$ *or* 0



3. **Consider the circuit on the left that corresponds to one bit of a register with parallel load:**
   When $L = 0, Q^+ = Q$; when $L = 1, Q^+ = I$.
   If the D Flip-Flop is replaced by an SR Flip-Flop, the circuit on the right must be obtained.
   Complete the circuit on the right such that:
   When $L = 0, Q^+ = Q$; when $L = 1, Q^+ = I$



Make a

Declare the characteristic equation for a D Flip-Flop:
$Q^+ = D$

Replace D with the D input equation based on the logic circuits:
$Q^+ = QL' + LI$
$\mathrm{D} = QL' + LI$

Declare the state table based on the description of the SR circuit in the question:

State Table					
L	**I**	Q	Q^+	S_Q	R_Q
0	0	0	0		
0	0	1	1		
0	1	0	0		
0	1	1	1		
1	0	0	0		
1	0	1	0		
1	1	0	1		
1	1	1	1		

Declare the excitation table for an SR Flip-Flop

SR Flip-Flop Excitation Table			
Q	Q^+	S_Q	R_Q
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0
$Q^+ = S + R'Q; S \times R = 0$			

Based on the excitation table for a SR flip-flop, fill in the input values for the S and R inputs in the state table:

State Table					
L	**I**	Q	Q^+	S_Q	R_Q
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	1	X	0
1	0	0	0	0	X
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	1	X	0

Use K-Maps to optimize the equations for the S and R inputs:

$$S = \sum m(6); S' = \prod M(0,2,4,5)$$

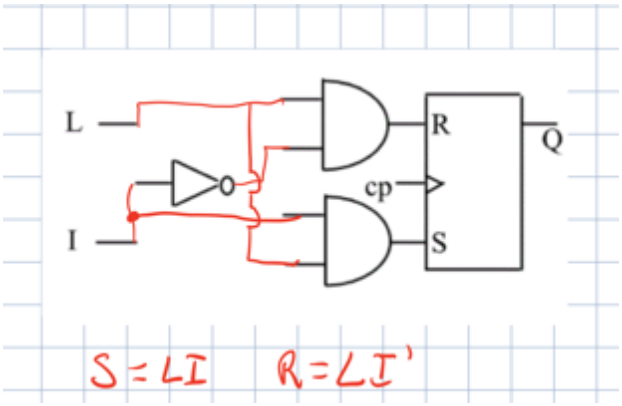
L/IQ	00	01	11	10
0	0 (0)	X (1)	X (3)	0 (2)
1	0 (4)	0 (5)	X (7)	1 (6)

$$S = LI$$

$$R = \sum m(5); R' = \prod M(1,3,6,7)$$

L/IQ	00	01	11	10
0	X (0)	0 (1)	0 (3)	X (2)
1	X (4)	1 (5)	0 (7)	0 (6)

$$R = LI'$$



[answer is different from posted solution on BrightSpace]

4. **Derive the optimal logic expression of the JK flip-flop inputs for the synchronous sequential circuit according to:**

A	B	C	A^+	B^+	C^+
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
1	0	0	0	1	0
1	1	0	0	0	0

Declare the Excitation Table of a JK Flip-Flop:

JK Flip-Flop Excitation Table			
Q	Q^+	J_Q	K_Q
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0
$Q^+ = JQ' + K'Q$			

Derive the State Table with the JK inputs based on the JK Flip-Flop Excitation Table.
Ensure to include the unused states:

A	B	C	A^+	B^+	C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	1	1	0	1	X	1	X	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	1	1	X	X	X	X	X	X	X	X	X
1	0	0	0	1	0	X	1	1	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	X	X	X	X	X	X	X	X	X

Derive the JK input equations in terms of A, B, C:

$$J_A = \sum m(0,1)\,; J'_A = \prod M(2)$$

A/BC	00	01	11	10
0	1 (0)	1 (1)	X (3)	0 (2)
1	X (4)	X (5)	X (7)	X (6)

$J_A = B'$
$K_A = 1$

$$J_B = \sum m(1,4)\,; J'_B = \prod M(0)$$

A/BC	00	01	11	10
0	0 (0)	1 (1)	X (3)	X (2)
1	1 (4)	X (5)	X (7)	X (6)

$J_B = A + C$
$K_B = 1$

$$J_C = \sum m(2)\,; J'_C = \prod M(0,4,6)$$

A/BC	00	01	11	10
0	0 (0)	X (1)	X (3)	1 (2)
1	0 (4)	X (5)	X (7)	0 (6)

$J_C = A'B$
$K_C = 1$

Fill in the "Don't Care" values for the JK inputs in the State Table:

$J_A = B'$
$K_A = 1$
$J_B = A + C$
$K_B = 1$
$J_C = A'B$
$K_C = 1$

$Q^+ = JQ' + K'Q$
$A^+ = (B')A' + (1)'A = A'B'$
$B^+ = (B')B' + (1)'B = B'$
$C^+ = (A'B)C' + (1)'C = A'BC'$

A	B	C	A^+	B^+	C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	1	1	0	1	X	1	X	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	1	1	X	X	X	X	X	X	X	X	X
1	0	0	0	1	0	X	1	1	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	X	X	X	X	X	X	X	X	X

Fill in the Next States based on the excitation equations:

$$A^+ = (B')A' + (1)'A = A'B'$$
$$B^+ = (B')B' + (1)'B = B'$$
$$C^+ = (A'B)C' + (1)'C = A'BC'$$

$$J_A = B'$$
$$K_A = 1$$
$$J_B = A + C$$
$$K_B = 1$$
$$J_C = A'B$$
$$K_C = 1$$

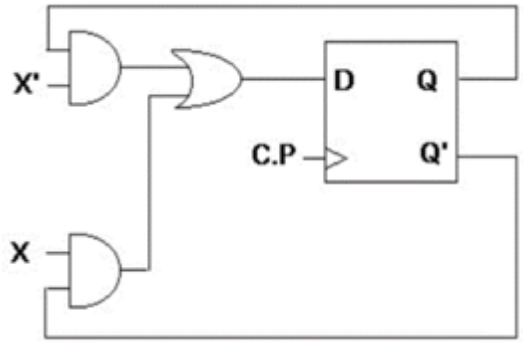
JK Flip-Flop Excitation Table			
Q	Q^+	J_Q	K_Q
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0
$Q^+ = JQ' + K'Q$			

Separate the State Diagram between used and unused states:

A	B	C	A^+	B^+	C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	1	1	0	1	X	1	X	X	1
0	1	0	0	0	1	0	X	X	1	1	X
1	0	0	0	1	0	X	1	1	X	0	X
1	1	0	0	0	0	X	1	X	1	0	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	1	0	1	0	X	1	1	X	X	1
1	1	1	0	0	0	X	1	X	1	X	1

The circuit is valid since the unused states transition to used states within one clock transition.

5. **Determine the flip flop input of the circuit shown below. What flip flop could be used instead if we did not want any of the gates included. (Draw the new circuit).**



First next state equation based on the logic circuit diagram:

$D = X'Q + XQ' = X \oplus Q$

$Q^+ = D = X \oplus Q$

Which Flip Flop shares the same next state equation as the logic circuit in the question:

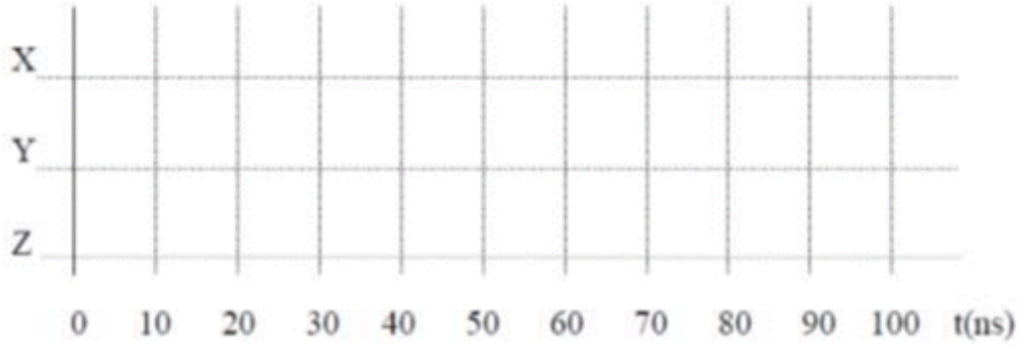
$Q_T^+ = QT' + Q'T = Q \oplus T$

$Q_{JK}^+ = JQ' + K'Q$

Therefore, instead of a D flip-flop, a T flip-flop could be used where $Q_T^+ = QX' + Q'X = X \oplus Q$, or a JK flip-flop could be used where $Q_{JK}^+ = XQ' + X'Q = X \oplus Q$.

Tutorial 7:

1. **Assume that the inverter in the network below has a propagation delay of 5 ns and the AND gate has a propagation delay of 10 ns. Draw a timing diagram for the network showing $X, Y, and\ Z$. Assume that X is initially 0, Y is initially 1, X becomes 1 for 80 ns, and then X is 0 again.**



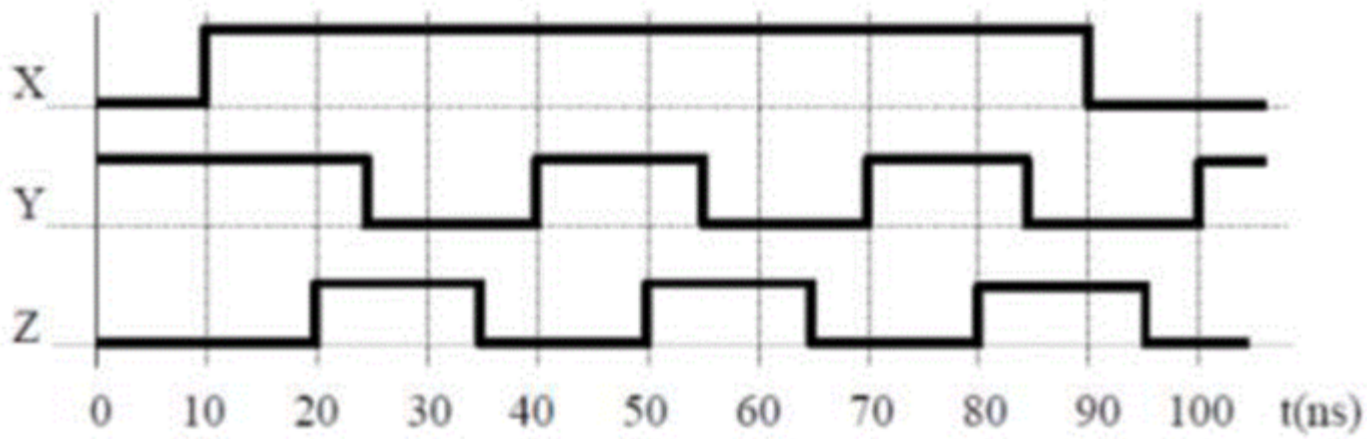
First derive the output equation:

$Z = XY'$

Declare the propagation traits based on the question description:

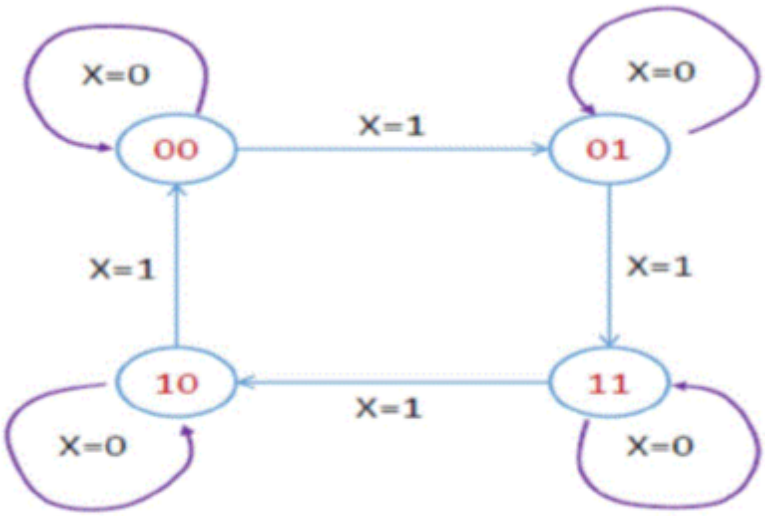
X has no propagation delay.
Y has a propagation delay of 5 ns due to the inverter.
Z has a propagation delay of 15 ns due to the 5 ns delay in the inverter and the 10 ns delay in the AND gate.

Plot the timing diagram:



2. Design a sequential circuit with two JK flip-flops A and B, and one input x.
   When $x = 0$, the state of the circuit remains the same.
   When $x = 1$, the circuit goes through the state transitions from 00 *to* 01 *to* 11 *to* 10 *back to* 00, and repeats.

   a. Draw the state diagram of the above system.



State Diagram

A	B	x	A^+	B^+
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

b. Derive the state transition table of the system.

A	B	x	A^+	B^+	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	0	1	0	X	X	0
0	1	1	1	1	1	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	0	0	X	1	0	X
1	1	0	1	1	X	0	X	0
1	1	1	1	0	X	0	X	1

JK Flip-Flop Excitation Table			
Q	Q^+	J_Q	K_Q
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0
$Q^+ = JQ' + K'Q$			

c. **Derive the optimized SOPs of the JK FFs.**

$$J_A = \sum m(3)\,;J'_A = \prod M(0,1,2)$$

A/Bx	00	01	11	10
0	0 (0)	0 (1)	1 (3)	0 (2)
1	X (4)	X (5)	X (7)	X (6)

$$J_A = Bx'$$

$$K_A = \sum m(5)\,;K'_A = \prod M(4,6,7)$$

A/Bx	00	01	11	10
0	X (0)	X (1)	X (3)	X (2)
1	0 (4)	1 (5)	0 (7)	0 (6)

$$K_A = B'x$$

$$A^+ = J_A A' + K'_A A$$
$$A^+ = (Bx')A' + (B'x)'A$$
$$A^+ = A'Bx' + A(B + x')$$
$$A^+ = A'Bx' + AB + Ax'$$

$$J_B = \sum m(1)\,; J_B' = \prod M(0,4,5)$$

A/Bx	00	01	11	10
0	0 (0)	1 (1)	X (3)	X (2)
1	0 (4)	0 (5)	X (7)	X (6)

$$J_B = A'x$$

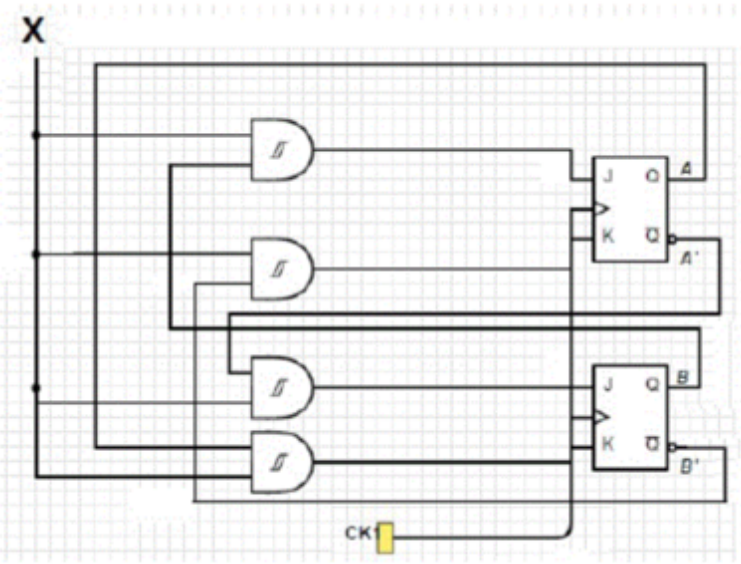
$$K_B = \sum m(7)\,; K_B' = \prod M(2,3,6)$$

A/Bx	00	01	11	10
0	X (0)	X (1)	0 (3)	0 (2)
1	X (4)	X (5)	1 (7)	0 (6)

$$K_B = Ax$$

$$B^+ = J_B B' + K_B' B$$
$$B^+ = (A'x)B' + (Ax)'B$$
$$B^+ = A'B'x + B(A' + x')$$
$$B^+ = A'B'x + A'B + Bx'$$

d. Draw the logic diagram.



3. A PN flip-flop has 4 operations:
   Clear to 0
   No change
   Complement and Set to 1 when inputs P and N are 00, 01, 10, and 11 respectively.

   a. Derive the PN characteristic table.

P	N	Q^+	Description
0	0	0	Reset
0	1	Q	No Change
1	0	Q'	Complement
1	1	1	Set to 1

   b. Derive the PN characteristic equation.

P	N	Q	Q^+
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$Q^+ = \sum m(3,4,6,7);\ Q^{+\prime} = \prod M(0,1,2,5)$$

P/NQ	00	01	11	10
0	0 (0)	0 (1)	1 (3)	0 (2)
1	1 (4)	0 (5)	1 (7)	1 (6)

$$Q^+ = PQ' + NQ$$

c. Derive the PN excitation table.

P	N	Q(t)	Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Excitation table

Q(t)	Q(t+1)	P	N
0	0	0	x
0	1	1	x
1	0	x	0
1	1	x	1

Q	Q^+	P	N
0	0	0	X
0	1	1	X
1	0	X	0
1	1	X	1

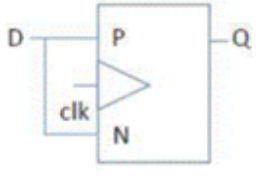
d. How to convert it to a D flip-flop.

D Flip-Flop Excitation Table		
Q	Q^+	D_Q
0	0	0
0	1	1
1	0	0
1	1	1
$Q^+ = D$		
$D = Q^+$		
$Q = Q^+D + Q^{+'}D' = Q^+ \odot D$		

P	N	Q	Q^+	D
0	X	0	0	0
1	X	0	1	1
X	0	1	0	0
X	1	1	1	1

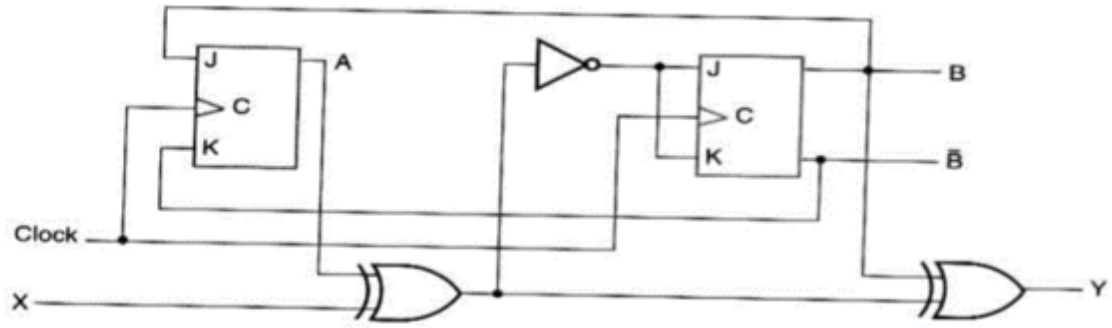
D	Q^+
0	0
1	1

$D = P = N = Q^+$



S

4. **Derive the transition table and diagram based on the system described by the logic circuit below:**



$Y = (A \oplus X) \oplus B$
$J_A = B$
$K_A = B'$

$J_B = (A \oplus X)' = A \odot X$
$K_B = (A \oplus X)' = A \odot X$

$Q^+ = JQ' + K'Q$
$A^+ = (B)A' + (B')'A = A'B + AB = B(A + A') = B$

$B^+ = (A \odot X)B' + (A \odot X)'B$
$B^+ = B \oplus (A \odot X)$

Present State		Input	Next State		FF Inputs				Outputs
A	B	x	A^+	B^+	J_A	K_A	J_B	K_B	Y
0	0	0	0	1	0	X	1	X	0
0	0	1	0	0	0	X	0	X	1
0	1	0	1	0	1	X	X	1	1
0	1	1	1	1	1	X	X	0	0
1	0	0	0	0	X	1	0	X	1
1	0	1	0	1	X	1	1	X	0
1	1	0	1	1	X	0	X	0	0
1	1	1	1	0	X	0	X	1	1

Present State		Input	Next State		FF Inputs		Outputs
A	B	x	A^+	B^+	T_A	T_B	Y
0	0	0	0	1	0	1	0
0	0	1	0	0	0	0	1
0	1	0	1	0	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	1
1	0	1	0	1	1	1	0
1	1	0	1	1	0	0	0
1	1	1	1	0	0	1	1

$$Q^+ = JQ' + K'Q$$
$$Q^+ = QT' + Q'T$$

$$T_A = \sum m\,(2,3,4,5)$$

A/Bx	00	01	11	10
0	0	1	1 3	1 2
1	1 4	1 5	7	6

$$A_T^+ = AB' + A'B = A \oplus B$$

$$T_B = \sum m\,(0,2,5,6)$$

A/Bx	00	01	11	10
0	1 0	1	3	1 2
1	4	1 5	7	1 6

5. **We wish to design a sequence detector circuit, which detects 3 or more consecutive 1's in a string of bits coming through an input line.**

    a. **Derive its state diagram of its Moore model implementation.**

Steps:
https://codestall.wordpress.com/2017/10/22/sequence-detection-of-3-1s-or-more-using-moore-model/

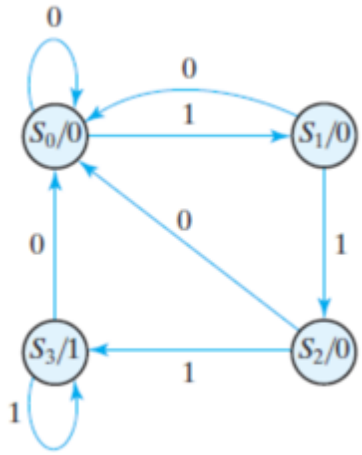
Moore Machine:
A Moore machine is defined as a machine whose output values are determined only by its current state.

There are 4 States, 1 input, 1 output:

$S_0 = 00$
$S_1 = 01$
$S_2 = 10$
$S_3 = 11$

$Y = 1$ when $x = 1$ three consecutive time in a row. When $x = 0$, the next state goes to $S_0 = 00$.

A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	0	0	1



b. Tabulate the state transition table of the system.

A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	0	0	1

c. Implement the system using D Flip-flops.

A	B	x	A^+	B^+	D_A	D_B	y
0	0	0	0	0			0
0	0	1	0	1			0
0	1	0	0	0			0
0	1	1	1	0			0
1	0	0	0	0			0
1	0	1	1	1			0
1	1	0	0	0			1
1	1	1	1	1			1

$$A^+ = \sum m(3,5,7);\ A^{+\prime} = \prod M(0,1,2,4,6)$$

A/Bx	00	01	11	10
0	0 (0)	0 (1)	1 (3)	0 (2)
1	0 (4)	1 (5)	1 (7)	0 (6)

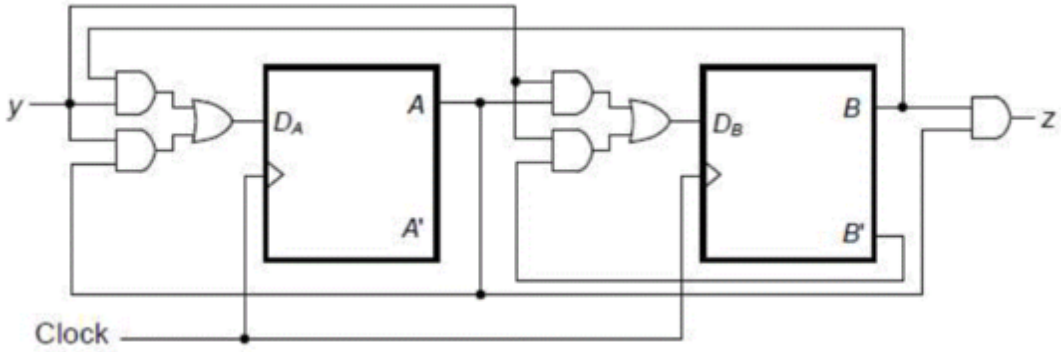
$$A^+ = Ax + Bx = x(A + B)$$

$B^+ = \sum m(1,5,7);\ B^{+\prime} = \prod M(0,2,3,4,6)$

A/Bx	00	01	11	10
0	0 (0)	1 (1)	0 (3)	0 (2)
1	0 (4)	1 (5)	1 (7)	0 (6)

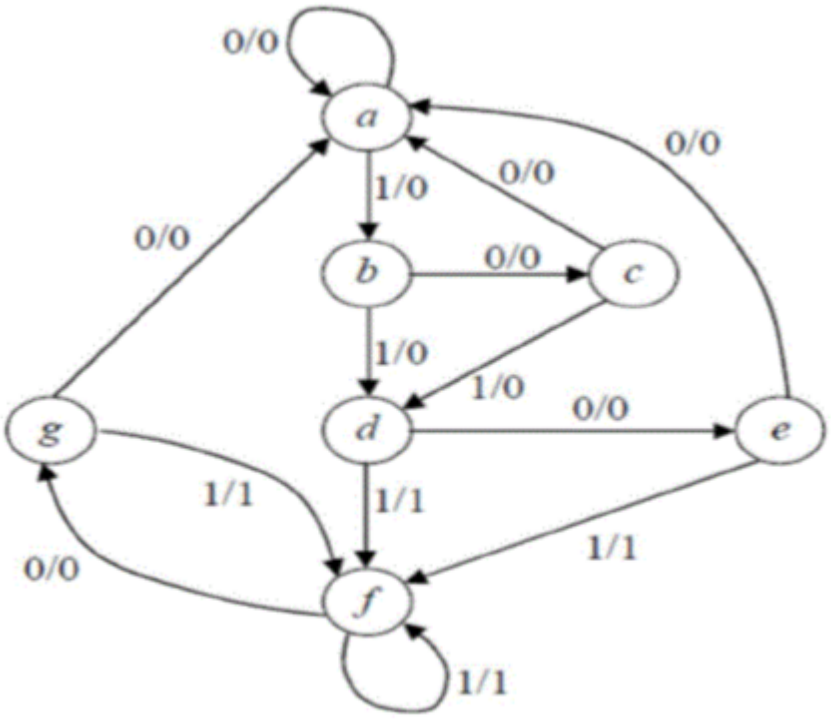
$B^+ = B'x + Ax$

$D = Q^+$
$A^+ = x(A + B)$
$B^+ = B'x + Ax$



Tutorial 8:

1. The state diagram of a sequential circuit is given as below:



a. Tabulate the related state table.

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

b. Reduce the state table to a minimum number of states using the implication table.

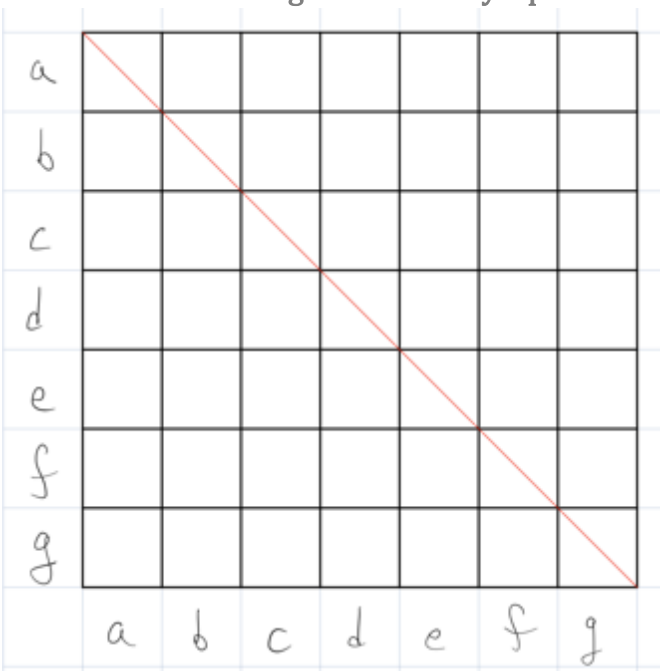
https://www.youtube.com/watch?v=L_KdLTMtkCg

See also how to establish equivalent sequential circuits using an implication table:
https://www.youtube.com/watch?v=U3zC7GogxmA

Two states that have different outputs cannot be equivalent, so we can put a cross in their corresponding intersection cells in the implication table:

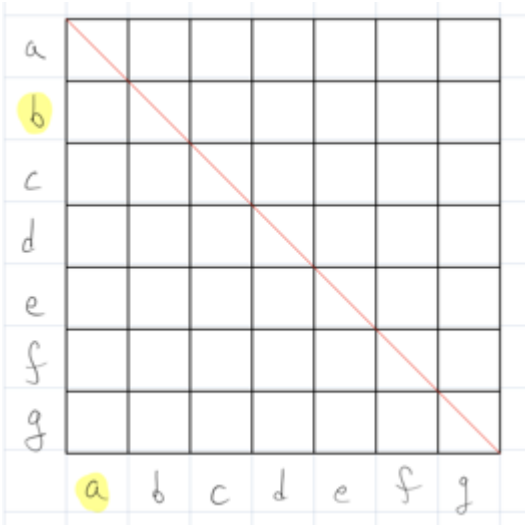
Notes:
If we want two states to be equivalent, then their outputs should be the same.

First, draw a chart with $x \times y$ rows where x and y equal the inputs. Then draw a line between the top left corner and bottom right corner. Any squares with a line through it will be ignored.



Next, look at the truth table and compare the present states for each column and row. Let's do the first few together:

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1



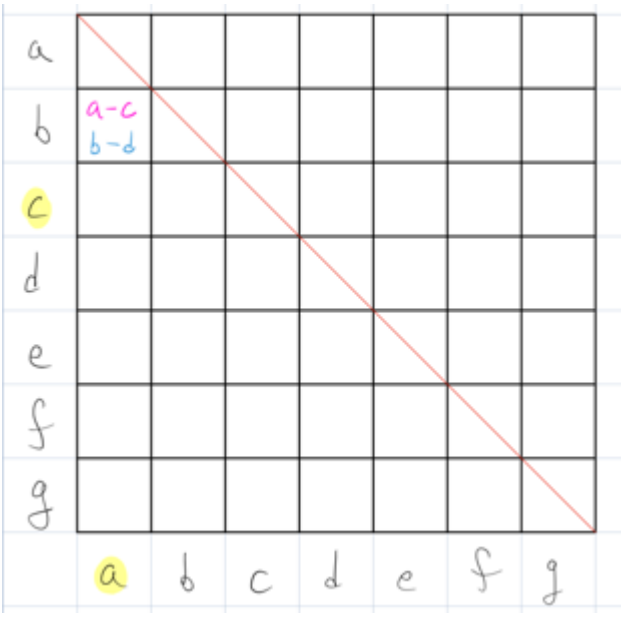
If the output for the present states a and b are equal values for when the input $x = 0$ **AND** $x = 1$, then we would write the corresponding next state values for a *and* b in the square: (a, b).

So, for the present states a, b:
When $x = 0$, the output for a, b are both 0.
When $x = 1$, the output for a, b are both 0.

Since the output values for a, b are then same when $x = 0$ *and* $x = 1$, we can write the respective next state values in the square (a, b) when $x = 0$ and $x = 1$.

When $x = 0$, the next state values for a, b is $a - c$. Write $a - c$ in the (a, b) square.
When $x = 1$, the next state values for a, b is $b - d$. Write $b - d$ in the (a, b) square.

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

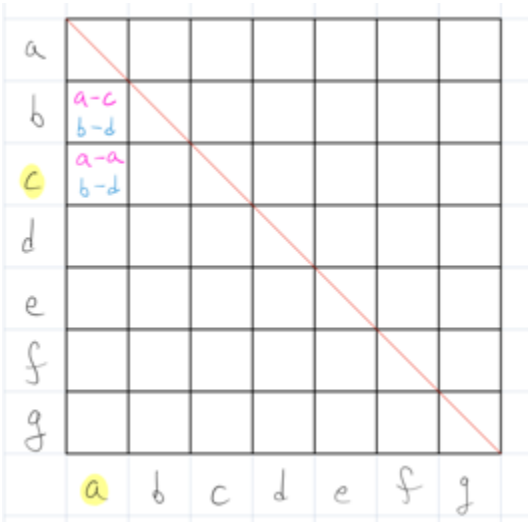


Continue for the present state a, c:

For the present states a, c:
When $x = 0$, the output for a, c are both 0.
When $x = 1$, the output for a, c are both 0.

Since the output values for a, c are then same when $x = 0$ *and* $x = 1$:
When $x = 0$, the output for a, a are both 0. We then write $a - a$ in the (a, c) square.
When $x = 1$, the output for b, d are both 0. We then write $b - d$ in the (a, c) square.

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

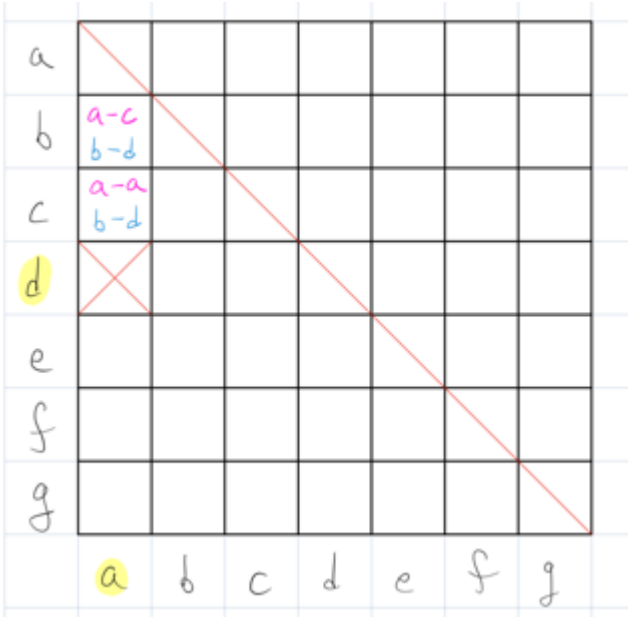


Continue for the present state a, d:

For the present states a, c:
When $x = 0$, the output for a, d are both 0.
When $x = 1$, the output for a, d are **NOT** equal.

Since the output values for a, d are **NOT** equal:
Write an "X" through the square, (a, d).

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

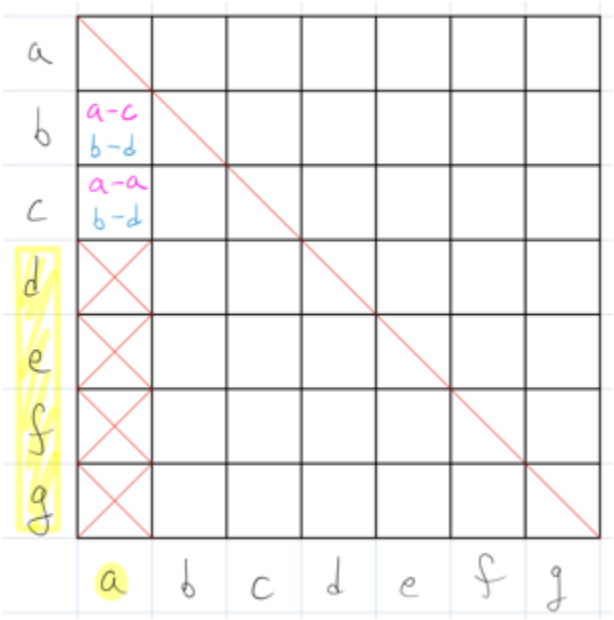


Continue for the rest of the present states in column a:

The present state a does not share any equal outputs with present states d through g.

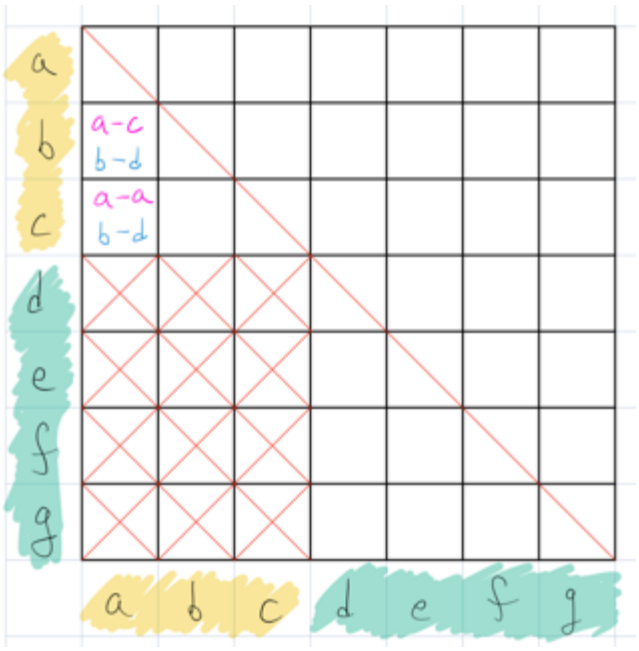
[Note: It is sometimes easier "X" the non-equal present states first.]

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1



Continue to cross-out the non-equal present states in the implication chart for all columns:
Present states a through c all share equal output values.
Present states d through g all share equal output values.
Present states a through c DO NOT share equal output values with present states d through g.

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

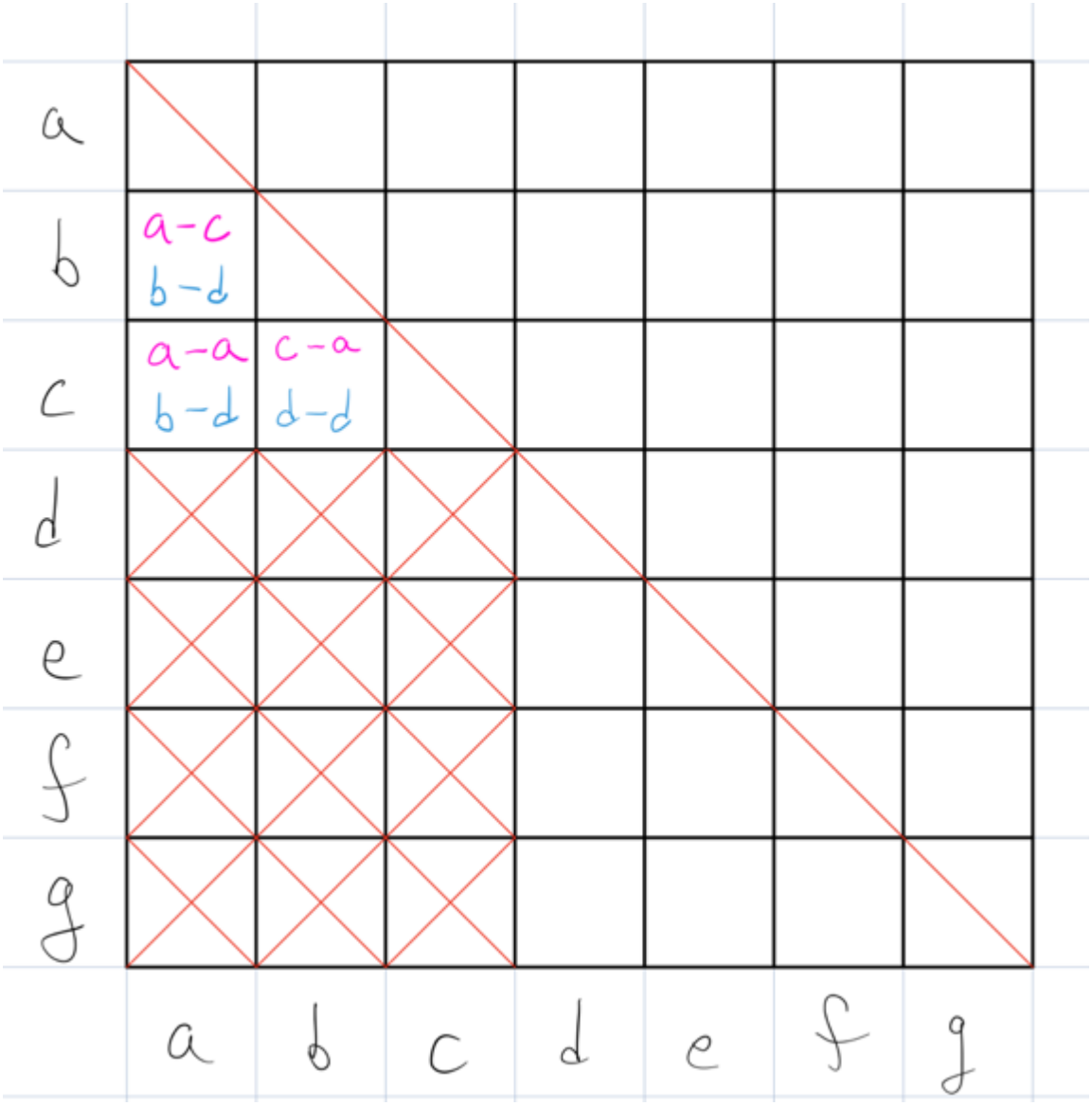


Continue for the rest of the present states in column b:

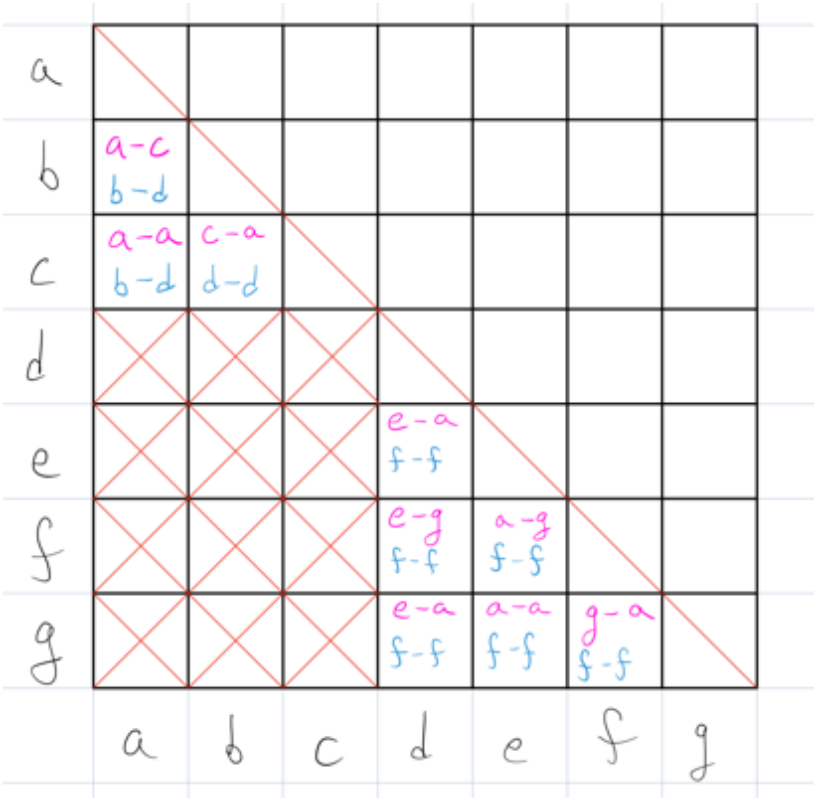
For the present states b, c:
When $x = 0$, the output for b, c are both 0.
When $x = 1$, the output for b, c are both 0.

Since the output values for b, c are then same when $x = 0$ *and* $x = 1$:
When $x = 0$, the output for c, a are both 0. We then write $c - a$ in the (b, c) square.
When $x = 1$, the output for d, d are both 0. We then write $d - d$ in the (b, c) square.

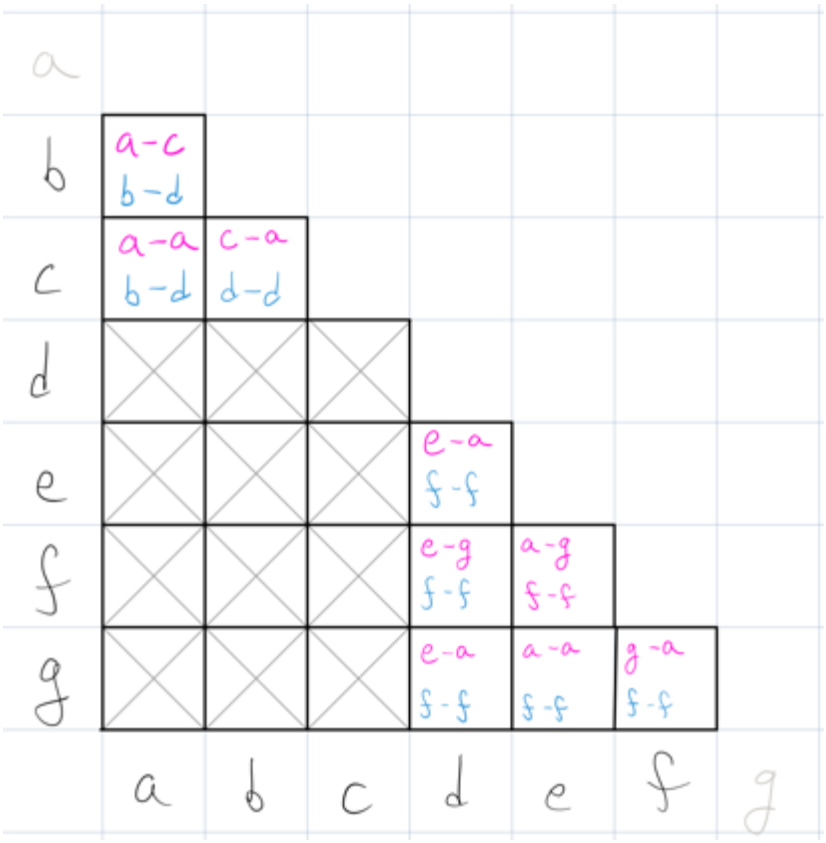
Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1



Continue for the rest of the present states in columns d through f:



You can also make the implication chart with squares just enough to fill the amount needed below the diagonal line:



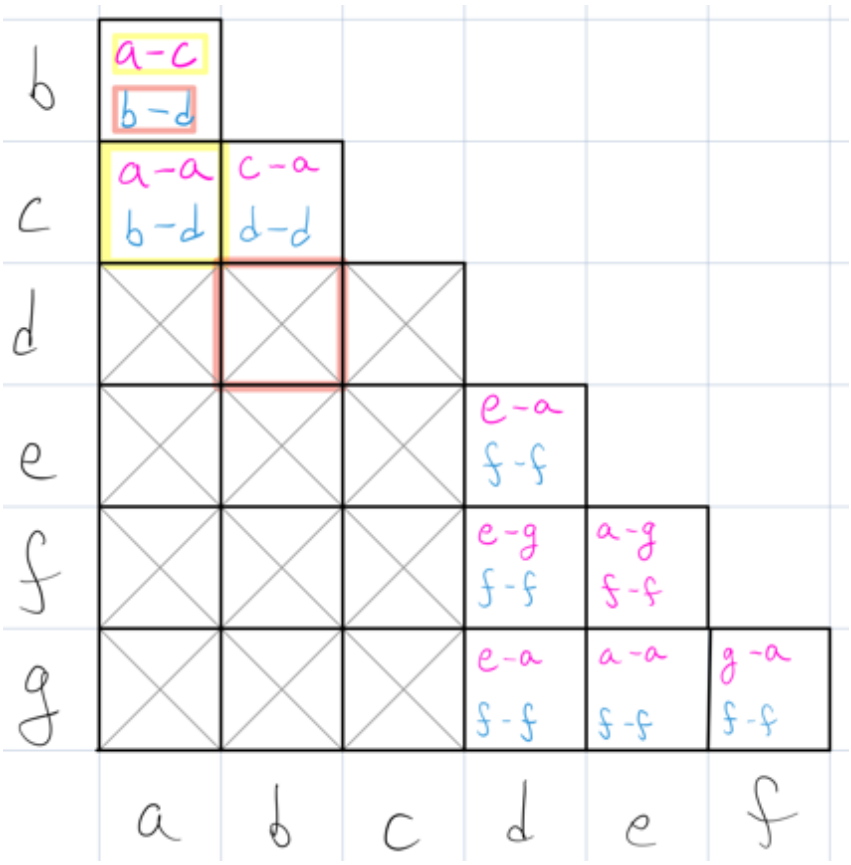
Now that the implication chart is all filled in as per the state table, we need to investigate equal states that cannot occur due to the unequal states (as indicated with the closed squares with a cross through it).

Compare the values in any open square (in any combination) with the coordinates of those values on the implication chart.
If the coordinate on the implication chart is closed "with a cross through it", put a cross through that open square with the respective next-state values

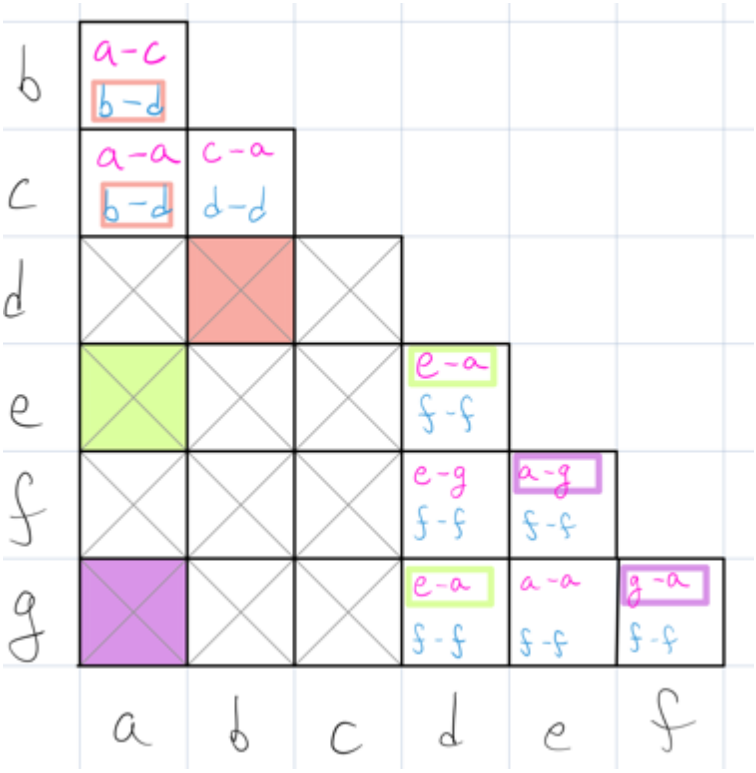
Square (a, b) has the equal next-state values of $a - c$, and $b - d$, which is the same as $c - a$, and $d - b$.

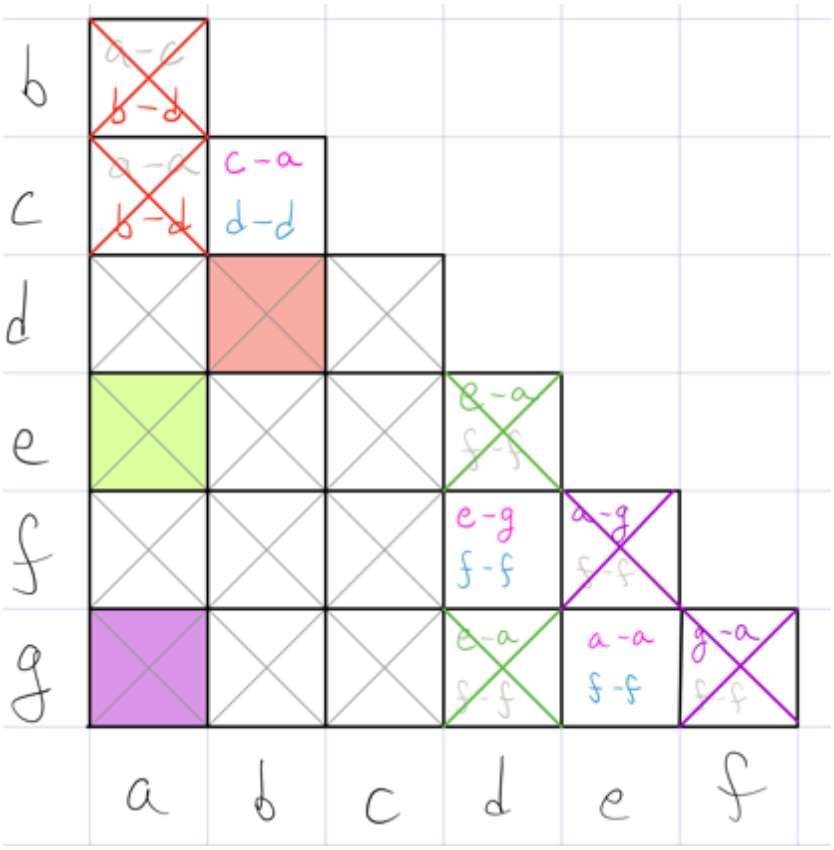
The square at coordinate (a, c) is an open square, so the square *could* remain open.
The square at coordinate (b, d) is a **CLOSED** square, so we must put a cross through the square $(a\ b)$.

Note:
If you find that the square outside of the bounds of the chart (ie. outside of beneath the diagonal), such as (c, a) or (d, b), this does not mean that the square should be closed. Try reverse the order of the coordinate: (a, c) ; (b, d). If the square is still outside of the bounds of the chart, keep it open (if there are no closed states), and move on. An example would be next state $f - f$ and coordinate (f, f).

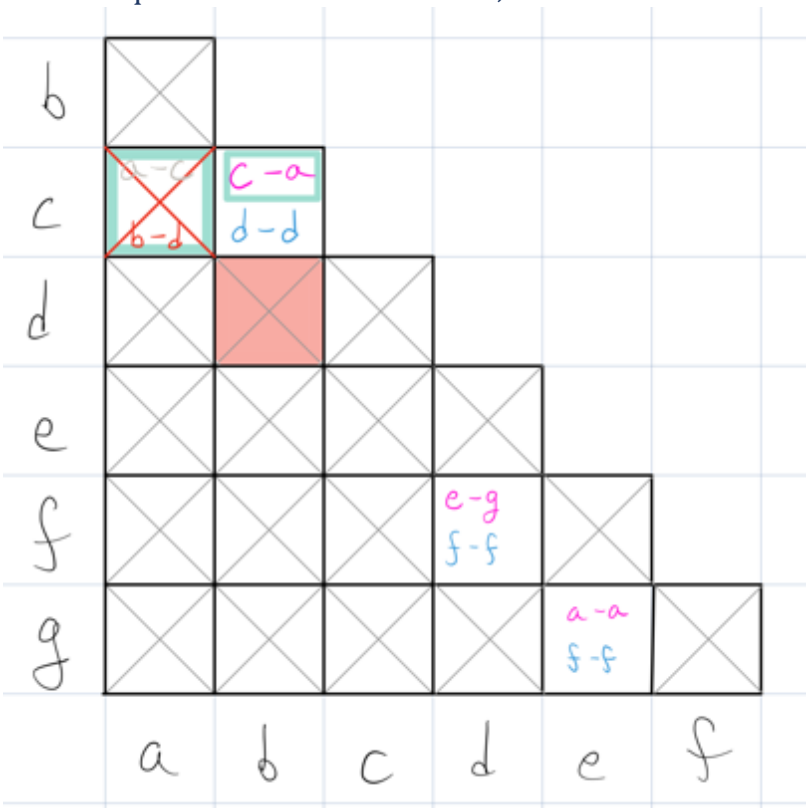


Continue for the rest of the open squares on the implication chart:

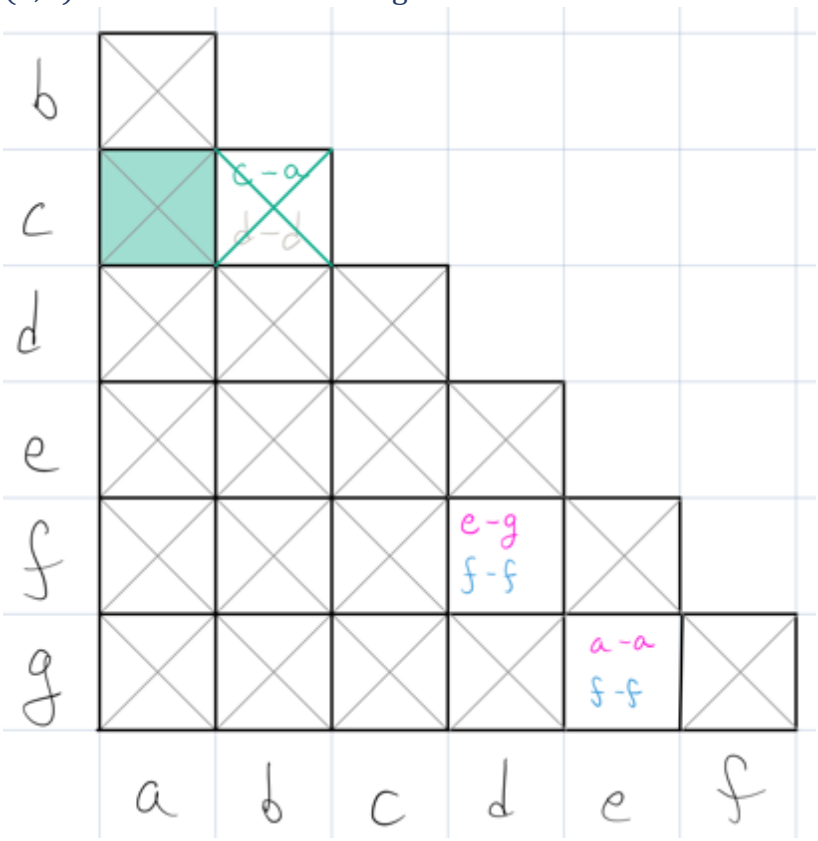




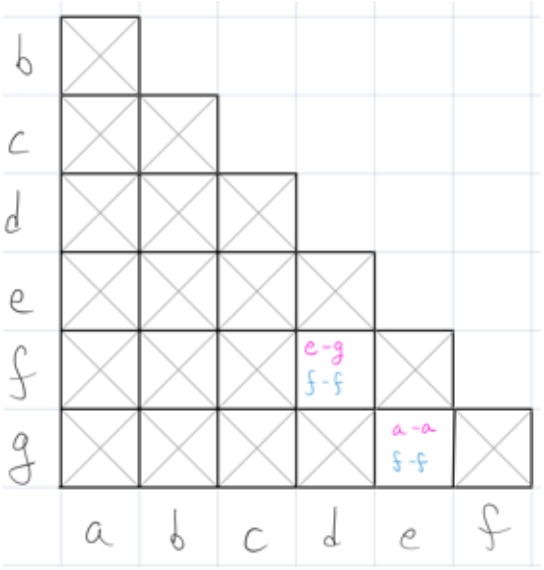
Notice that square at coordinate (a, c) used to be an open square but has since been crossed out due to the unequal next-state it contained, $b - d$.



Therefore, since the square at coordinate (b, c) contains the equal next state value $c - a$ (which is equivalent to $a - c$), and the square at coordinate (a, c) is now closed, then the square at coordinate (b, c) will need a cross through it.



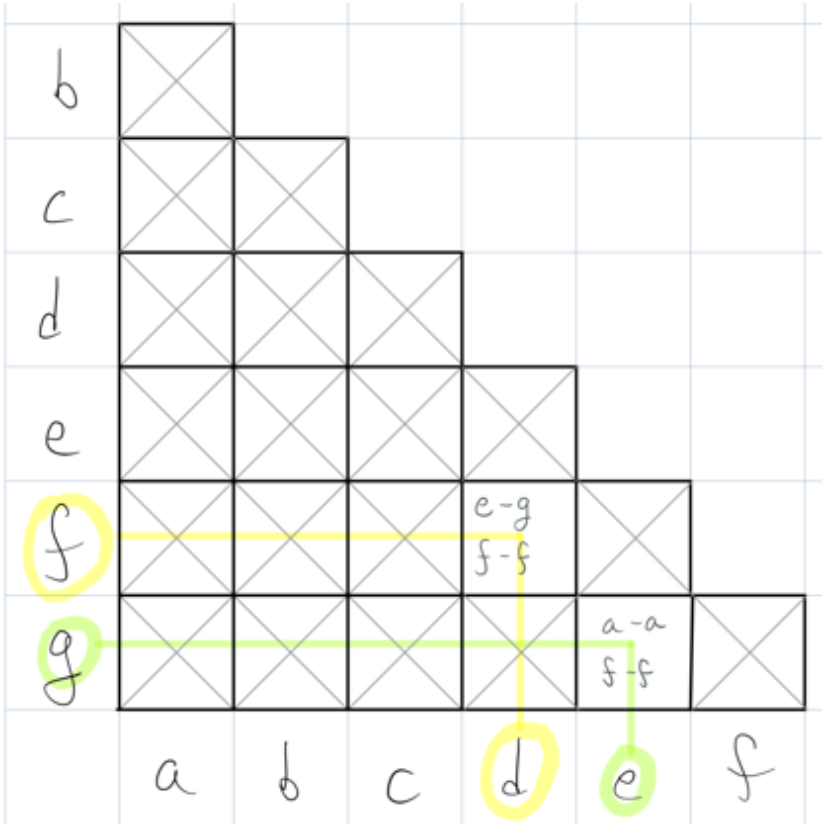
We repeat the same process taking into account the new crosses and stop when no new cross has been derived:



Once the final implication chart is derived, conclude the equivalent states based on the coordinates of the open squares.

The coordinates of the remaining open squares on the implication chart are:

(d, f) and (e, g).



We can conclude that:

State f ie equivalent to state d.

State e is equivalent to state g.

c. Derive the final reduced state table and diagram.

Derive the optimized state table for the sequential circuit by replacing all the states (present and next) that have equivalencies, with their equivalent values:

Equivalent states:

$f \equiv d$

$e \equiv g$

Original State Table:

Q	Q^+		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State Table with equivalent state labels:

Row #	Q	Q^+		Output	
		$x = 0$	$x = 1$	$x = 0$	$x = 1$
0	a	a	b	0	0
1	b	c	$d \equiv f$	0	0
2	c	a	$d \equiv f$	0	0
3	$d \equiv f$	$e \equiv g$	$d \equiv f$	0	1
4	$e \equiv g$	a	$d \equiv f$	0	1
5	$d \equiv f$	$e \equiv g$	$d \equiv f$	0	1
6	$e \equiv g$	a	$d \equiv f$	0	1

For all of the respective equivalent rows, relabel the equivalent states with the original present state designation of the first equivalent instance: [I know this is wordy… see below for an example]

Rows 3 and 5 have equivalent next states since they have the same next states and same outputs..

Since row 3 is the first instance of equivalency, and the original present-state designation for that row was d, we change $d \equiv f$ back to d.

We then relabel all the remaining $d \equiv f$ in the state table to d.

Rows 4 and 6 have equivalent next states.

Since row 4 is the first instance of equivalency, and the original present-state designation for that row was e, we change $e \equiv g$ back to e.

We then relabel all the remaining $e \equiv g$ in the state table to e.

Row #	Q	Q^+		Output	
		$x = 0$	$x = 1$	$x = 0$	$x = 1$
0	a	a	b	0	0
1	b	c	$d \equiv f$	0	0
2	c	a	$d \equiv f$	0	0
3	d	$e \equiv g$	$d \equiv f$	0	1
4	e	a	$d \equiv f$	0	1
5	d	$e \equiv g$	$d \equiv f$	0	1
6	e	a	$d \equiv f$	0	1

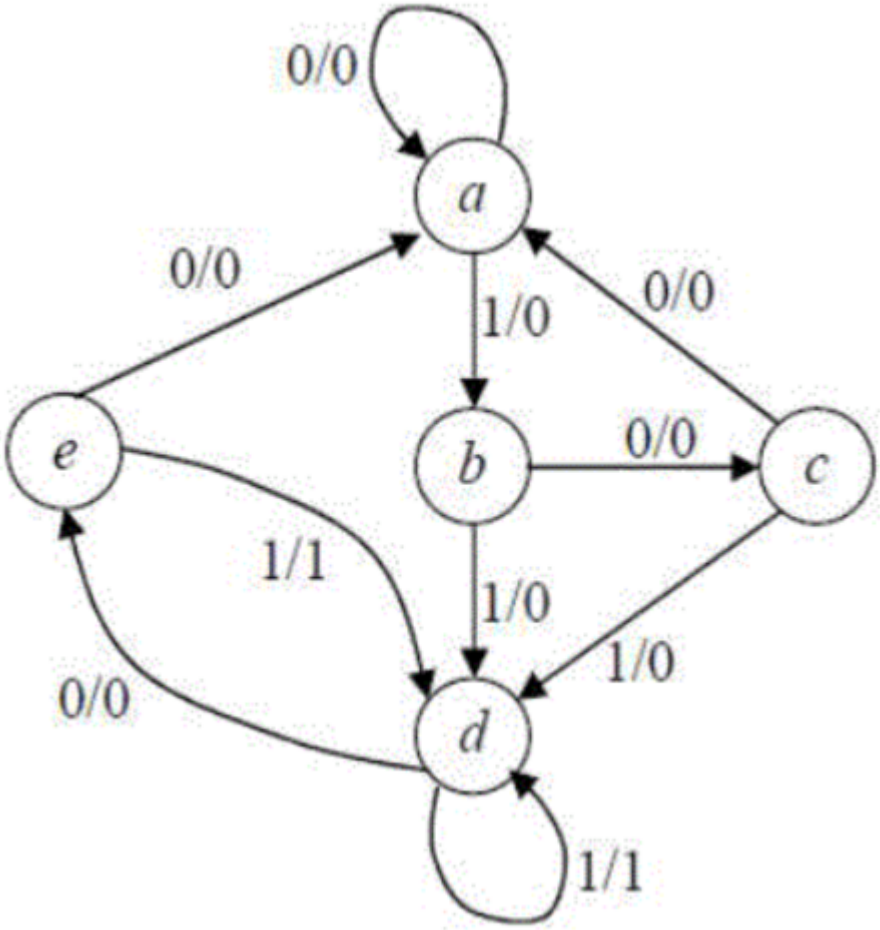
Row #	Q	Q^+		Output	
		$x = 0$	$x = 1$	$x = 0$	$x = 1$
0	a	a	b	0	0
1	b	c	d	0	0
2	c	a	d	0	0
3	d	e	d	0	1
4	e	a	d	0	1
5	d	e	d	0	1
6	e	a	d	0	1

Remove all of the redundant equivalent rows from the state table, keeping only the first instance:

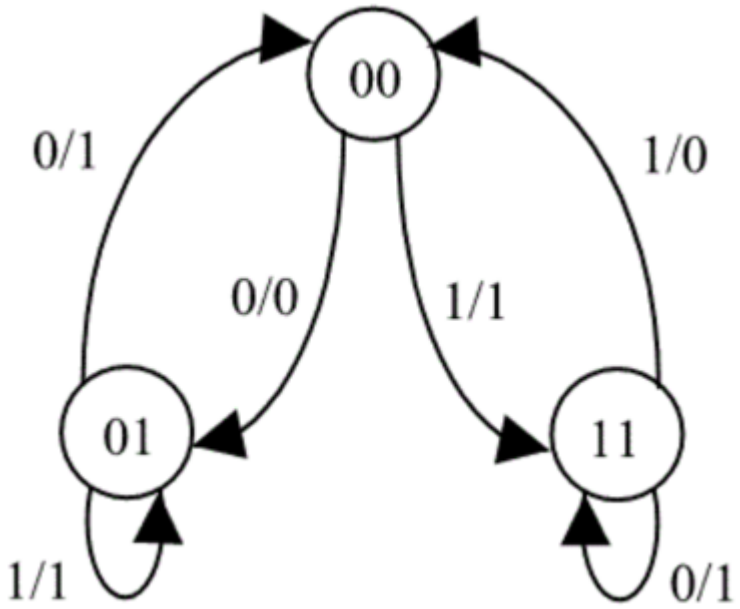
Row #	Q	Q^+		Output	
		$x = 0$	$x = 1$	$x = 0$	$x = 1$
0	a	a	b	0	0
1	b	c	d	0	0
2	c	a	d	0	0
3	d	e	d	0	1
4	e	a	d	0	1

Draw the state diagram based on the finalized reduced state table:

Row #	Q	Q^+		Output	
		$x = 0$	$x = 1$	$x = 0$	$x = 1$
0	a	a	b	0	0
1	b	c	d	0	0
2	c	a	d	0	0
3	d	e	d	0	1
4	e	a	d	0	1



2. Given the following state diagram (input: x, output: z, states: AB):



a. What are the states and output equations of the circuit?

Derive the state table based on the state diagram. Don't forget the unused states.

Present State		Inputs	Next States		Output
A	B	x	A^+	B^+	y
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	1	0	1	1
1	0	0	X	X	X
1	0	1	X	X	X
1	1	0	1	1	1
1	1	1	0	0	0

Derive the next-state and output equations based on the state table.

$$A^+ = \sum m(1,6)\ ; A^{+\prime} = \prod M(0,2,3,7)$$

A/Bx	00	01	11	10
0	0 (0)	1 (1)	0 (3)	0 (2)
1	X (4)	X (5)	0 (7)	1 (6)

$$A^+ = B'x + Ax'$$

$$B^+ = \sum m(0,1,3,6)\ ; B^{+\prime} = \prod M(2,7)$$

A/Bx	00	01	11	10
0	1 (0)	1 (1)	1 (3)	0 (2)
1	X (4)	X (5)	0 (7)	1 (6)

$$B^+ = B' + A'x + Ax' = B' + A \oplus x$$

$$y = \sum m(1,2,3,6)\ ; y' = \prod M(0,7)$$

A/Bx	00	01	11	10
0	0 (0)	1 (1)	1 (3)	1 (2)
1	X (4)	X (5)	0 (7)	1 (6)

$$y = A'x + Bx'$$

Fill in the Next-State and output values for the unused states based on the derived equations:

Summary of Equations:

$A^+ = B'x + Ax'$

$B^+ = B' + A \oplus x$

$y = A'x + Bx'$

Note: These equations are different from the posted solutions on BrightSpace because the posted solution for B^+ is not optimized using the "don't care" values, (since it was derived from the state table). Therefore, the posted solution for the remainder of this will be different from our answer. The answer on BrightSpace is incorrect.

Present State		Inputs	Next States		Output
A	B	x	A^+	B^+	y
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	0	0	0

b. **Implement the circuit using JK flip-flops. Derive the optimized SOP and the flip flop's inputs (J_A, K_A, J_B, K_B) and the output z.**

Declare the excitation table for a JK Flip-Flop:

JK Flip-Flop Excitation Table			
Q	Q^+	J_Q	K_Q
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0
$Q^+ = JQ' + K'Q$			

Fill in the values of the JK flip-flops based on the Present- and Next-States, and the JK Flip-Flop Excitation Table.

	Present State		Inputs	Next States		Output	JK Flip-Flop Inputs			
	A	B	x	A^+	B^+	y	J_A	K_A	J_B	K_B
Used States	0	0	0	0	1	0	0	X	1	X
	0	0	1	1	1	1	1	X	1	X
	0	1	0	0	0	1	0	X	X	1
	0	1	1	0	1	1	0	X	X	0
	1	1	0	1	1	1	X	0	X	0
	1	1	1	0	0	0	X	1	X	1
Unused States	1	0	0	1	1	0	X	0	1	X
	1	0	1	1	1	1	X	0	1	X

Derive the JK Input equations using the SOP terms for each input for KMap:

$$J_A = \sum m(1) \; ; J_A' = \prod M(0,2,3)$$
$$K_A = \sum m(7) \; ; K_A{}' = \prod M(4,5,6)$$

$J_B = 1$ [directly from state table]

$$K_B = \sum m(2,7) \; ; K_B{}' = \prod M(3,6)$$

$$J_A = \sum m(1)\ ; J_A' = \prod M(0,2,3)$$

A/Bx	00	01	11	10
0	0 (0)	1 (1)	0 (3)	0 (2)
1	X (4)	X (5)	X (7)	X (6)

$$J_A = B'x$$

$$K_A = \sum m(7)\ ; K_A' = \prod M(4,5,6)$$

A/Bx	00	01	11	10
0	X (0)	X (1)	X (3)	X (2)
1	0 (4)	0 (5)	1 (7)	0 (6)

$$K_A = Bx$$

$J_B = \sum m(0,1,4,5)$;

A/Bx	00	01	11	10
0	1 (0)	1 (1)	X (3)	X (2)
1	1 (4)	1 (5)	X (7)	X (6)

$J_B = 1$

$K_B = \sum m(2,7)$; $K_B{}' = \prod M(3,6)$

A/Bx	00	01	11	10
0	X (0)	X (1)	0 (3)	1 (2)
1	X (4)	X (5)	1 (7)	0 (6)

$K_B = A'x' + Ax = A \odot x$

Summary of Equations:

$A^+ = B'x + Ax'$
$B^+ = B' + A \oplus x$
$y = A'x + Bx'$

$J_A = B'x$
$K_A = Bx$
$J_B = 1$
$K_B = A'x' + Ax = A \odot x$

Check the Equations [if you have time]:

Summary of Equations:

$A^+ = B'x + Ax'$
$B^+ = B' + A \oplus x$
$y = A'x + Bx'$

$J_A = B'x$
$K_A = Bx$
$J_B = 1$
$K_B = A'x' + Ax = A \odot x$

$\boldsymbol{Q^+ = JQ' + K'Q}$
$A^+ = J_A A' + K_A' A$
$A^+ = (B'x)A' + (Bx)'A$
$A^+ = A'B'x + A(B' + x')$
$A^+ = A'B'x + AB' + Ax'$
$A^+ = B'(A'x + A) + Ax'$
$A^+ = B'((A'x)'A')' + Ax'$
$A^+ = B'\big((A + x')A'\big)' + Ax'$
$A^+ = B'(AA' + A'x')' + Ax'$
$A^+ = B'(0 + A'x')' + Ax'$
$A^+ = B'(A'x')' + Ax'$
$A^+ = B'(A + x) + Ax'$
$A^+ = AB' + B'x + Ax'$???

$\boldsymbol{Q^+ = JQ' + K'Q}$
$B^+ = J_B B' + K_B' B$
$B^+ = (1)B' + (A \odot x)'B$
$B^+ = B' + (A \oplus x)B$
$B^+ = B' + (A \oplus x)B$
$B^+ = \Big(B\big((A \odot x) + B'\big)\Big)'$
$B^+ = \Big(B\big((A \odot x) + B'\big)\Big)'$
$B^+ = \big((B(A \odot x) + BB')\big)'$
$B^+ = \Big(\big(B(A \odot x) + (0)\big)\Big)'$
$B^+ = \big(B(A \odot x)\big)'$
$B^+ = B' + A \oplus x$

Draw the logic circuit based on the equations:

Summary of Equations:

$A^+ = B'x + Ax'$

$B^+ = B' + A \oplus x$

$y = A'x + Bx'$

$J_A = B'x$

$K_A = Bx$

$J_B = 1$

$K_B = A'x' + Ax = A \odot x$

3. A synchronous sequential circuit has 3 SR flip flops whose inputs are:
   $S_A = A'BC$
   $R_A = ABC$

   $S_B = B'C$
   $R_B = BC$

   $S_C = C'$
   $R_C = C$

   a. Derive the state transition table of the system.

Fill in the SR Flip-Flop values based on the SR input equations and the present states:

Present State			Next State			SR Flip-Flop Inputs					
A	B	C	A^+	B^+	C^+	S_A	R_A	S_B	R_B	S_C	R_C
0	0	0				0	0	0	0	1	0
0	0	1				0	0	1	0	0	1
0	1	0				0	0	0	0	1	0
0	1	1				1	0	0	1	0	1
1	0	0				0	0	0	0	1	0
1	0	1				0	0	1	0	0	1
1	1	0				0	0	0	0	1	0
1	1	1				0	1	0	1	0	1

Declare the characteristic table for an SR Flip-Flop:

SR Flip-Flop Characteristic Table			
S_Q	R_Q	Q^+	Description
0	0	Q	No Change
0	1	0	Reset
1	0	1	Set
1	1	X	Complement

Use the SR Flip-Flop characteristic table to derive the values for the next states in the transition table:

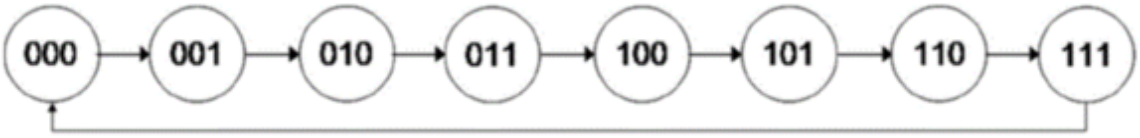
Present State			Next State			SR Flip-Flop Inputs					
A	B	C	A^+	B^+	C^+	S_A	R_A	S_B	R_B	S_C	R_C
0	0	0	0	0	1	0	0	0	0	1	0
0	0	1	0	1	0	0	0	1	0	0	1
0	1	0	0	1	1	0	0	0	0	1	0
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	1	0	1	0	0	0	0	1	0
1	0	1	1	1	0	0	0	1	0	0	1
1	1	0	1	1	1	0	0	0	0	1	0
1	1	1	0	0	0	0	1	0	1	0	1

b. **Derive the state diagram of the system.**

[Optional] You can put the transition table in state order to make it easier to draw the diagram:

The table is already in sequential order:

Present State			Next State			SR Flip-Flop Inputs					
A	B	C	A^+	B^+	C^+	S_A	R_A	S_B	R_B	S_C	R_C
0	0	0	0	0	1	0	0	0	0	1	0
0	0	1	0	1	0	0	0	1	0	0	1
0	1	0	0	1	1	0	0	0	0	1	0
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	1	0	1	0	0	0	0	1	0
1	0	1	1	1	0	0	0	1	0	0	1
1	1	0	1	1	1	0	0	0	0	1	0
1	1	1	0	0	0	0	1	0	1	0	1



c. **Derive the state equations of the system.**

Since the transition table is completely filled out, you can find the state equations by deriving them from the minterms of each "Next-State". We will do it a different way below though by using the excitation equation for the flip-flop being used.

[see next page]

Declare the excitation table for a SR Flip-Flop:

SR Flip-Flop Excitation Table			
Q	Q^+	S_Q	R_Q
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0
$Q^+ = S + R'Q$			

Establish the Next-State equations by substituting the values for the respective S R inputs with the S-input equation, and R-input equation:

Summary of SR Input Equations:

$S_A = A'BC$
$R_A = ABC$

$S_B = B'C$
$R_B = BC$

$S_C = C'$
$R_C = C$

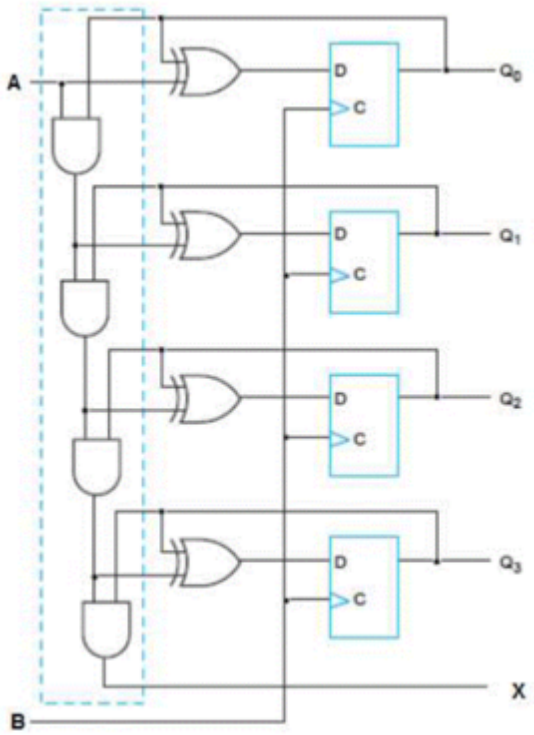
$Q^+ = S + R'Q$
$A^+ = S_A + R'_A A$
$A^+ = (A'BC) + (ABC)'A$
$A^+ = A'BC + A(A' + B' + C')$
$A^+ = A'BC + AA' + AB' + AC'$
$A^+ = A'BC + (0) + AB' + AC'$
$A^+ = A'BC + A(B' + C')$
$A^+ = A'BC + A(BC)'$
$A^+ = A \oplus (BC)$

$B^+ = S_B + R'_B B$
$B^+ = (B'C) + (BC)'B$
$B^+ = B'C + B(B' + C')$
$B^+ = B'C + BB' + BC'$
$B^+ = B'C + (0) + BC'$
$B^+ = B'C + BC'$
$B^+ = B \oplus C$

$C^+ = S_C + R_C{}'C$
$C^+ = (C') + (C)'C$
$C^+ = C' + CC'$
$C^+ = C' + (0)$
$C^+ = C'$

1. **Analyze the following circuit and describe how the Q and x functions behave in terms of $A, B, and\ D$. (Derive the short descriptive table).**

[NOTE: As mentioned in lecture, similar question likely to be asked about registers, such as serial transfer, serial addition, parallel loading registers, parallel-access shift registers, etc].



First derive the equations based on the circuit diagram:

A = enable input (since shared among all D_i inputs).
B = PET clk (synchronous positive edge trigger)
$x = AQ_0Q_1Q_2Q_3$
$D_0 = A \oplus Q_0$
$D_1 = (AQ_0) \oplus Q_1$
$D_2 = (AQ_0Q_1) \oplus Q_2$
$D_3 = (AQ_0Q_1Q_2) \oplus Q_3$
$Q_i^+ = D_i$

(PET) CLK	Input	D Flip-Flop Inputs	Next States		Description
B	A	D_i	Q_i^+	x^+	
X	X	X	Q	$AQ_0Q_1Q_2Q_3$	No Change for Q
↑	0	Q	Q	0	No Change for Q
↑	1	X	(Q+1) mod-16	Carry (1 if $Q_i = 15$)	Count

Declare the Excitation table for a D Flip-Flop to help explain the behavior of the circuit:

D Flip-Flop Excitation Table		
Q	Q^+	D_Q
0	0	0
0	1	1
1	0	0
1	1	1
$Q^+ = D$		
$D = Q^+$		
$Q = Q^+D + Q^{+'}D' = Q^+ \odot D$		

Identify the type of logic used in the circuit, positive or negative. This trait can be identified with an active-low or active-high input, or a negative- or positive-edge trigger (NET or PET) clock.

B is a PET clock, so the values of Q_i are only affected when B goes from 0 to 1, regardless the value of the input A.

If B changes from 0 to 1 (↑), then $Q^+ = D$, otherwise $Q^+ = Q$.

Describe the behavior of the inputs in relation to the clock:

A is common among all of the D_i inputs, therefore, they are only "enabled" if $A = 1$ and the clock B changes from 0 to 1 (↑).

If B changes from 0 to 1 (↑) and $A = 0$, then the enable is off so $D = Q$, (the state remains unchanged). In this case, the output of $X = 1$.

If B changes from 0 to 1 (↑) and $A = 1$ (active enable), then Q_0 alternates, Q_1 if Q_0 changes from 1 to 0, Q_2 alternates if Q_1 changes from 1 to 0, and Q_3 alternates if Q_2 changes from 1 to 0. Thus, $Q^+ = Q + 1$ modulo 16.
In this case, the output $X = 1$ only when $Q = 15$ (carry at the next clock transition).

This is thus a 4-bit synchronous binary counter, where A is the input *count* enable, B is the clock, Q is the 4-bit output, and X is the carry.