Boolean Algebra

Dr. Shilpi Gupta
Associate Professor
ECED
SVNIT, Surat

The "WHY" Boolean Algebra

- Algebra

When we learned numbers like 1, 2, 3, we also then learned how to add, multiply, etc. with them. Boolean Algebra covers operations that we can do with 0's and 1's. Computers do these operations ALL THE TIME and they are basic building blocks of computation inside your computer program.

Axioms, laws, theorems

We need to know some rules about how those 0's and 1's can be operated on together. There are similar axioms to decimal number algebra, and there are some laws and theorems that are good for you to use to simplify your operation.

How does Boolean Algebra fit into the big picture?

- It is part of the Combinational Logic topics (memoryless)
  - Different from the Sequential logic topics (can store information)

- Learning Axioms and theorems of Boolean algebra
  - Allows you to design logic functions
  - Allows you to know how to combine different logic gates
  - Allows you to simplify or optimize on the complex operations

Boolean algebra

- A Boolean algebra comprises...
  - A set of elements B
  - Binary operators {+ , •} Boolean sum and product
  - A unary operation { ' } (or { $\bar{\ }$ }) example: A’ or $\bar{A}$
- ...and the following axioms
  - 1. The set B contains at least two elements {a b} with a $\neq$ b
  - 2. Closure: a+b is in B a•b is in B
  - 3. Commutative: a+b = b+a a•b = b•a
  - 4. Associative: a+(b+c) = (a+b)+c a•(b•c) = (a•b)•c
  - 5. Identity: a+0 = a a•1 = a
  - 6. Distributive: a+(b•c)=(a+b)•(a+c) a•(b+c)=(a•b)+(a•c)
  - 7. Complementarity: a+a' = 1 a•a' = 0

Digital (binary) logic is a Boolean algebra

Substitute

- {0, 1} for B
- AND for • Boolean Product.
- OR for + Boolean Sum.
- NOT for ‘ Complement.

All the axioms hold for binary logic

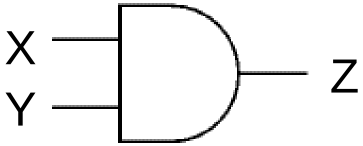
Definitions

- Boolean function
  - Maps inputs from the set {0,1} to the set {0,1}
- Boolean expression
  - An algebraic statement of Boolean variables and operators

‘

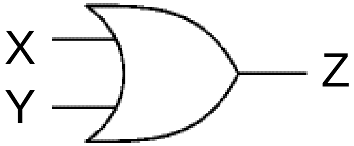
Logic Gates (AND, OR, Not) & Truth Table

AND $X \bullet Y$ XY



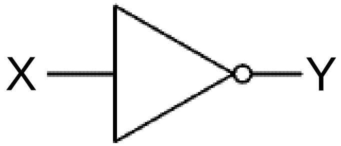
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR $X+Y$



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

NOT $\overline{X}$ X'



X	Y
0	1
1	0

Logic functions and Boolean algebra

- *Any* logic function that is expressible as a truth table can be written in Boolean algebra using +, •, and '

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Z=X•Y

X	Y	X'	Z
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

Z=X'•Y

X	Y	X'	Y'	X • Y	X' •Y'	Z
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

Z=(X•Y)+(X' •Y')

Two key concepts

- **Duality (a meta-theorem—** *a theorem about theorems*)
  - All Boolean expressions have logical duals
  - Any theorem that can be proved is also proved for its dual
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Leave the variables unchanged

- **De Morgan's Theorem**
  - Procedure for complementing Boolean functions
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Replace all variables with their complements

Universal Gate Implementation

NAND Gate as Universal Gate

OR Gate using NAND Gate

NOR Gate using NAND Gate

EXOR using NAND Gate

NOR Gate as Universal Gate: TO DO

EX-OR Gate Using NOR gate

Boolean Functions

- Boolean function is described by an a algebraic expression called Boolean expressions which consists of binary variables and logic operation symbols.
- Mathematical functions can be expressed in two ways:

An expression is finite but not unique

$$
\begin{aligned}
f(x,y) &= 2x + y \\
&= x + x + y \\
&= 2(x + y/2) \\
&= \ldots
\end{aligned}
$$

A function table is unique

x	y	f(x,y)
0	0	0
…	…	…
2	2	6
…	…	…
23	41	87
…	…	…

We can represent logical functions in two analogous ways too:

- A finite, but non-unique Boolean expression.
- A truth table, which will turn out to be unique.

Boolean expressions

- We can use these basic operations to form more complex expressions:

$$f(x,y,z) = (x + y')z + x'$$

  - f is the name of the function.
  - (x, y, z) are the input variables, each representing 1 or 0. Listing the inputs is optional, but sometimes helpful.
  - A literal is any occurrence of an input variable or its complement. The function above has four literals: x, y', z, and x'.
- Precedence is important, but not too difficult.
  - NOT has the highest precedence, followed by AND, and then OR.
  - Fully parenthesized, the function above would be kind of messy:

$$f(x,y,z) = (((x +(y'))z) + x')$$

Useful laws and theorems

Identity: $X + 0 = X$ Dual: $X \bullet 1 = X$

Null: $X + 1 = 1$ Dual: $X \bullet 0 = 0$

Idempotent: $X + X = X$ Dual: $X \bullet X = X$

Involution: $(X')' = X$

Complementarity: $X + X' = 1$ Dual: $X \bullet X' = 0$

Commutative: $X + Y = Y + X$ Dual: $X \bullet Y = Y \bullet X$

Associative: $(X+Y)+Z=X+(Y+Z)$ Dual: $(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z)$

Distributive: $X \bullet (Y+Z)=(X \bullet Y)+(X \bullet Z)$ Dual: $X+(Y \bullet Z)=(X+Y) \bullet (X+Z)$

Uniting: $X \bullet Y+X \bullet Y'=X$ Dual: $(X+Y) \bullet (X+Y')=X$

Proof of law with Truth table

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

$$(A + B) + C = A + (B + C)$$

Useful laws and theorems (con't)

Absorption: X+X•Y=X Dual: X•(X+Y)=X

Absorption (#2): (X+Y')•Y=X•Y Dual: (X•Y')+Y=X+Y

$$x + \bar{x}y = x + y$$

$$(x + y)(x + z) = x + yz$$

de Morgan's: (X+Y+...)'=X'•Y'•... Dual: (X•Y•...)'=X'+Y'+...

Duality: $(X+Y+...)^D$=X•Y•... Dual: $(X•Y•...)^D$=X+Y+...

Multiplying & factoring: (X+Y)•(X'+Z)=X•Z+X'•Y

Dual: X•Y+X'•Z=(X+Z)•(X'+Y)

Consensus: (X•Y)+(Y•Z)+(X'•Z)= X•Y+X'•Z

Dual: (X+Y)•(Y+Z)•(X'+Z)=(X+Y)•(X'+Z)

Proofing the theorems using axioms

- Idempotency: $x + x = x$

  Proof:

$$
\begin{aligned}
x + x &= (x + x) \bullet 1 && \text{by identity} \\
&= (x + x) \bullet (x + x') && \text{by complement} \\
&= x + x \bullet x' && \text{by distributivity} \\
&= x + 0 && \text{by complement} \\
&= x && \text{by identity}
\end{aligned}
$$

- Idempotency: $x \bullet x = x$

  Proof:

$$
\begin{aligned}
x \bullet x &= (x \bullet x) + 0 && \text{by identity} \\
&= (x \bullet x) + (x \bullet x') && \text{by complement} \\
&= x \bullet (x + x') && \text{by distributivity} \\
&= x \bullet 1 && \text{by complement} \\
&= x && \text{by identity}
\end{aligned}
$$

Theorems

Prove the uniting theorem-- $X \bullet Y + X \bullet Y' = X$

Distributive	$X \bullet Y + X \bullet Y' = X \bullet (Y + Y')$
Complementarity	$= X \bullet (1)$
Identity	$= X$

Prove the absorption theorem-- $X + X \bullet Y = X$

Identity	$X + X \bullet Y = (X \bullet 1) + (X \bullet Y)$
Distributive	$= X \bullet (1 + Y)$
Null	$= X \bullet (1)$
Identity	$= X$

To Prove

$$A + \bar{A}B = A + B$$

$$(\boldsymbol{A} + \boldsymbol{B})(\boldsymbol{A} + \boldsymbol{C}) = \boldsymbol{A} + \boldsymbol{BC}$$

More proves

$$(A+\bar{B}+AB)(A+\bar{B})(\bar{A}B)=0$$

$$A+\bar{A}B+AB=A+B$$

Simplify the following expression

$$Y=(\overline{\overline{AB}+\bar{A}+AB})$$

Theorems

Prove the consensus theorem--

$$(XY)+(YZ)+(X'Z)= XY+X'Z$$

Complementarity $XY+YZ+X'Z = XY+(X+X')YZ + X'Z$

Distributive $= XYZ+XY+X'YZ+X'Z$

- *Use absorption {AB+A=A} with A=XY and B=Z*

$= XY+X'YZ+X'Z$

Rearrange terms $= XY+X'ZY+X'Z$

- *Use absorption {AB+A=A} with A=X'Z and B=Y*

$$XY+YZ+X'Z = XY+X'Z$$

Logic simplification

Example:

$$Z = A'BC + AB'C' + AB'C + ABC' + ABC$$

$= A'BC + AB'(C' + C) + AB(C' + C)$ distributive

$= A'BC + AB' + AB$ complementary

$= A'BC + A(B' + B)$ distributive

$= A'BC \quad + A$ complementary

$= BC + A$ absorption #2 Duality

$(X \bullet Y')+Y=X+Y$ with $X=BC$ and $Y=A$

Algebraic manipulation

$x'y' + xyz + x'y$

$= x'(y' + y) + xyz$ [Distributive; $x'y' + x'y = x'(y' + y)$]

$= x' \bullet 1 + xyz$ [Axiom 5; $y' + y = 1$]

$= x' + xyz$ [Axiom 2; $x' \bullet 1 = x'$]

$= (x' + x)(x' + yz)$ [Distributive]

$= 1 \bullet (x' + yz)$ [Axiom 5; $x' + x = 1$]

$= x' + yz$ [Axiom 2 ; $x' \bullet 1 = x'$]

More Problems:

$$A\bar{B} + \bar{A}B + \bar{A}\bar{B} + AB$$

$$(AB + C)(AB + D)$$

$$AB + ABC + A\bar{B} = A \quad \text{(Prove)}$$

De Morgan's Theorem

Use de Morgan's Theorem to find complements

Example: $F=(A+B)\bullet(A'+C)$, so $F'=(A'\bullet B')+(A\bullet C')$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A	B	C	F'
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Complement of a function

- The complement of a function always outputs 0 where the original function outputted 1, and 1 where the original produced 0.
- In a truth table, we can just exchange 0s and 1s in the output column(s)

$$f(x,y,z) = x(y'z' + yz)$$

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

x	y	z	f'(x,y,z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Complementing a function algebraically

$$f(x,y,z) = x(y'z' + yz)$$

$$\begin{aligned} f'(x,y,z) &= (\ x(y'z' + yz)\ )' && [\text{ complement both sides }] \\ &= x' + (y'z' + yz)' && [\text{ because } (xy)' = x' + y' \text{ }] \\ &= x' + (y'z')'\,(yz)' && [\text{ because } (x + y)' = x'\,y' \text{ }] \\ &= x' + (y + z)(y' + z') && [\text{ because } (xy)' = x' + y',\text{ twice}] \end{aligned}$$

- You can use DeMorgan's law to keep "pushing" the complements inwards
- You can also take the dual of the function, and then complement each literal
  - If $f(x,y,z) = x(y'z' + yz)$…
  - …the dual of f is $x + (y' + z')(y + z)$…
  - …then complementing each literal gives $x' + (y + z)(y' + z')$…
  - …so $f'(x,y,z) = x' + (y + z)(y' + z')$

Canonical Forms

- Any boolean function that is expressed as a sum of minterms or as a product of maxterms is said to be in its canonical form.
- A minterm is a special product of literals, in which each input variable appears exactly once.
- A function with n variables has 2^n minterms (since each variable can appear complemented or not) A three-variable function, such as f(x,y,z), has $2^3 = 8$ minterms:

$$x'y'z' \qquad x'y'z \qquad x'yz' \qquad x'yz$$
$$xy'z' \qquad xy'z \qquad xyz' \qquad xyz$$

Minterms

- Each minterm is true for exactly one combination of inputs:

Minterm	Is true when…	Shorthand
$x'y'z'$	$x=0,\ y=0,\ z=0$	m_0
$x'y'z$	$x=0,\ y=0,\ z=1$	m_1
$x'yz'$	$x=0,\ y=1,\ z=0$	m_2
$x'yz$	$x=0,\ y=1,\ z=1$	m_3
$xy'z'$	$x=1,\ y=0,\ z=0$	m_4
$xy'z$	$x=1,\ y=0,\ z=1$	m_5
xyz'	$x=1,\ y=1,\ z=0$	m_6
xyz	$x=1,\ y=1,\ z=1$	m_7

Sum of minterms form

- Every function can be written as a sum of minterms, which is a special kind of sum of products form
- The sum of minterms form for any function is *unique*
- If you have a truth table for a function, you can write a sum of minterms expression just by picking out the rows of the table where the function output is 1 (1-minterm).

x	y	z	f(x,y,z)	f'(x,y,z)
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

$$
\begin{aligned}
f &= x'y'z' + x'y'z + x'yz' + x'yz + xyz' \\
&= m_0 + m_1 + m_2 + m_3 + m_6 \\
&= \Sigma(0,1,2,3,6)
\end{aligned}
$$

$$
\begin{aligned}
f' &= xy'z' + xy'z + xyz \\
&= m_4 + m_5 + m_7 \\
&= \Sigma(4,5,7)
\end{aligned}
$$

f' contains all the minterms not in f

Sum of minterms: practise

F = x + yz, how to express this in the sum of minterms?

$$= x(y + y')(z + z') + (x + x')yz$$
$$= xyz + xyz' + xy'z + xy'z' + xyz + x'yz$$
$$= x'yz + xy'z' + xy'z + xyz' + xyz$$
$$= \Sigma(3,4,5,6,7)$$

or, convert the expression into truth-table and then read the minterms from the table

Maxterms

- A maxterm is a *sum* of literals, in which each input variable appears exactly once.
- A function with n variables has 2^n maxterms

$$
\begin{array}{llll}
x' + y' + z' & x' + y' + z & x' + y + z' & x' + y + z \\
x + y' + z' & x + y' + z & x + y + z' & x + y + z
\end{array}
$$

xyz

000

x'y'z'

~~x+y+z~~

POS

x+y+z

- The maxterms for a three-variable function f(x,y,z):

Maxterm	Is *false* when…	Shorthand
$x + y + z$	x=0, y=0, z=0	M_0
$x + y + z'$	x=0, y=0, z=1	M_1
$x + y' + z$	x=0, y=1, z=0	M_2
$x + y' + z'$	x=0, y=1, z=1	M_3
$x' + y + z$	x=1, y=0, z=0	M_4
$x' + y + z'$	x=1, y=0, z=1	M_5
$x' + y' + z$	x=1, y=1, z=0	M_6
$x' + y' + z'$	x=1, y=1, z=1	M_7

M_0

m_0

minterm

Max Term

111

111 xyz

x'+y'+z'

- Each maxterm is *false* for exactly one combination of inputs:

Product of maxterms form

- Every function can be written as a *unique* product of maxterms
- If you have a truth table for a function, you can write a product of maxterms expression by picking out the rows of the table where the function output is 0 (0-maxterm).

SOP min

O/P

x	y	z	f(x,y,z)	f'(x,y,z)
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

$x' + y + z$

M_4

M_5

M_7

POS

$$f = (x' + y + z)(x' + y + z')(x' + y' + z') = M_4\ M_5\ M_7 = \Pi(4,5,7)$$

Max

O/P→0

$\Pi M(4,5,7)$

SOP

$$f' = (x + y + z)(x + y + z')(x + y' + z)(x + y' + z')(x' + y' + z) = M_0\ M_1\ M_2\ M_3\ M_6 = \Pi(0,1,2,3,6)$$

O/P→1

SOP

$\Sigma m(0,1,2)$

f' contains all the maxterms not in f

Product of maxterms: practise

POS

Std. POS

Canonical

SOP

- $F = x'y' + xz$, how to express this in the product of maxterms?

$$= (x'y' + x)(x'y' + z)$$
$$= (x' + x)(y' + x)(x' + z)(y' + z)$$
$$= (x + y')(x' + z)(y' + z)$$
$$= (x + y' + zz')(x' + z + yy')(xx' + y' + z)$$
$$= (x + y' + z)(x + y' + z')(x' + y + z)(x' + y' + z)(x + y' + z)(x' + y' + z)$$
$$= (x + y' + z)(x + y' + z')(x' + y + z)(x' + y' + z)$$
$$= \prod(2,3,4,6)$$

Std.

$xY(Z+Z')$

$xYZ + xYZ'$

Std. POS

SOP

Std SOP

010   011   100   110

POS

$F(A,B,C) = (A+B)(A'+C)(B'+C')$

or, convert the expression into truth-table and then read the minterms from the table

Minterms and maxterms are related

- Any minterm m_i is the *complement* of the corresponding maxterm M_i

Minterm	Shorthand	Maxterm	Shorthand
$x'y'z'$	m_0	$x + y + z$	M_0
$x'y'z$	m_1	$x + y + z'$	M_1
$x'yz'$	m_2	$x + y' + z$	M_2
$x'yz$	m_3	$x + y' + z'$	M_3
$xy'z'$	m_4	$x' + y + z$	M_4
$xy'z$	m_5	$x' + y + z'$	M_5
xyz'	m_6	$x' + y' + z$	M_6
xyz	m_7	$x' + y' + z'$	M_7

$(xy'z')'$

$x' + y + z$

$(m_4)' = M_4$

- For example, $m_4' = M_4$ because $(xy'z')' = x' + y + z$

Converting between canonical forms

SOP f^n → POS f^n

POS

- We can convert a sum of minterms to a product of maxterms

$$
\begin{array}{lll}
\text{From before} & f & = \Sigma(0,1,2,3,6) \\
\text{and} & f' & = \Sigma(4,5,7) \\
 & & = m_4 + m_5 + m_7 \\
\text{complementing} & (f')' & = (m_4 + m_5 + m_7)' \\
\text{so} & f & = m_4'\, m_5'\, m_7' \qquad \text{[DeMorgan's law]} \\
 & & = M_4\, M_5\, M_7 \qquad \text{[By the previous page]} \\
 & & = \Pi(4,5,7)
\end{array}
$$

SOP

POS f^n

- In general, just replace the minterms with maxterms, using maxterm numbers that don't appear in the sum of minterms:

$$
\begin{aligned}
f &= \Sigma(0,1,2,3,6) \\
&= \Pi(4,5,7)
\end{aligned}
$$

- The same thing works for converting from a product of maxterms to a sum of minterms

Standard Forms

- Any boolean function that is expressed as a sum of products (SOP) or as a product of sums (POS), where each product-term or sum-term may require fewer than (n-1) operations, is said to be in its standard form.
- Standard forms are not unique, there can be several different SOPs and POSs for a given function.
- A SOP expression contains:
  - Only OR (sum) operations at the "outermost" level
  - Each term (implicant) must be a product of literals

$$f(x,y,z) = xy + x'yz + xy'z$$

- A POS expression contains:
  - Only AND (product) operations at the "outermost" level
  - Each term (implicate) must be a sum of literals

$$f(x,y,z) = (x' + y')(x + y' + z')(x' + y + z')$$

Minimization of Switching Function

$F = X'Y'Z + XY + X'Y'$

K'MAP

Algebric manipulation

Theorem, Laws

3 Variable

2V

	Y'	Y
X'	0 $x'y'$	1 $x'y$
X	2 xy'	3 xy

X \ YZ	00 $Y'Z'$	01 $Y'Z$	YZ	YZ'
X'	0: 0	1: 1	3: 0	2: 0
X	4: 0	5: 0	7: 1	6: 1

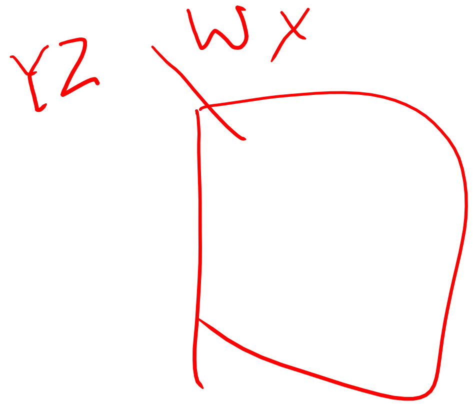
X	Y	Z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

4 V - K-MAP

WX \ YZ	Y'Z'	Y'Z	YZ	YZ'
W'X'	0 W'X'Y'Z'	1	3	2
W'X	4	5	7	6
WX	12	13	15	14
WX'	8	9	11	10

X Y Z

W X Y Z	F
0 W'X'Y'Z'	1
1 W'X'Y'Z	0
	1
16	1
	0
	0
	1
	1
	1
	1
	0
15	0



	A	B	C	Y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

$\sum m(0, 3, 5, 7)$

A \ BC	B'C'	B'C	BC	BC'
A'	0: 1	1: 1	3: 1	2: 1
A	4: 1	5: 1	7: 1	6: 1

= 1

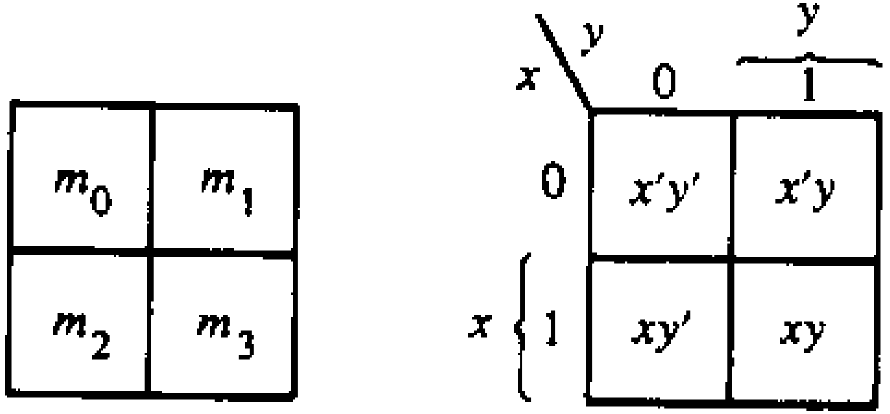
Minimize Group

$2^0, 2^1, 2^2, 2^3, 2^4$

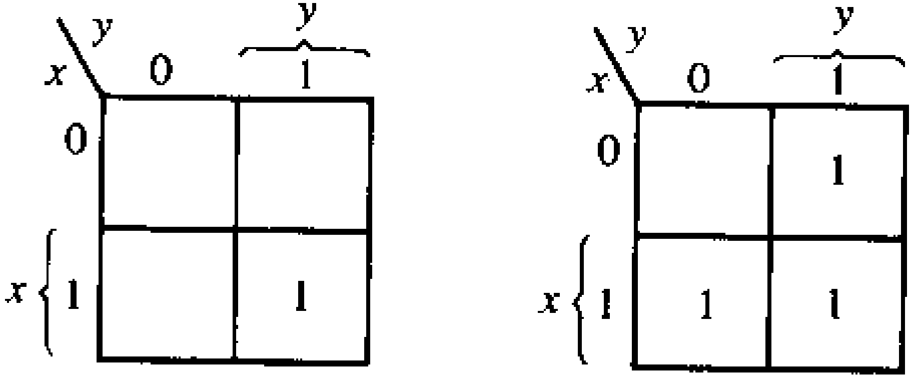
$\boxed{1, 2, 4, 8}, 16$

6

Two variable K Map



(a) (b)



(a) xy (b) $x + y$

3- variable K-Map

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

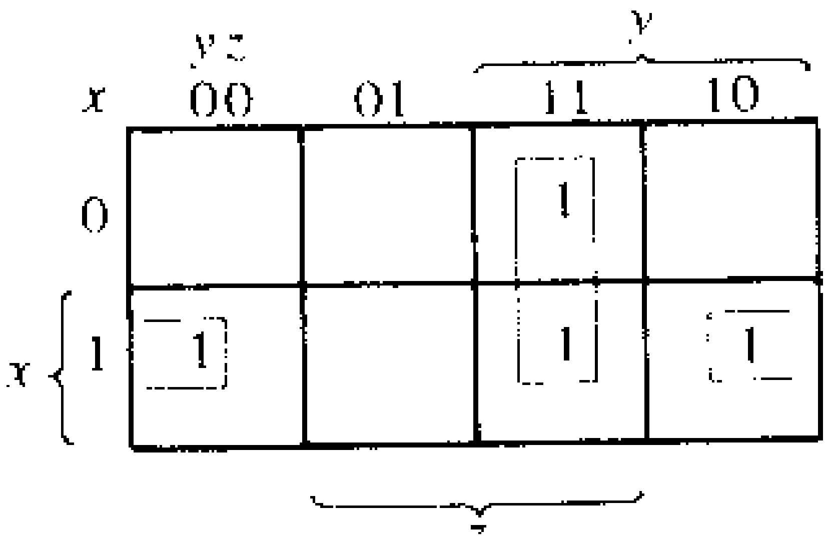
$x \backslash yz$	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

$$F(x, y, z) = \Sigma(2, 3, 4, 5)$$

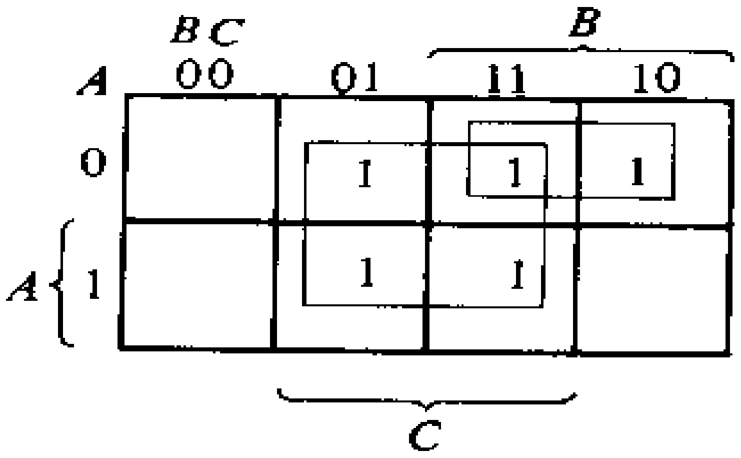
$x \backslash yz$	00	01	11	10
0			1	1
1	1	1		

Simplify the Boolean function

$$F(x, y, z) = \Sigma(3, 4, 6, 7)$$

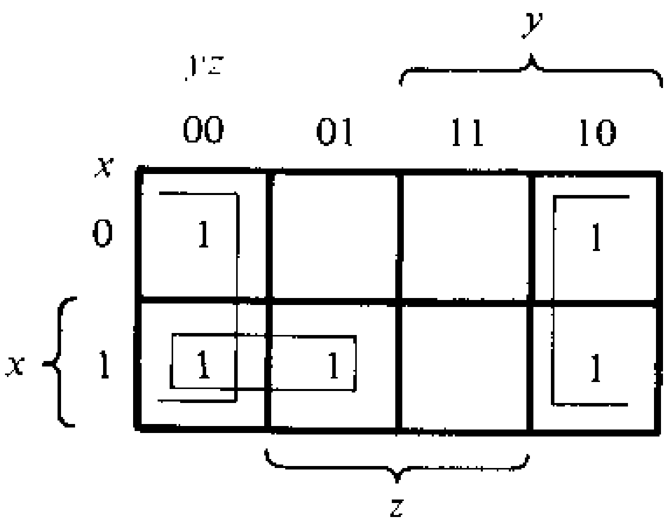


$$F = A'C + A'B + AB'C + BC$$



Simplify the Boolean function

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$



$$\Sigma(0, 2, 4, 5, 6) = z' + xy'$$

$$A'C + A'B + AB'C + BC = C + A'B$$

4- Variable K-Map

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

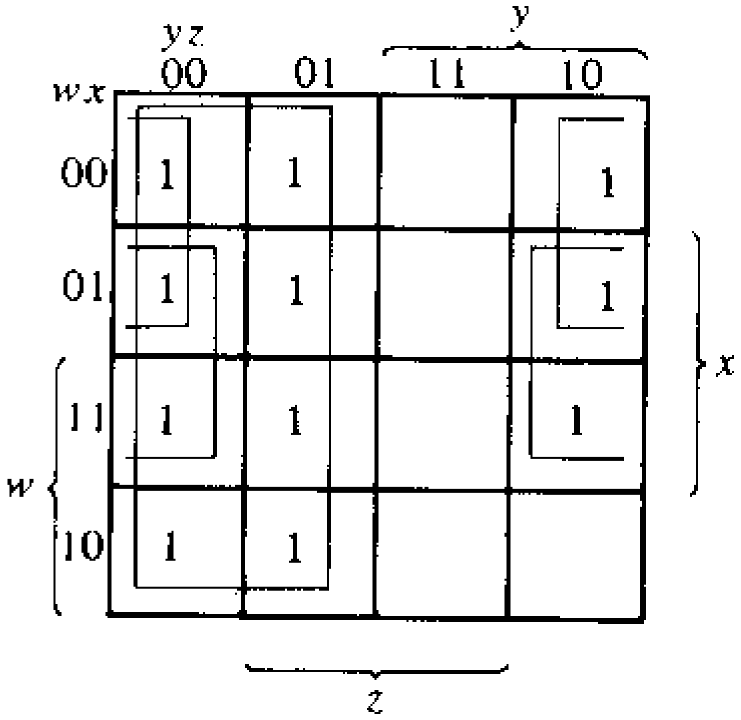
(a)

$wx \backslash yz$	00	01	11	10
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

(b)

Simplify the Boolean function

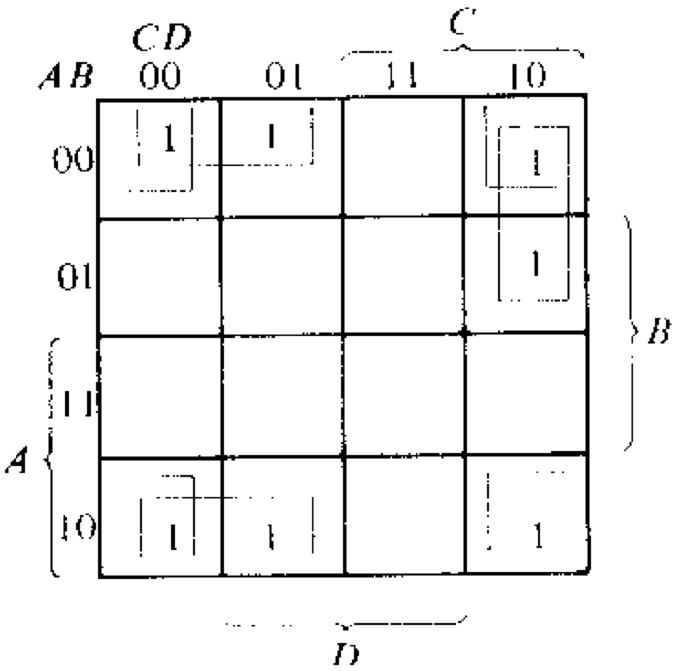
$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$



$$y' + w'z' + xz'$$

Simplify the Boolean function

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$



$$B'D' + B'C' + A'CD'$$

PRODUCT OF SUMS SIMPLIFICATION

Simplify the following Boolean function in (a) sum of products and (b) product of sums.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

$$F = B'D' + B'C' + A'C'D$$

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

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

C (columns 11, 10); D (columns 01, 11); A (rows 11, 10); B (rows 01, 11)

Completely and Incompletely Specified Logic Functions

Logical functions are of two types:

1. completely specified logical function
2. incompletely specified logical function

A logical function whose output is specified for all possible input combination called completely specified logical function.

A logical function whose output may not be specified for certain input combinations/conditions or for which a certain input combinations may never occur is called incomplete specified logical function.

Inputs			Output
A	B	C	F
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	X
1	1	1	X

Minimization of Incompletely Specified Logic Functions

- For a four bit binary 9's complementary circuit input > 9 is not possible or output for input > 9 is do not care. So, we put the do not care conditions for all *WXYZ* in the O/P.

$$w = \sum m(0,1) + \sum d(10,11,12,13,14,15) = f(A,B,C,D)$$

$$X = \sum m\ (2, 3, 4, 5) + \sum d\ (10, 11, 12, 13, 14, 15) = f(A, B, C, D)$$

$$Y = \sum m\ (2, 3, 6, 7) + \sum d\ (10, 11, 12, 13, 14, 15) = f(A, B, C, D)$$

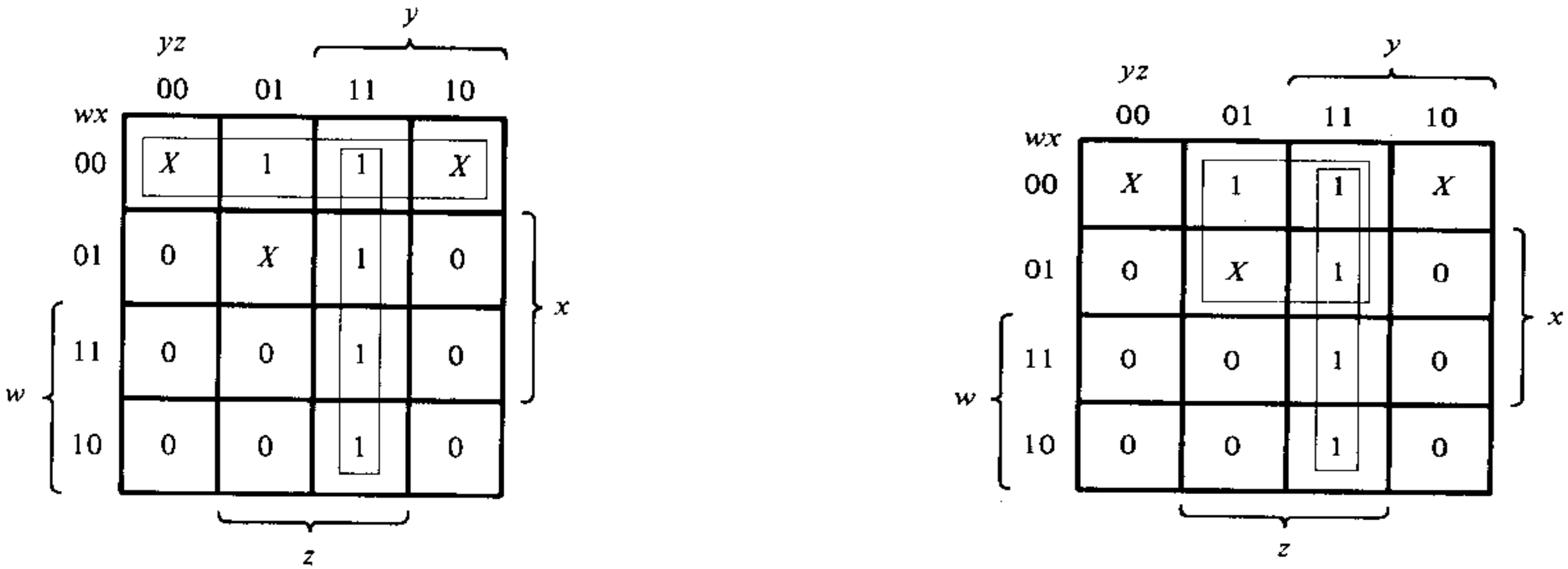
$$Z = \sum m\ (0, 2, 4, 6, 8) + \sum d\ (10, 11, 12, 13, 14, 15)$$

Simplify the Boolean function

$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

that has the don't-care conditions

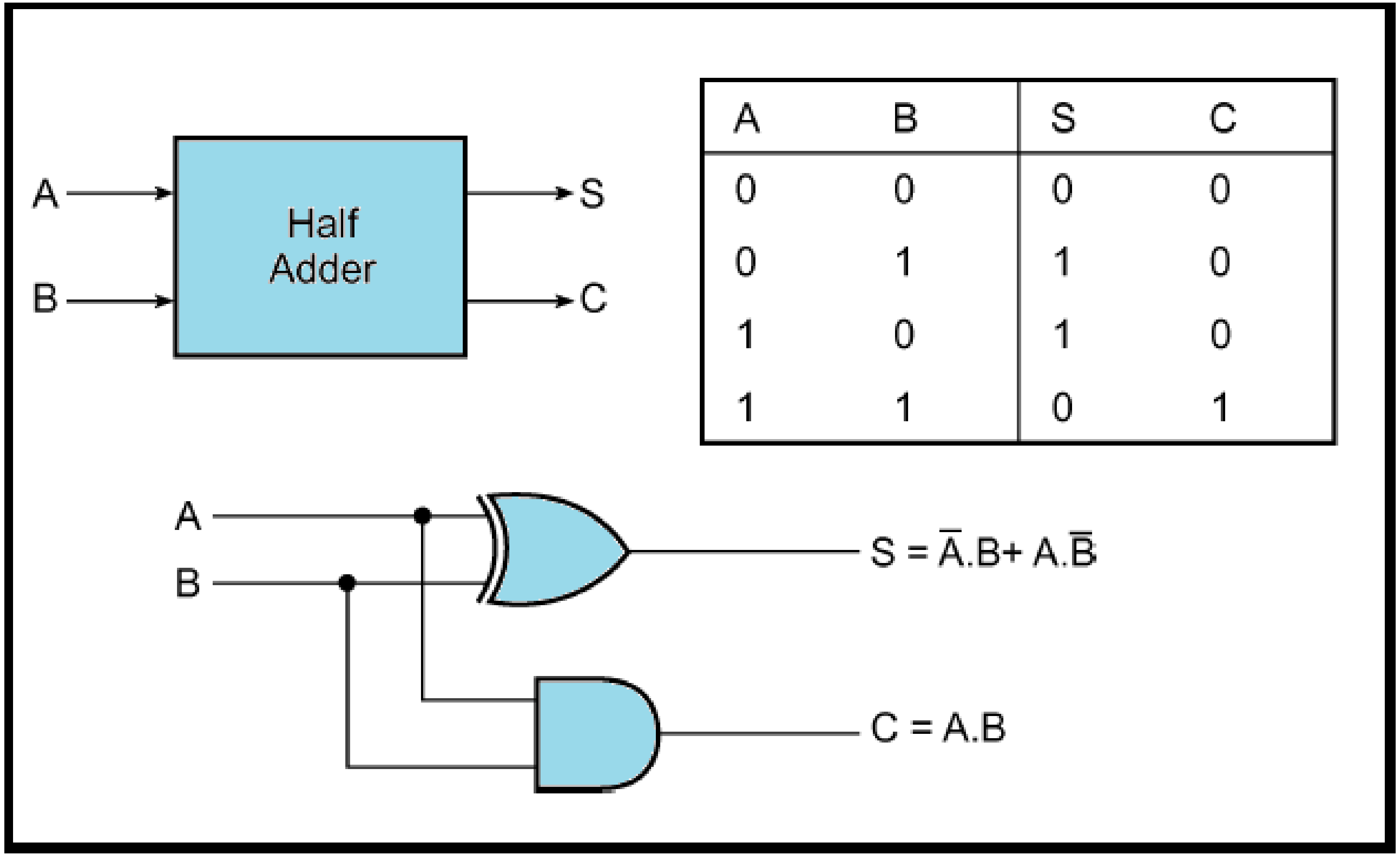
$$d(w, x, y, z) = \Sigma(0, 2, 5)$$



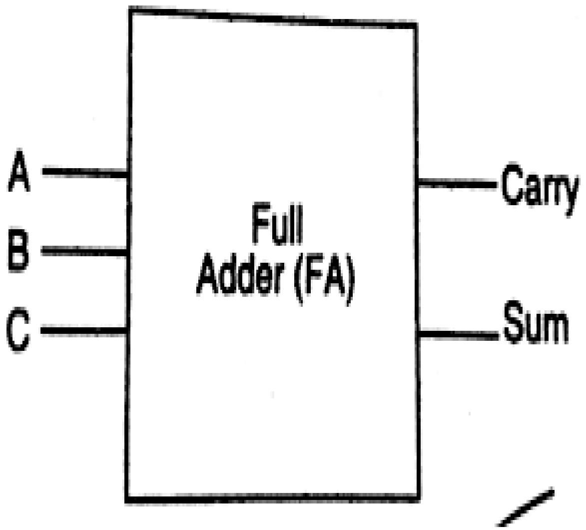
(a) $F = yz + w'x'$

(b) $F = yz + w'z$

Combinational Logic



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

For sum

A \ BC	00	01	11	10
0		1		1
1	1		1	

$$\therefore \text{Sum} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC)$$

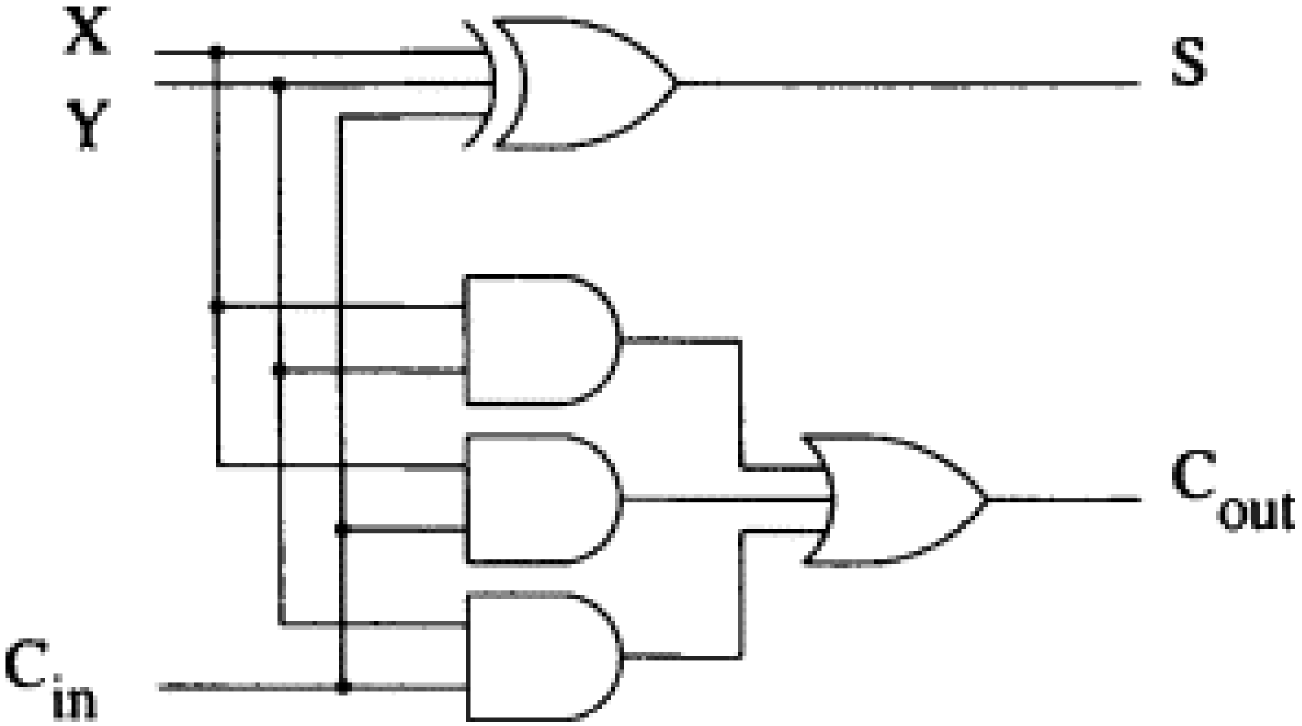
$$= \bar{A}(B \oplus C) + A(\overline{B \oplus C}) \qquad (\because \bar{x}y + x\bar{y} = x \oplus y)$$

$$= A \oplus B \oplus C$$

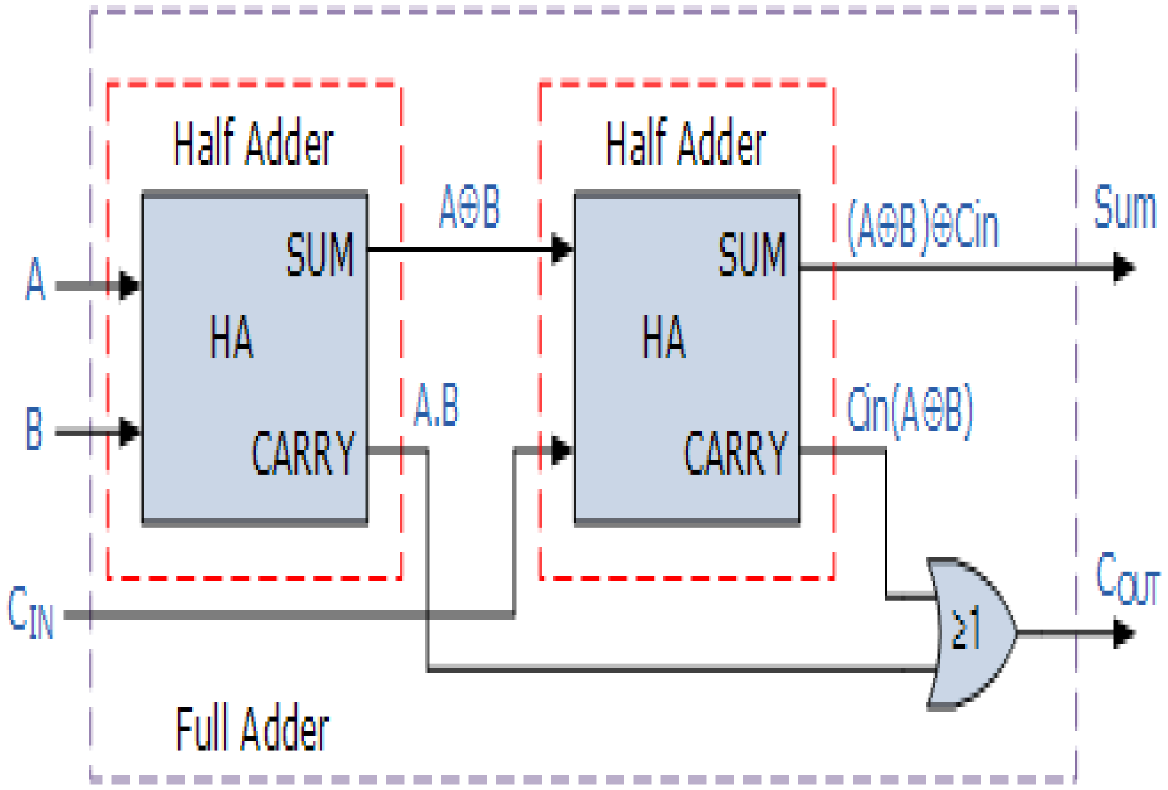
For carry

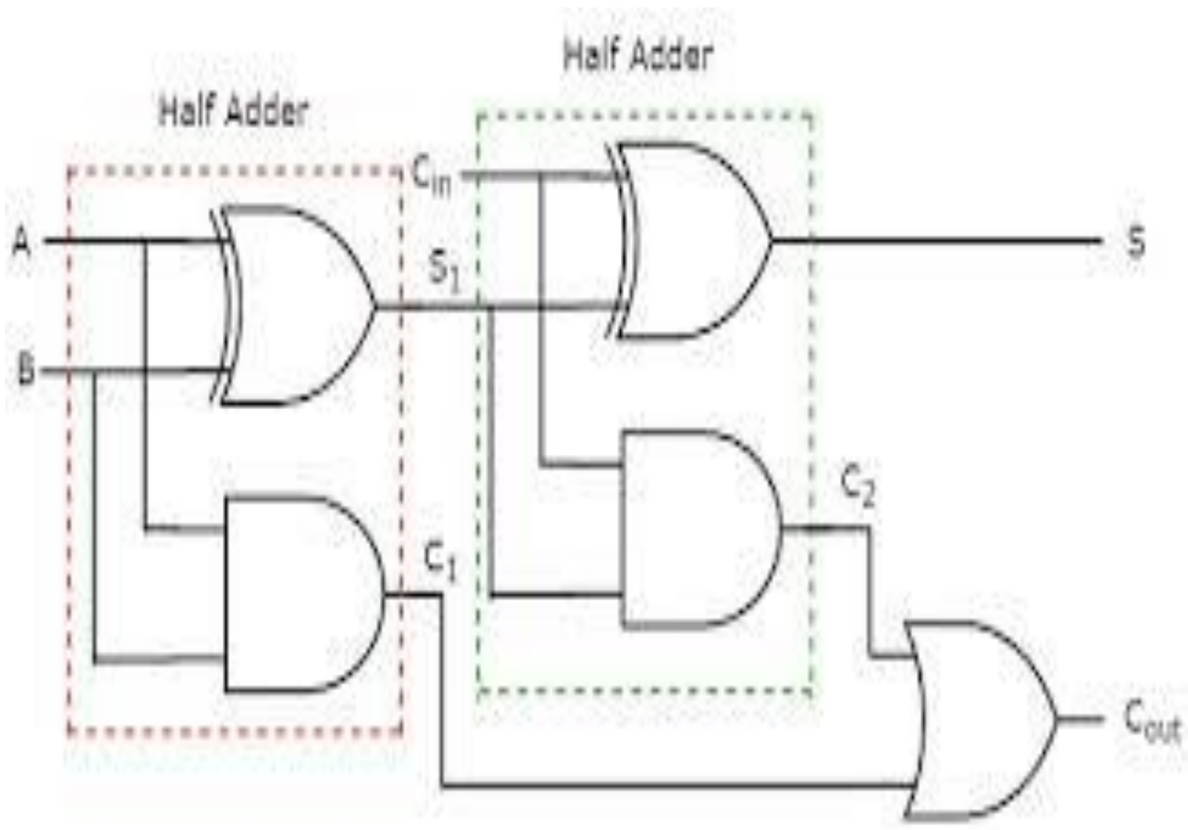
A \ BC	00	01	11	10
0			1	
1		1	1	1

Carry = AB+BC+AC



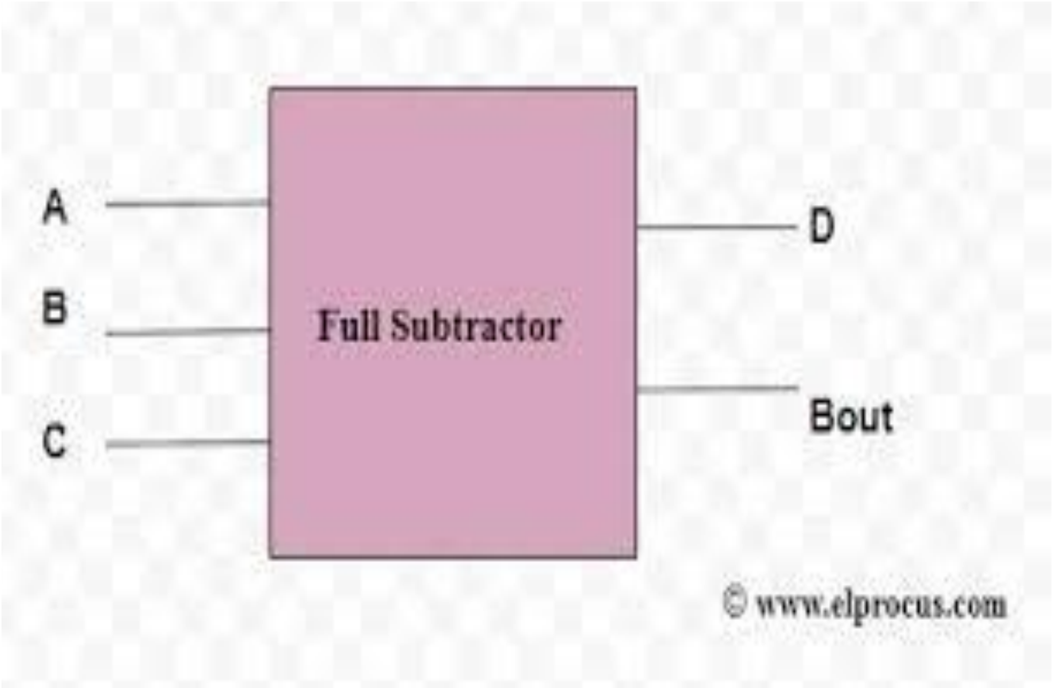
FULL ADDER Using HALF ADDERS





Full Subtractor

INPUT			OUTPUT	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



For Difference :-

A \ BB_{in}	$\bar{B}\bar{B}_{in}$ 00	$\bar{B}B_{in}$ 01	BB_{in} 11	$B\bar{B}_{in}$ 10
$\bar{A}$ 0		1		1
A 1	1		1	

$\therefore$ $\boxed{\text{Difference} = A \oplus B \oplus B_{in}}$

$\therefore$ Difference $= \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

$= \bar{A}(\bar{B}B_{in} + B\bar{B}_{in}) + A(\bar{B}\bar{B}_{in} + BB_{in})$

$= \bar{A}(B \oplus B_{in}) + A(B \odot B_{in}) = \bar{A}(B \oplus B_{in}) + A(\overline{B \oplus B_{in}})$

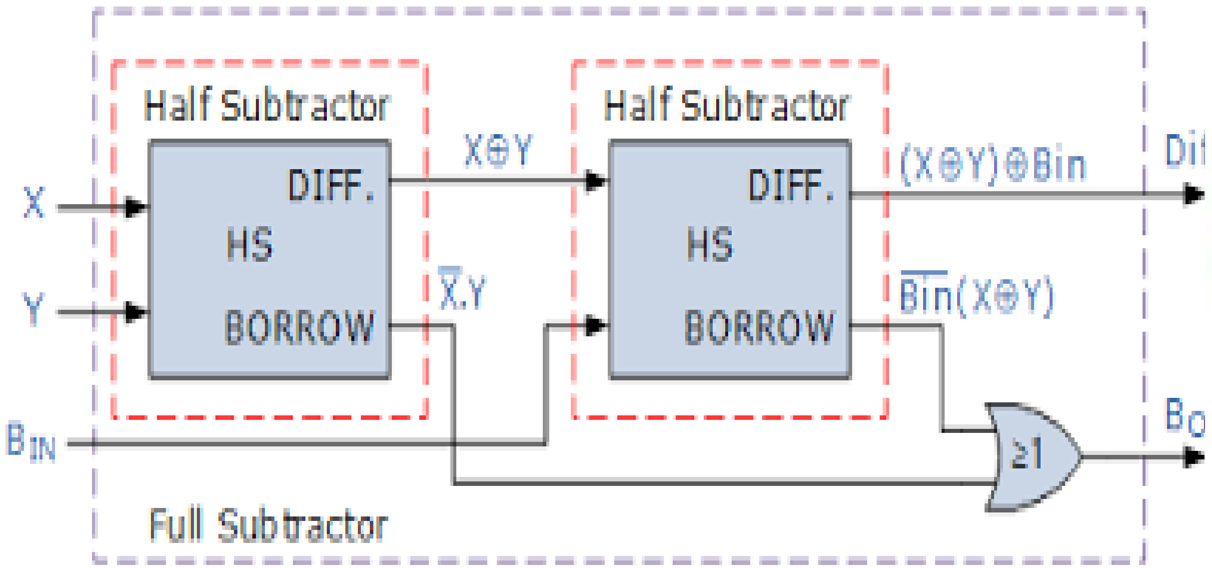
$= A \oplus B \oplus B_{in} = A \oplus B \oplus B_{in}$

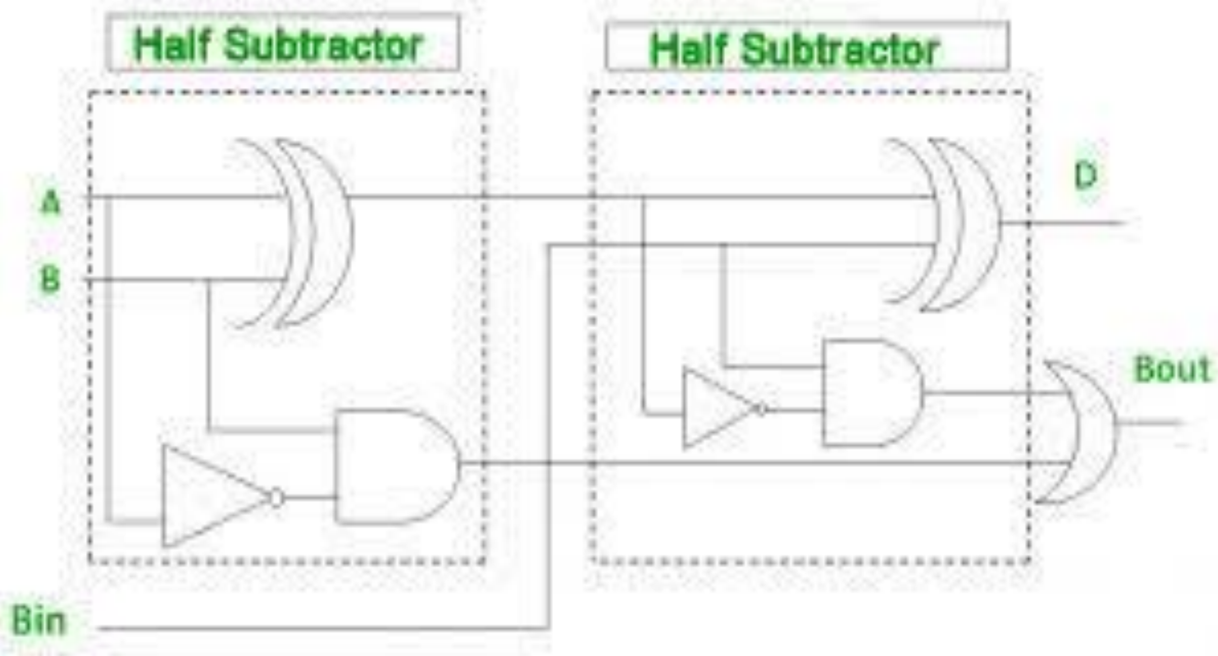
For B_{out} :-

A \ BB_{in}	$\bar{B}\bar{B}_{in}$ 00	$\bar{B}B_{in}$ 01	BB_{in} 11	$B\bar{B}_{in}$ 10
$\bar{A}$ 0		1	1	1
A 1			1	

$\therefore$ $\boxed{B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}}$

$\therefore$ $B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}$


Half Subtractor
Half Subtractor
X
Y
DIFF.
HS
BORROW
X⊕Y
X̄.Y
DIFF.
HS
BORROW
(X⊕Y)⊕Bin
B̄in(X⊕Y)
≥1
B
IN
Full Subtractor


Half Subtractor
Half Subtractor
A
B
D
Bout
Bin

Binary to Gray Code Conversion

Decimal Number	4 bit Binary Number ABCD	4 bit Gray Code $G_1G_2G_3G_4$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

Simplification using K-maps:

G_3

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$\boxed{G_3 = B_3}$$

G_2

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$G_2 = \overline{B_3}\, B_2 + B_3\, \overline{B_2}$$

$$\boxed{G_2 = B_3 \oplus B_2}$$

G_1

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$$G_1 = B_2\, \overline{B_1} + \overline{B_2}\, B_1$$

$$\boxed{G_1 = B_2 \oplus B_1}$$

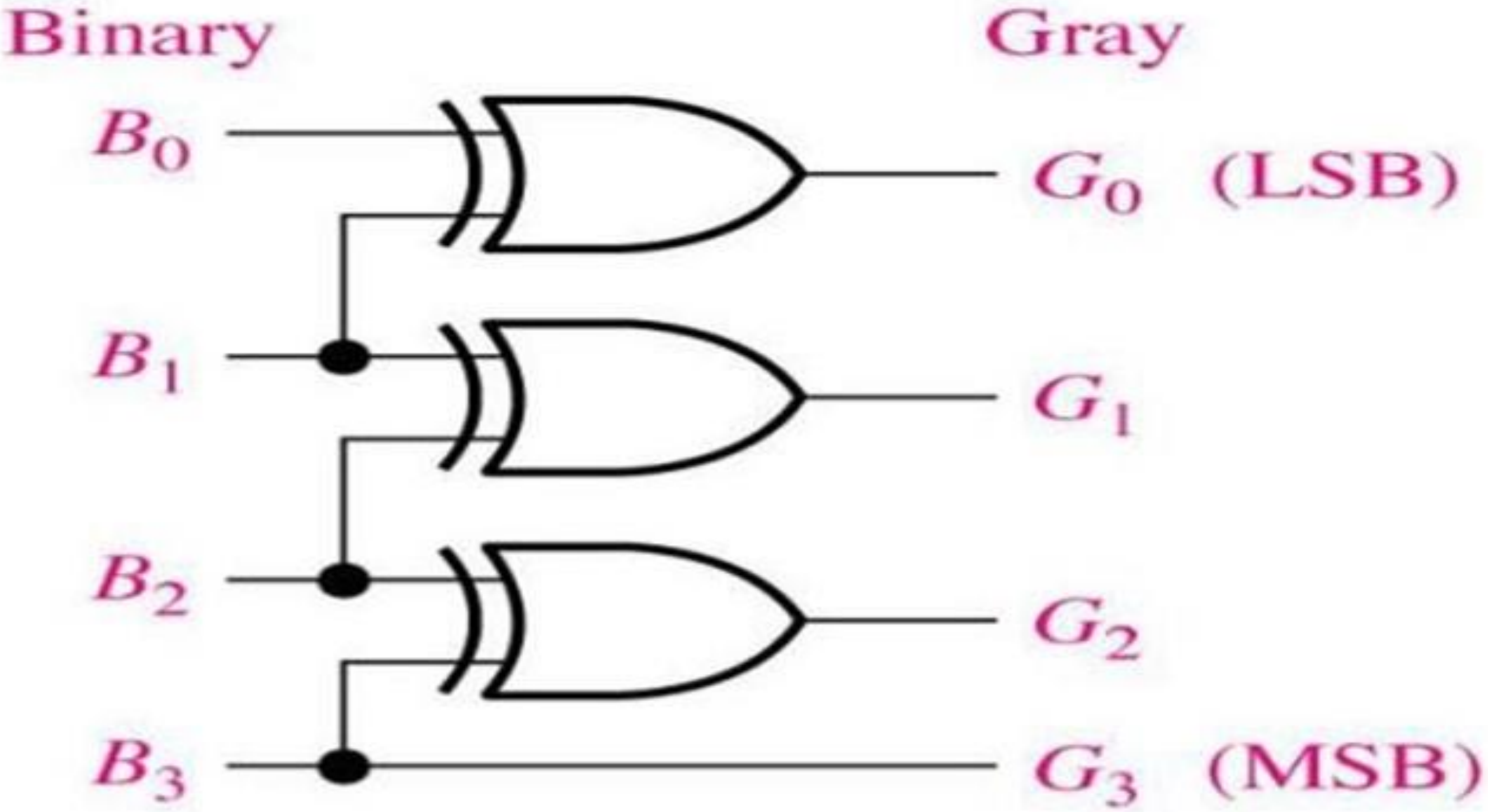
G_0

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$G_0 = \overline{B_1}\, B_0 + B_1\, \overline{B_0}$$

$$\boxed{G_0 = B_1 \oplus B_0}$$

Logic Diagram:



Gray Code to Binary Code Conversion

- **Truth Table:**

INPUT (GRAY CODE)					OUTPUTS (BINARY)			
G3	G2	G1	G0		B3	B2	B1	B0
0	0	0	0		0	0	0	0
0	0	0	1		0	0	0	1
0	0	1	0		0	0	1	1
0	0	1	1		0	0	1	0
0	1	0	0		0	1	1	1
0	1	0	1		0	1	1	0
0	1	1	0		0	1	0	0
0	1	1	1		0	1	0	1
1	0	0	0		1	1	1	1
1	0	0	1		1	1	1	0
1	0	1	0		1	1	0	0
1	0	1	1		1	1	0	1
1	1	0	0		1	0	0	0
1	1	0	1		1	0	0	1
1	1	1	0		1	0	1	1
1	1	1	1		1	0	1	0

Simplification using K-Maps:

B_3

$G_3G_2 \backslash G_1G_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$\boxed{B_3 = G_3}$$

B_2

$G_3G_2 \backslash G_1G_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$B_2 = \overline{G_3}\,G_2 + G_3\,\overline{G_2}$$

$$\boxed{B_2 = G_3 \oplus G_2}$$

B_1

$G_3G_2 \backslash G_1G_0$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

$$B_1 = \overline{G_3}\,\overline{G_2}\,G_1 + G_3G_2G_1 + \overline{G_3}\,G_2\,\overline{G_1} + G_3\,\overline{G_2}\,\overline{G_1}$$

$$= G_1(\overline{G_3}\,\overline{G_2} + G_3G_2) + \overline{G_1}(\overline{G_3}\,G_2 + G_3\,\overline{G_2})$$

$$= G_1(\overline{G_3 \oplus G_2}) + \overline{G_1}(G_3 \oplus G_2)$$

$$= G_1 \oplus G_3 \oplus G_2$$

$$\boxed{B_1 = G_1 \oplus B_2}$$

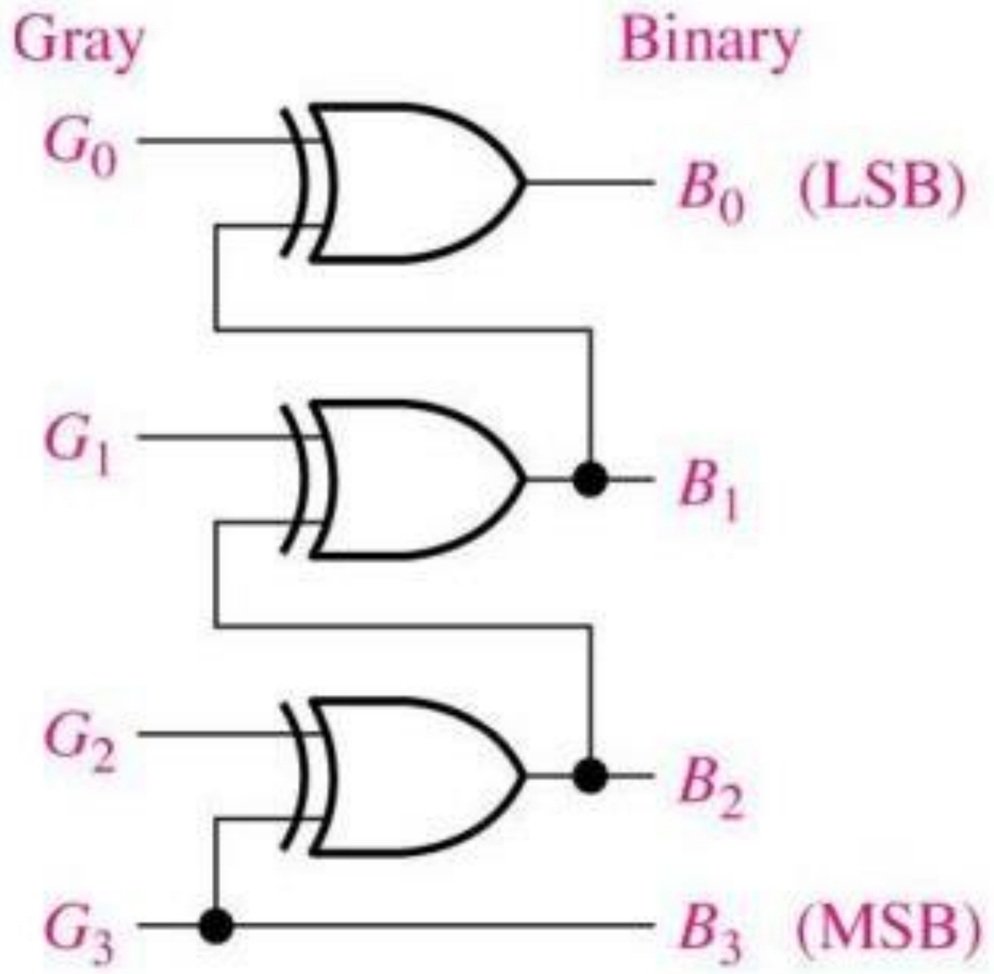
Simplification using K-Maps:

$\underline{B_0}$

$G_3G_2 \backslash G_1G_0$	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

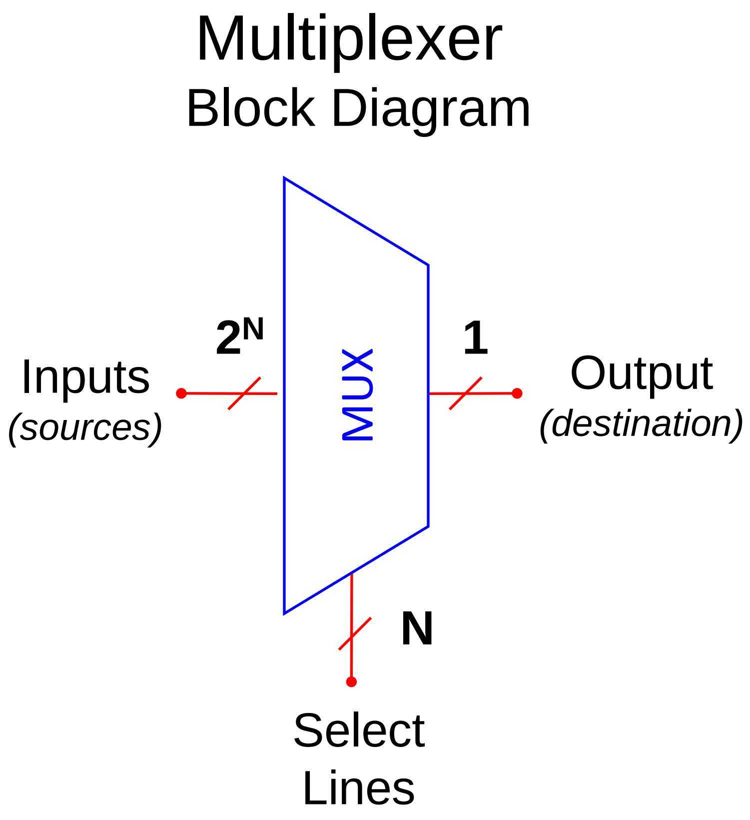
$$B_0 = \overline{G_3}\,\overline{G_2}\,\overline{G_1}G_0 + \overline{G_3}\,\overline{G_2}G_1\overline{G_0} + \overline{G_3}G_2\overline{G_1}\,\overline{G_0} + \overline{G_3}G_2G_1G_0$$
$$+ G_3G_2\overline{G_1}G_0 + G_3G_2G_1\overline{G_0} + G_3\overline{G_2}\,\overline{G_1}\,\overline{G_0} + G_3\overline{G_2}G_1G_0$$
$$= \overline{G_3}\,\overline{G_2}(\overline{G_1}G_0 + G_1\overline{G_0}) + \overline{G_3}G_2(\overline{G_1}\,\overline{G_0} + G_1G_0)$$
$$+ G_3G_2(\overline{G_1}G_0 + G_1\overline{G_0}) + G_3\overline{G_2}(\overline{G_1}\,\overline{G_0} + G_1G_0)$$
$$= \overline{G_3}\,\overline{G_2}(G_1 \oplus G_0) + \overline{G_3}G_2(\overline{G_1 \oplus G_0})$$
$$+ G_3G_2(G_1 \oplus G_0) + G_3\overline{G_2}(\overline{G_1 \oplus G_0})$$
$$= (G_1 \oplus G_0)(\overline{G_3}\,\overline{G_2} + G_3G_2) + (\overline{G_1 \oplus G_0})(\overline{G_3}G_2 + G_3\overline{G_2})$$
$$= (G_1 \oplus G_0)(\overline{G_3 \oplus G_2}) + (\overline{G_1 \oplus G_0})(G_3 \oplus G_2)$$
$$= G_0 \oplus G_1 \oplus G_2 \oplus G_3$$
$$\boxed{B_0 = G_0 \oplus B_1}$$

Logic Diagram:

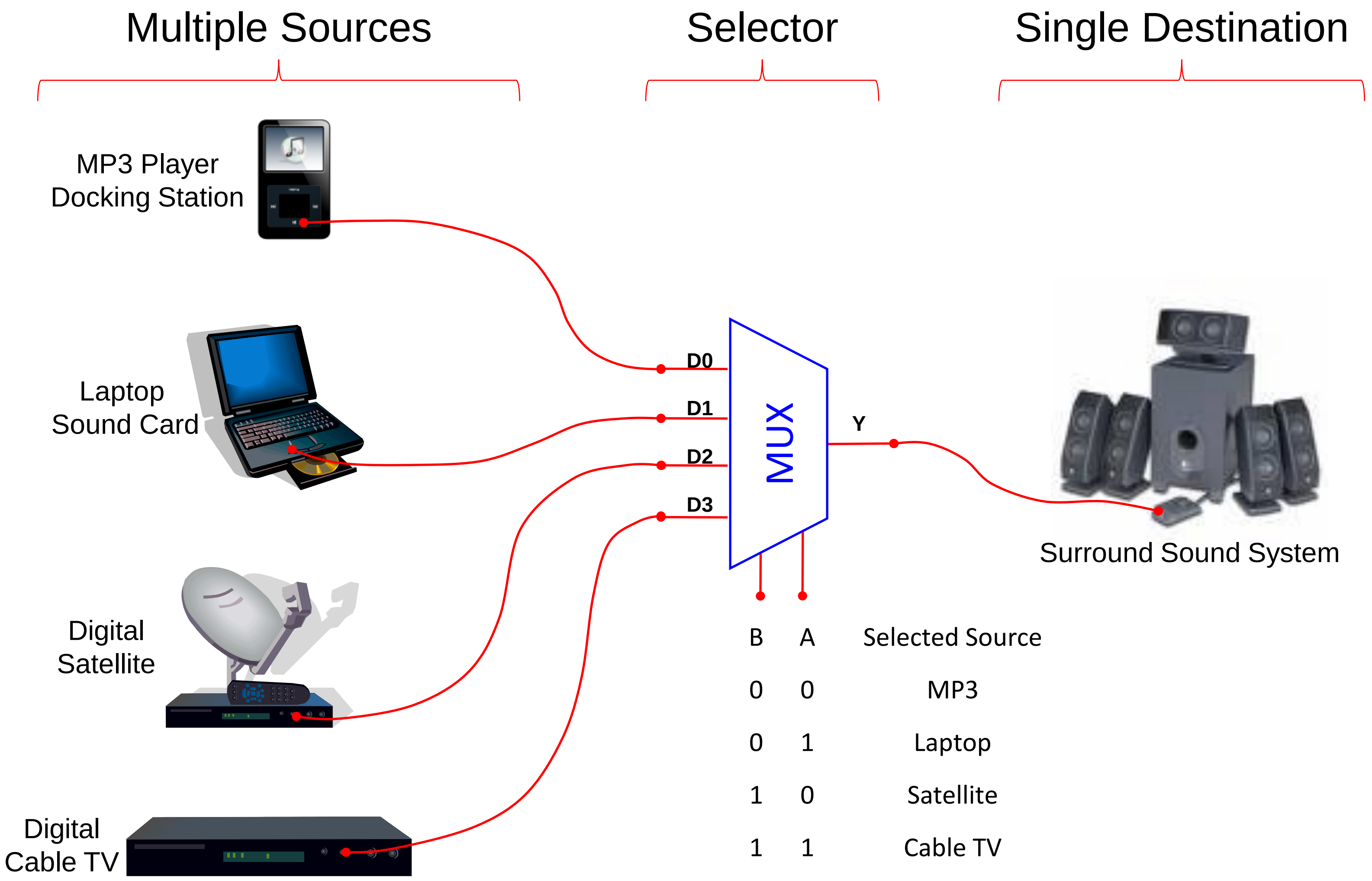


What is a Multiplexer (MUX)?

- A MUX is a digital switch that has multiple inputs (sources) and a single output (destination).
- The select lines determine which input is connected to the output.
- MUX Types
  - → 2-to-1 (1 select line)
  - → 4-to-1 (2 select lines)
  - → 8-to-1 (3 select lines)
  - → 16-to-1 (4 select lines)



Typical Application of a MUX

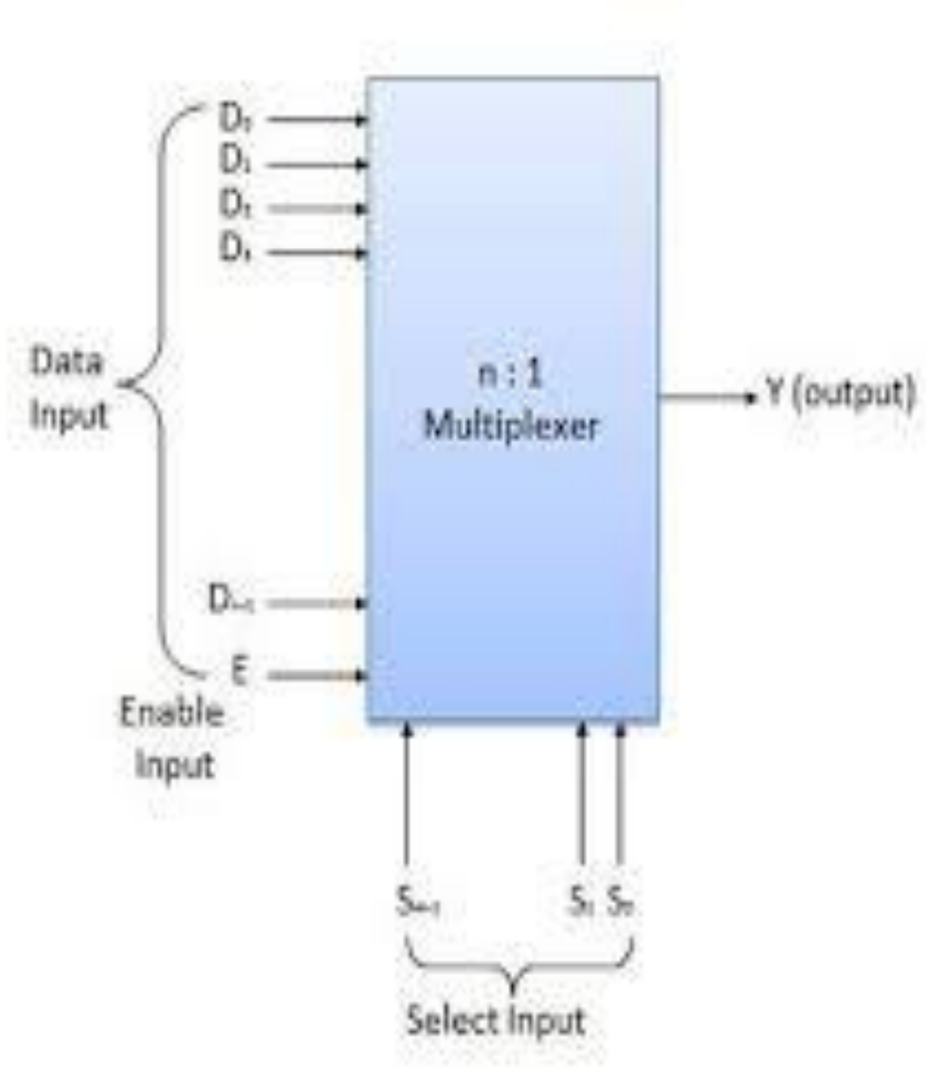


B	A	Selected Source
0	0	MP3
0	1	Laptop
1	0	Satellite
1	1	Cable TV

Multiplexer/Demultiplexers

Multiplexing means Generally a (n : 1) multiplexer or a data selector contains the following:

1. n number of inputs of the multiplexer
2. only one output of the multiplexer
3. m select/Address lines
4. one strobe/enable input (optional)



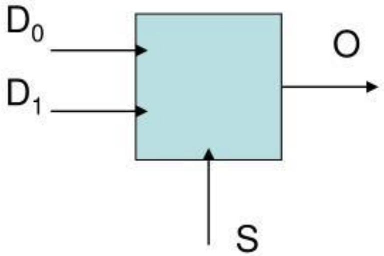
All inputs and outputs should be purely logical or binary input.

For *n* : 1 multiplexer, each input will be selected at the single output at any instant of time, depends upon the input applied to the select/address inputs. Switching circuit connects a particular input to the output.

Here *m* and *n* are related by $2^m = n$

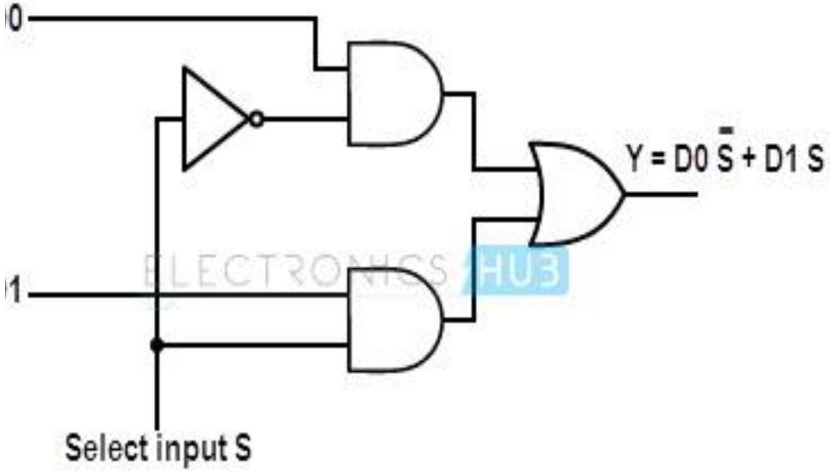
Design of a 2/1 Mux

- 2/1 mux Block Diagram

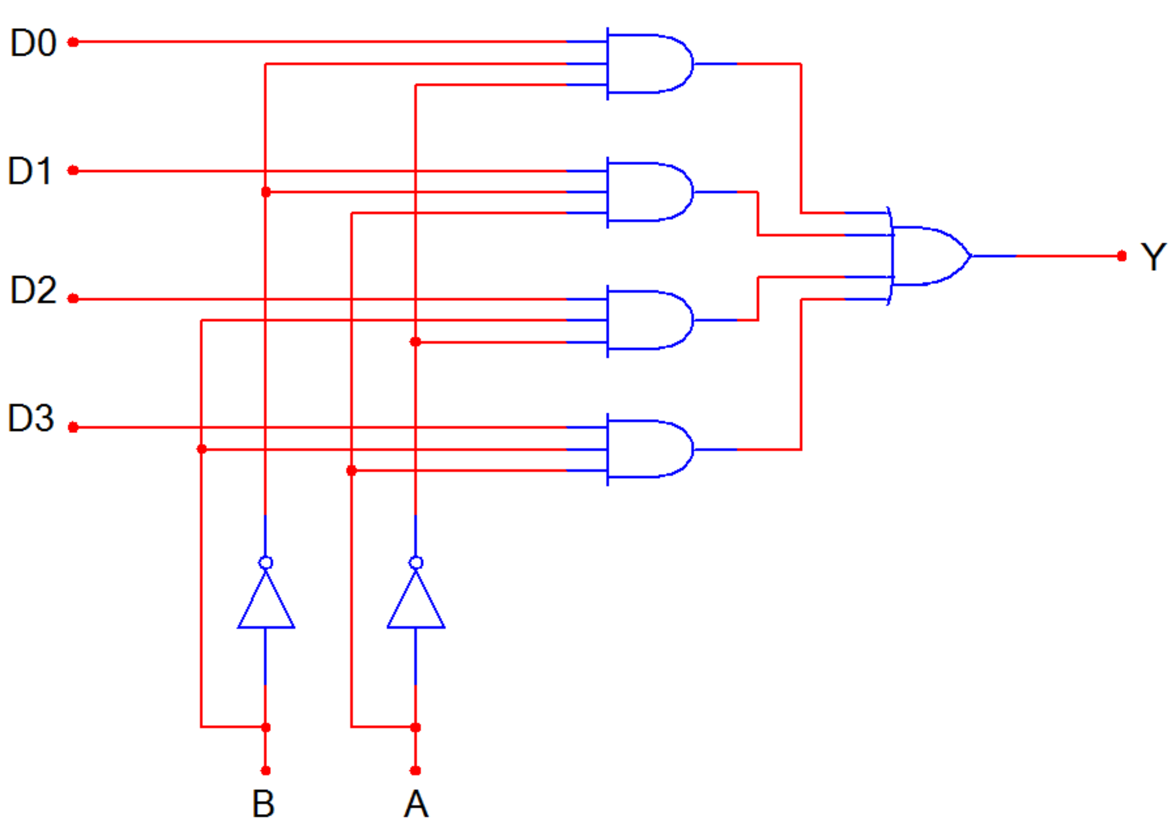


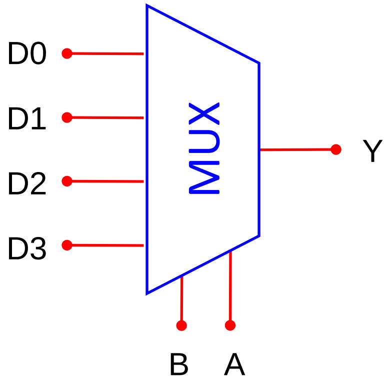
- Truth Table

S	D_1	D_0	O
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



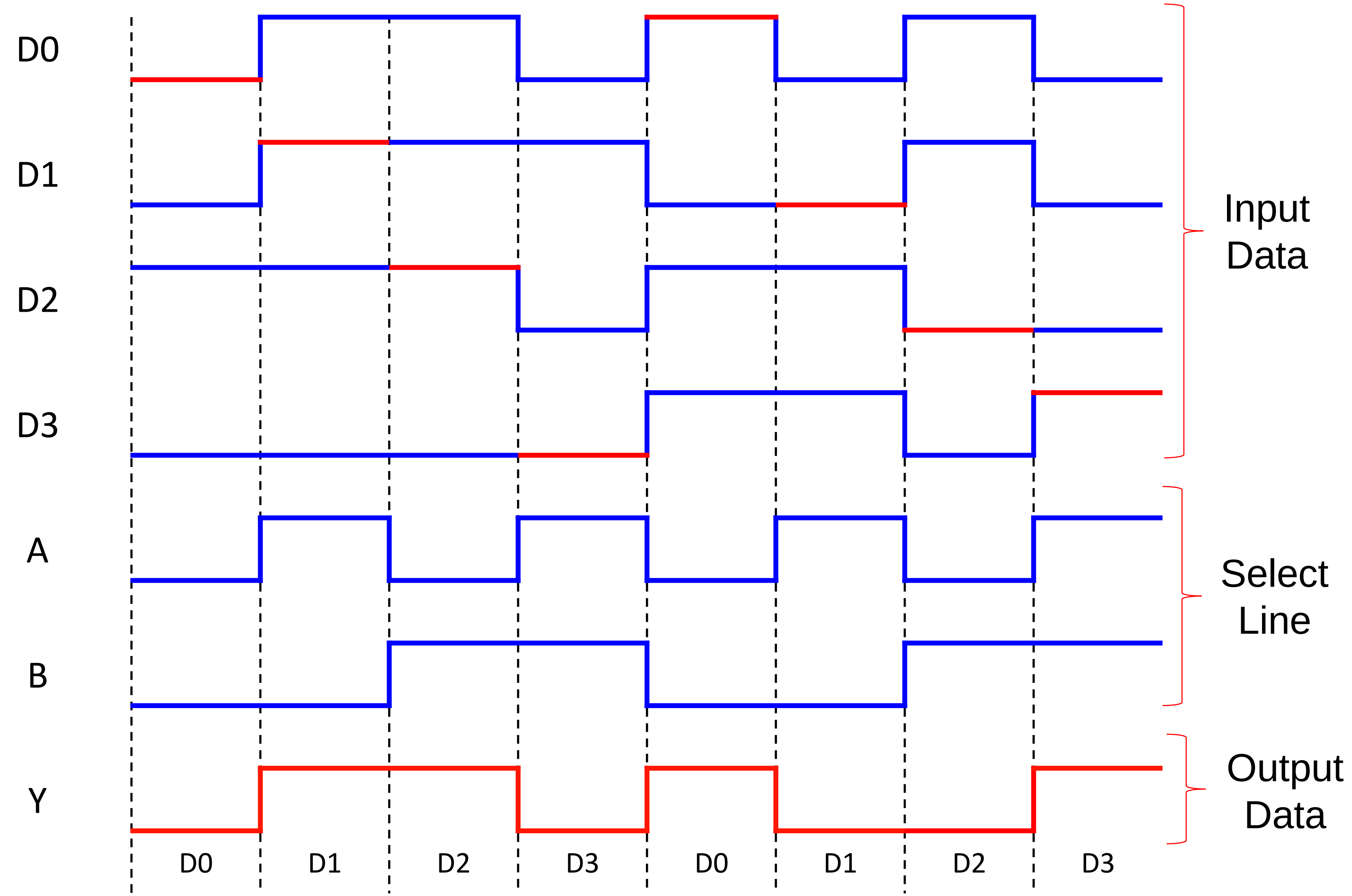
4-to-1 Multiplexer (MUX)





B	A	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

4-to-1 Multiplexer Waveforms



Medium Scale Integration MUX

4-to-1 MUX

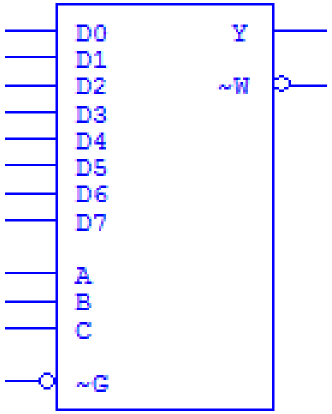
Inputs

Select

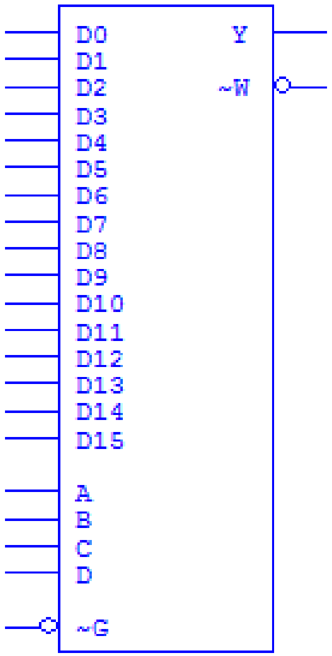
Enable

Output (Y)
(and inverted output)

8-to-1 MUX



16-to-1 MUX

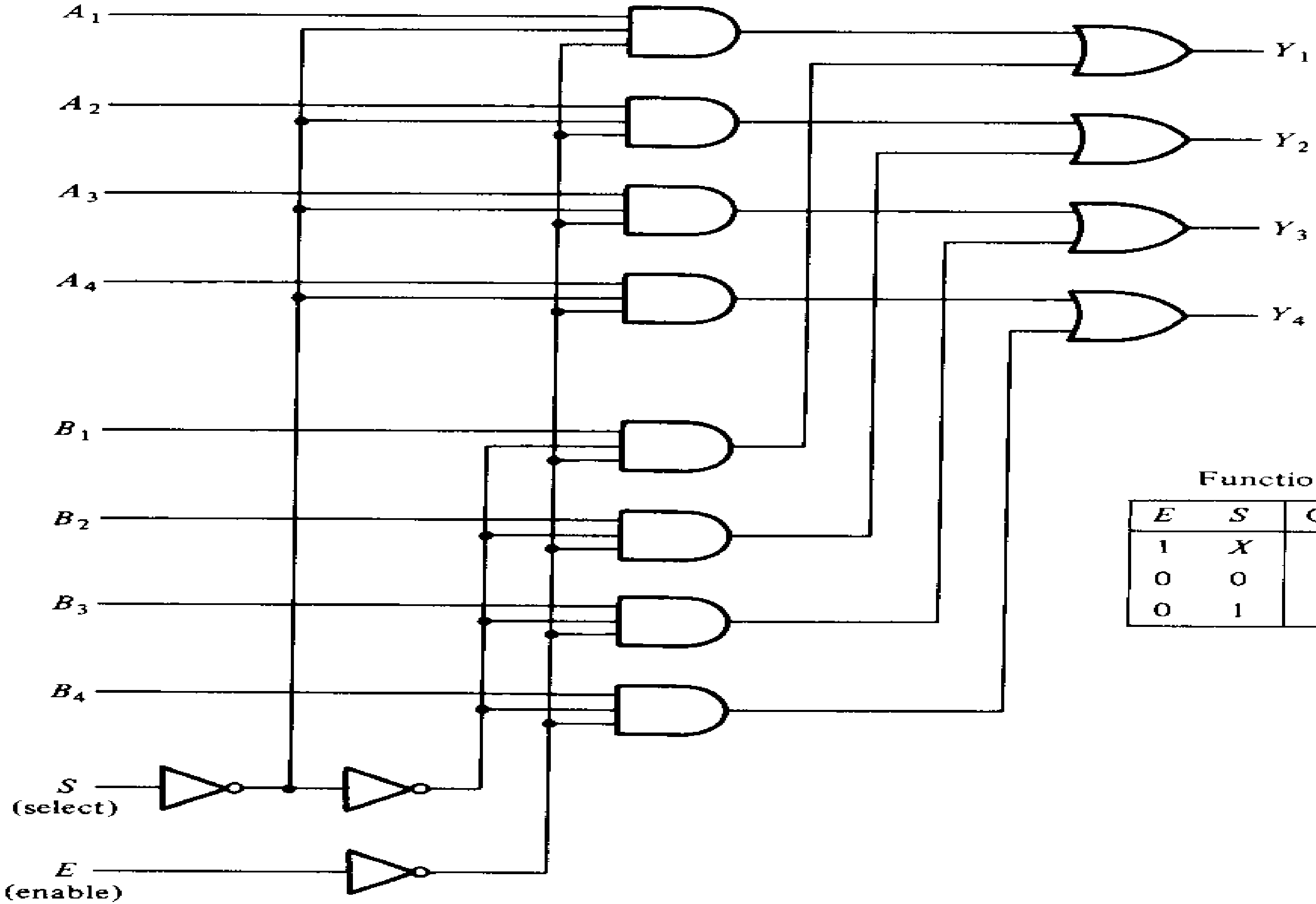


When two or more MUX are enclosed with in single chip

Selection and enable inputs in multiple unit IC may be common to all multiplexers. A Quadruple 2 to 1 line multiplexer IC is shown here.

It has 4 MUX each capable of selecting one of 2 input lines.

O/P Y1 can be selected to be equal to either A1 or B1..similarly others



Function table

E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

Boolean Function Implementation

For implementing any Boolean function of n variable with 2^n to 1 MUX is possible. However more optimize way is also there.

If we have Boolean function of n+1 variable then out of it 1 is used for input data and rest n are used as select I/P.

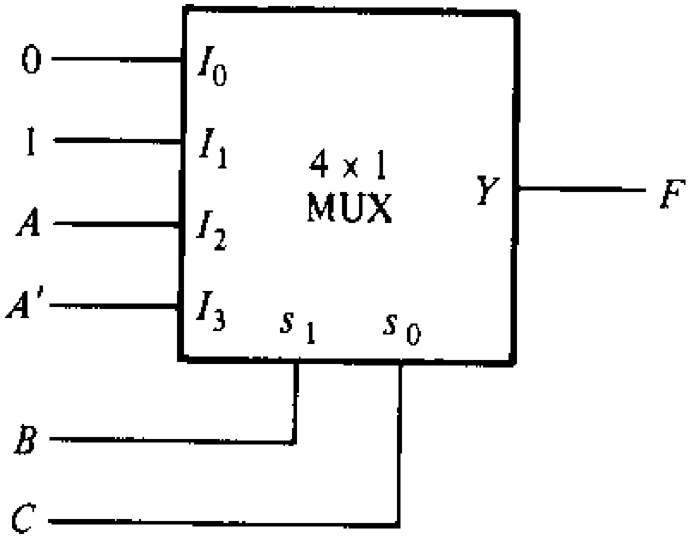
If A is a single variable then input of MUX are to be chosen either A or A' or 1 or 0.

By judicious use of these four values to inputs and others as select I/P variables one can implement any Boolean function of n+1 variables using 2^n to 1 MUX.

$$F(A, B, C) = \Sigma(1, 3, 5, 6)$$

	I_0	I_1	I_2	I_3
A'	0	1	2	3
A	4	5	6	7
	0	1	A	A'

(c) Implementation table



(a) Multiplexer implementation

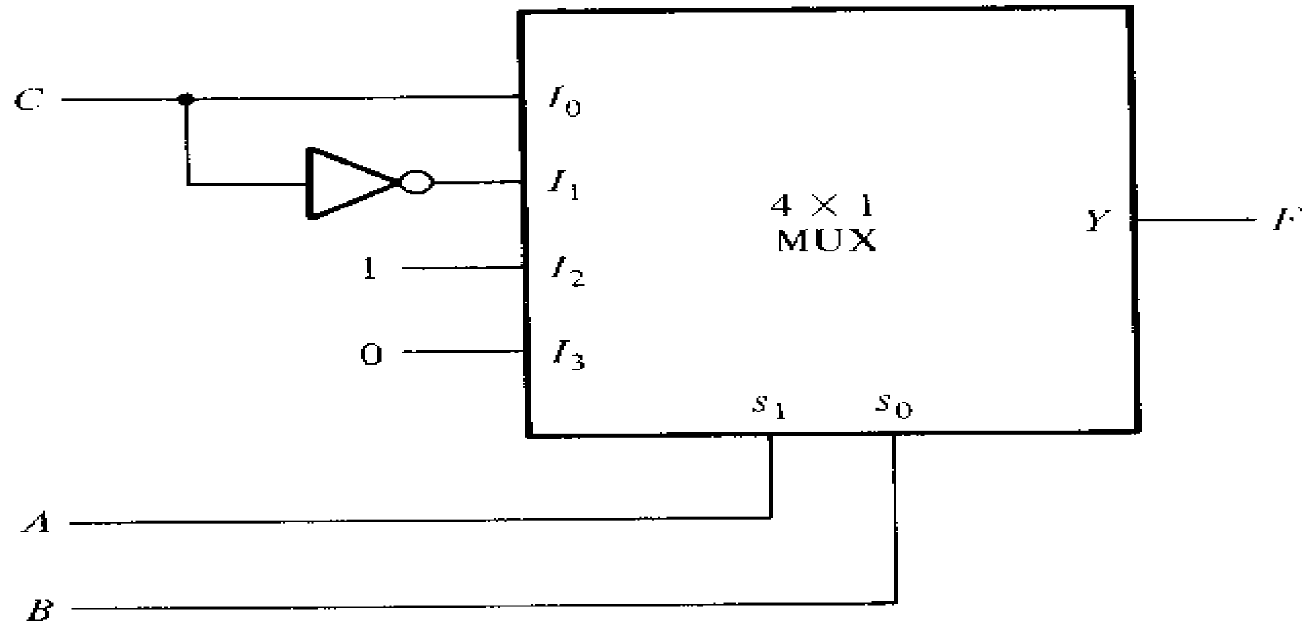
Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(b) Truth table

$$F(A, B, C) = \Sigma(1, 2, 4, 5)$$

A	B	C	F	
0	0	0	0	$F = C$
0	0	1	1	
0	1	0	1	$F = C'$
0	1	1	0	
1	0	0	1	$F = 1$
1	0	1	1	
1	1	0	0	$F = 0$
1	1	1	0	

(a) Truth table



(b) Multiplexer implementation

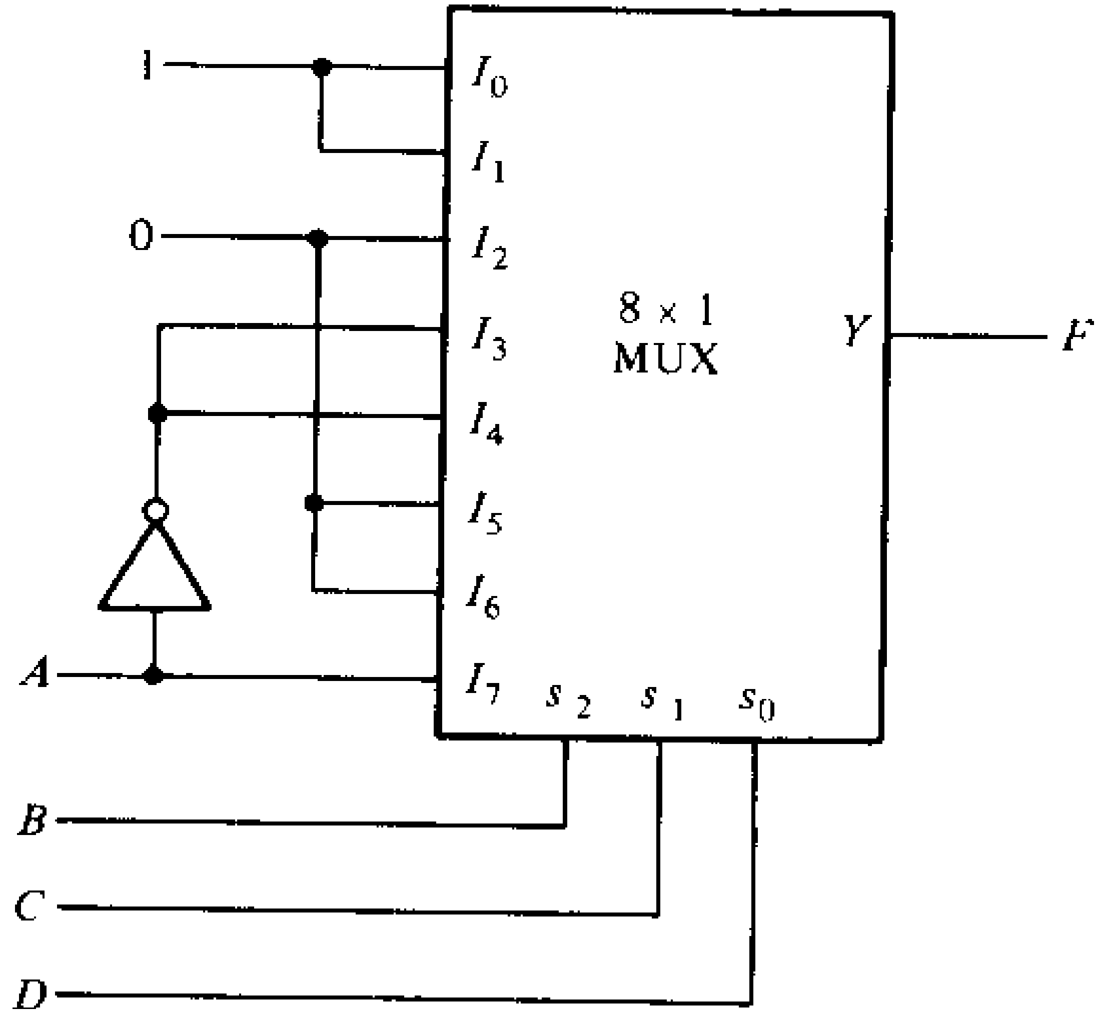
	I_0	I_1	I_2	I_3
C'	0	2	4	6
C	1	3	5	7
	C	C'	1	0

(c) Implementation table

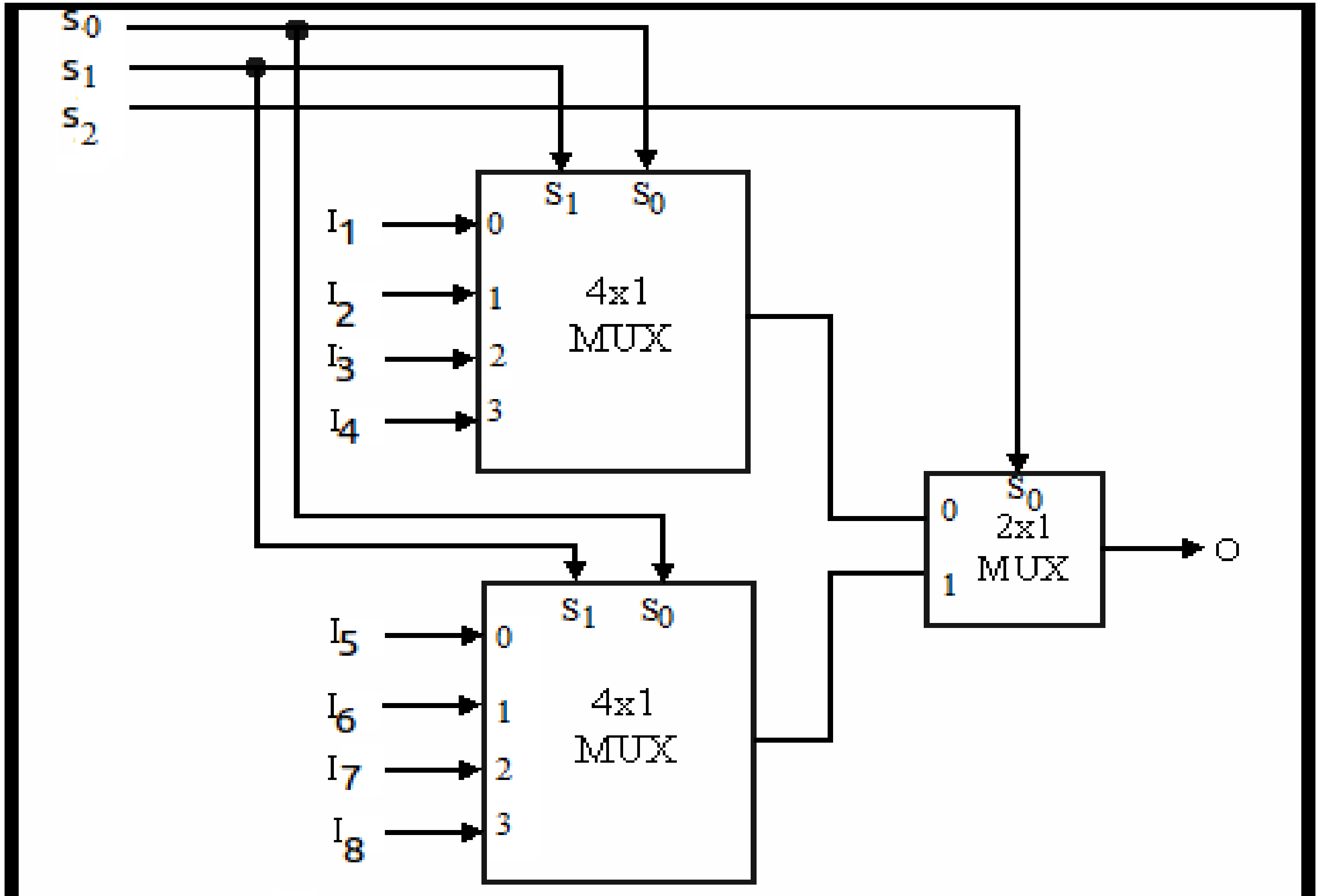
Implement the following function with a multiplexer:

$$F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9, 15)$$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	A'	A'	0	0	A

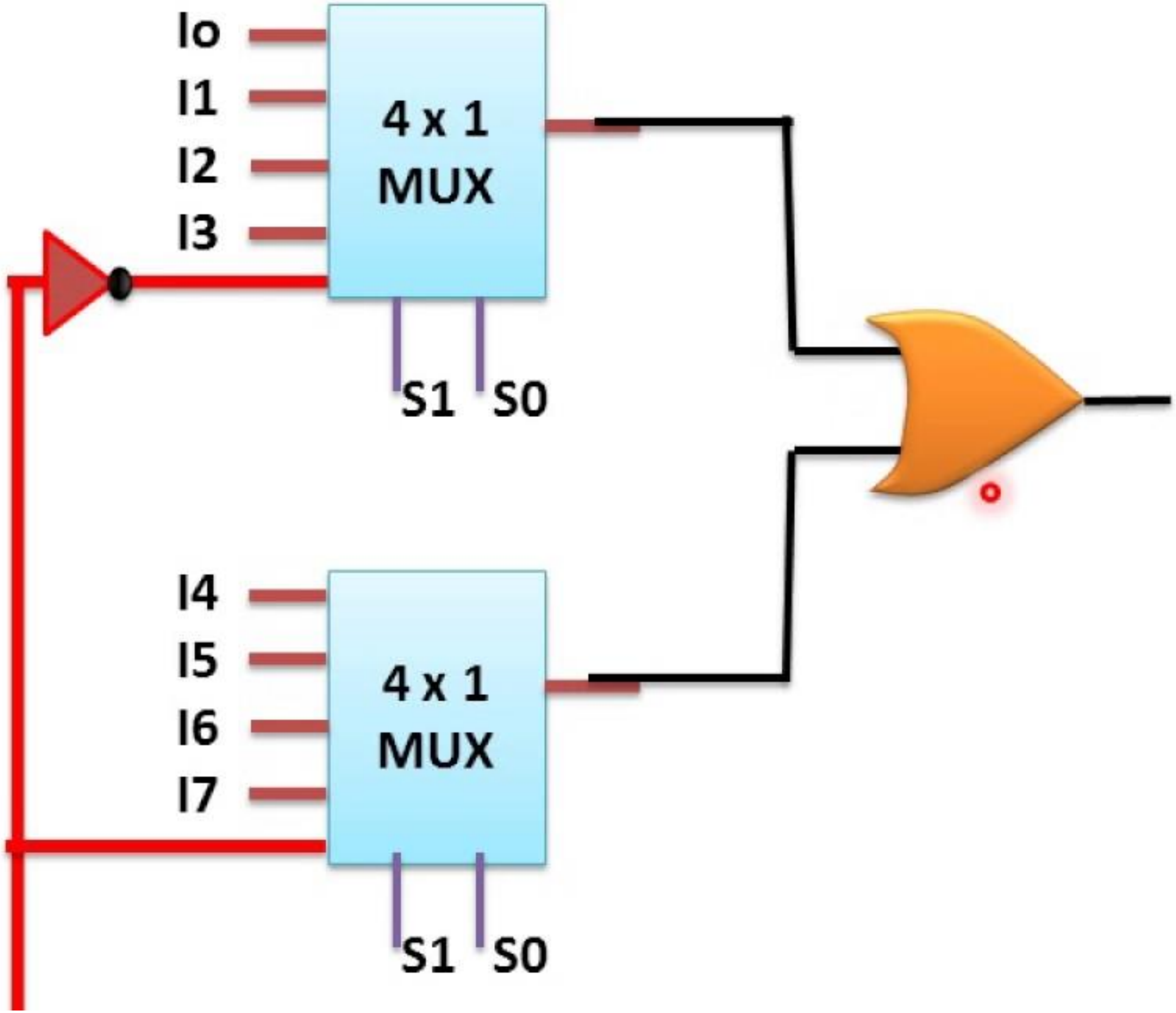


Higher order MUX 8:1 MUX using 4:1 MUXs



Implementation of 8x1 MUX using 4x1 and 2x1 MUXs

8:1 MUX using two 4:1 MUX



S2	S1	S0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

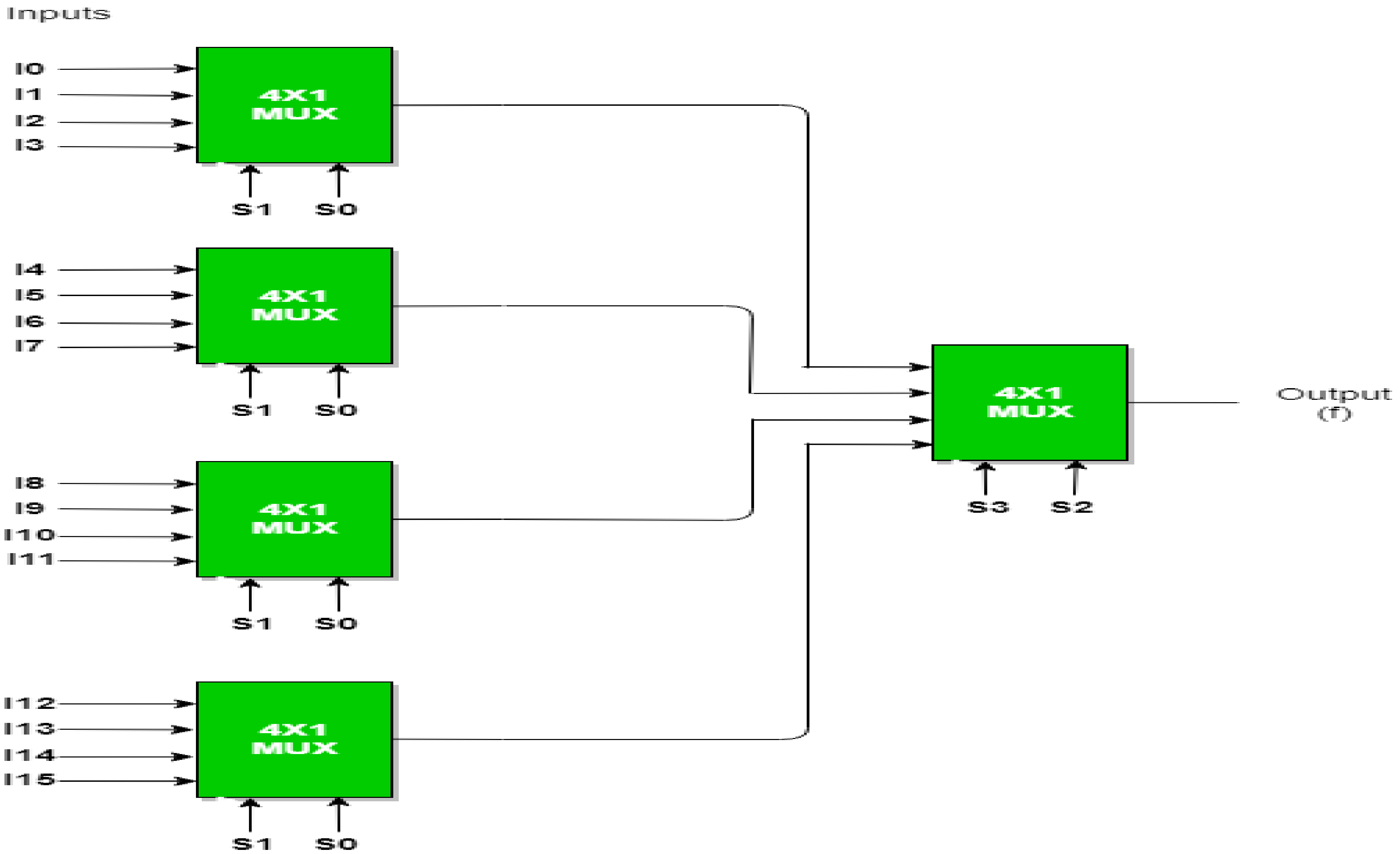
$$n = 8$$

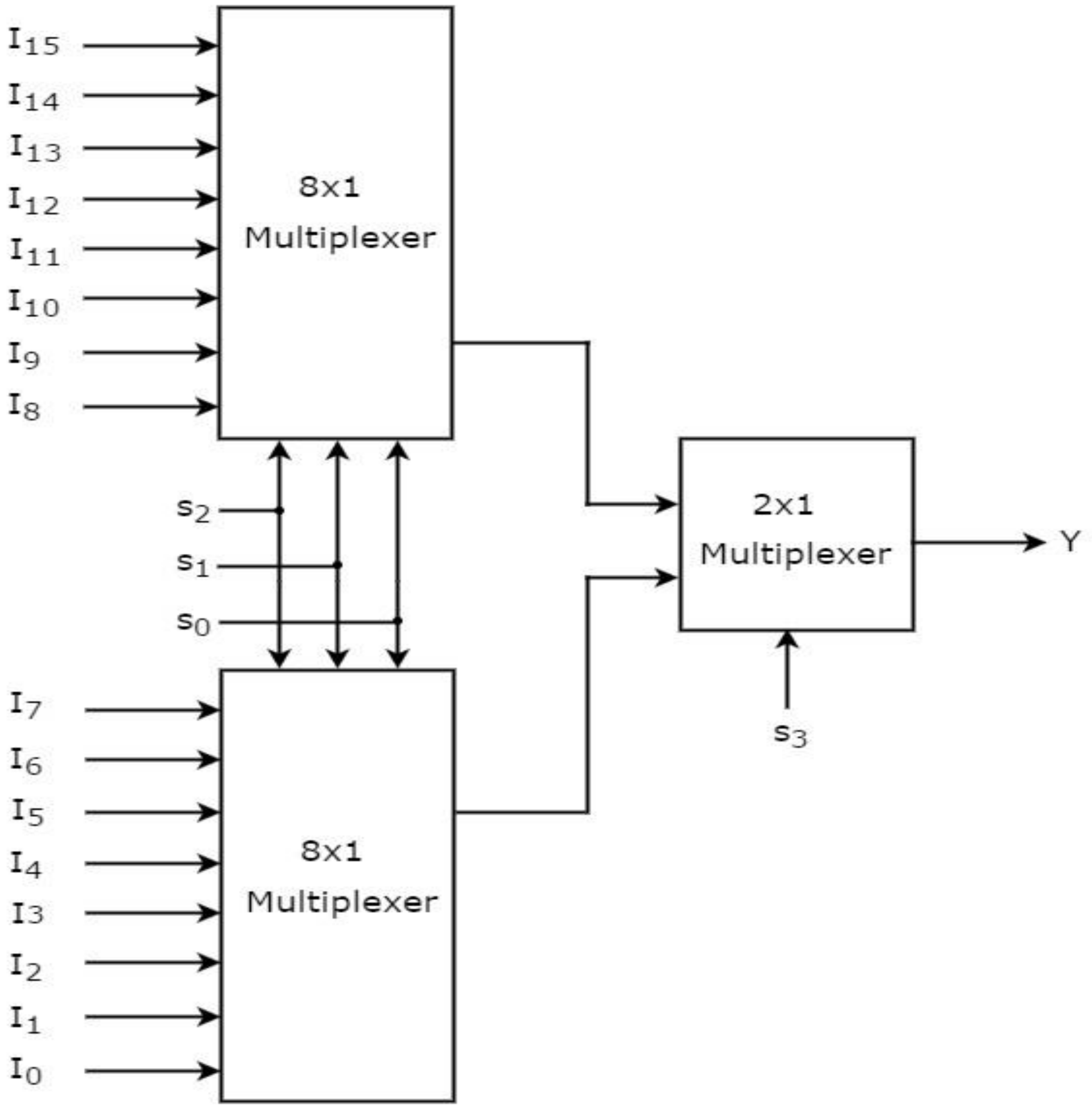
$$8/4 = 2$$

$$2/4 = 0.5$$

$$= 2.5$$

16:1 MUX using 4:1 MUX

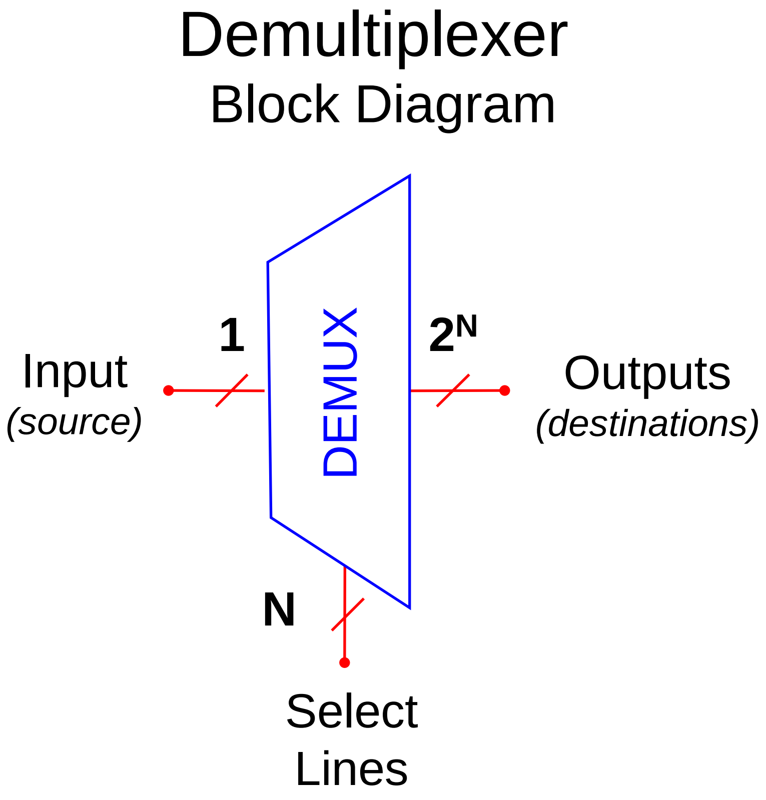




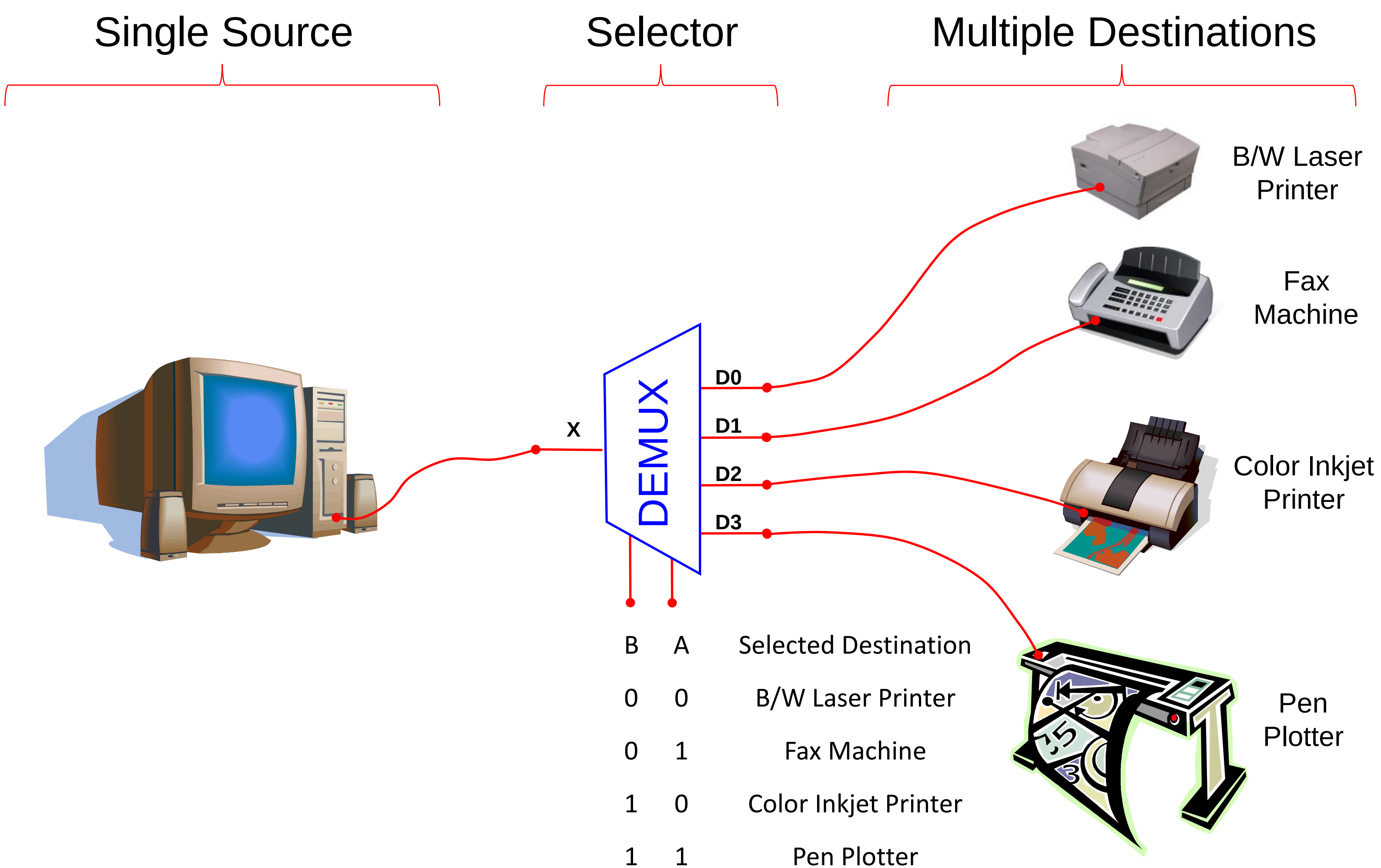
16:1 MUX using 8:1 MUX

What is a Demultiplexer (DEMUX)?

- A DEMUX is a digital switch with a single input (source) and a multiple outputs (destinations).
- The select lines determine which output the input is connected to.
- DEMUX Types
  - → 1-to-2 (1 select line)
  - → 1-to-4 (2 select lines)
  - → 1-to-8 (3 select lines)
  - → 1-to-16 (4 select lines)

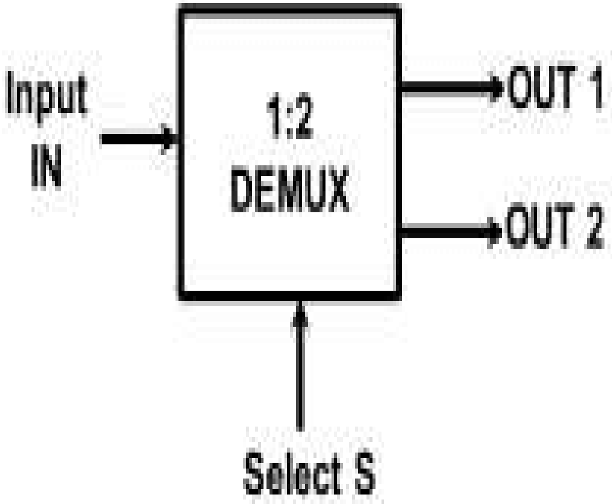


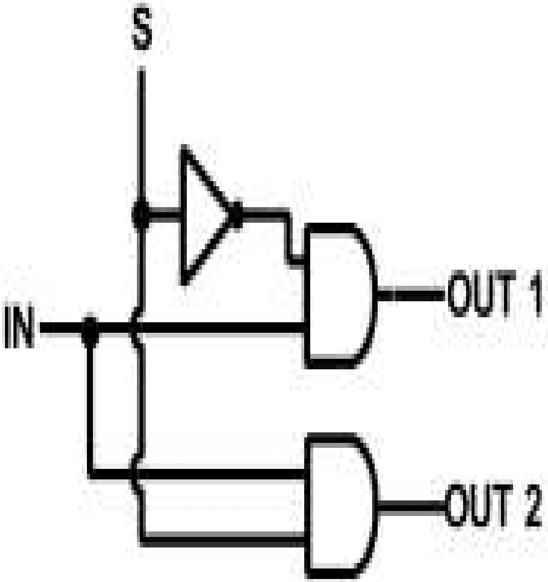
Typical Application of a DEMUX



B	A	Selected Destination
0	0	B/W Laser Printer
0	1	Fax Machine
1	0	Color Inkjet Printer
1	1	Pen Plotter

1: 2 Demux

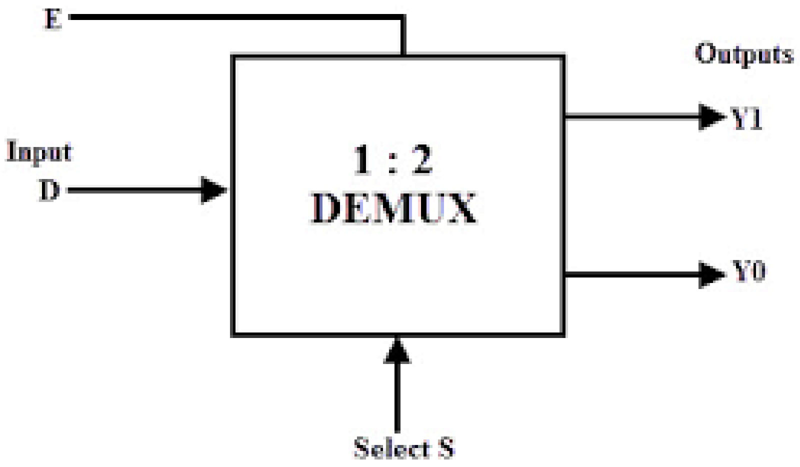


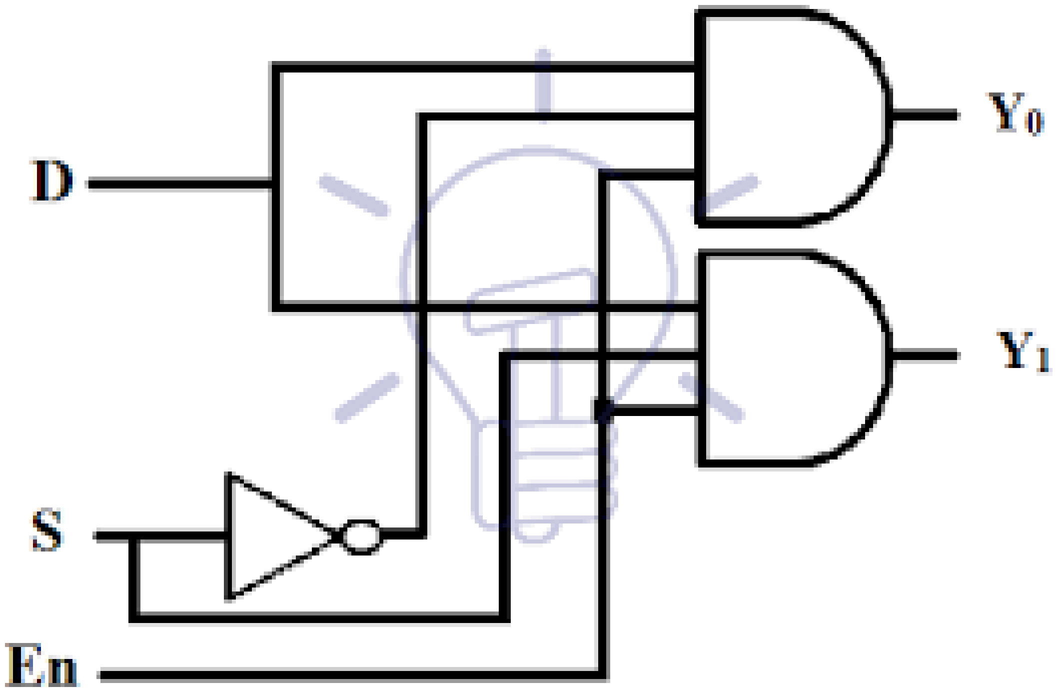


Select (S)	Input (IN)	Out 1	Out 2
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

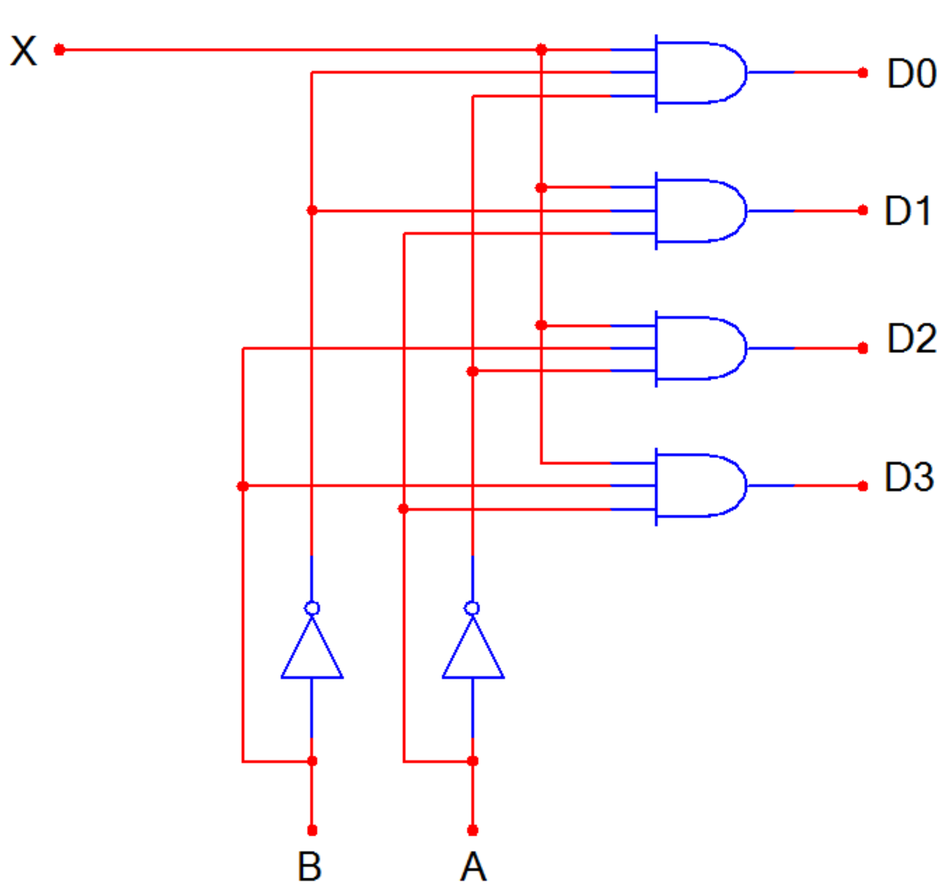
Enable	Select	Output	
E	S	Y0	Y1
0	x	0	0
1	0	0	D_{in}
1	1	D_{in}	0

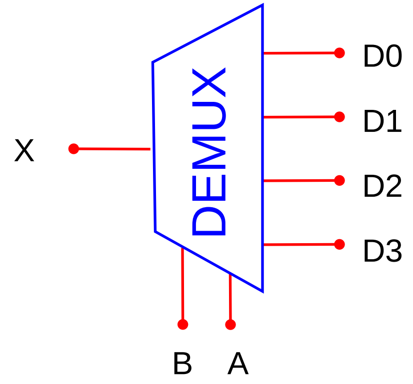
x = Don't care





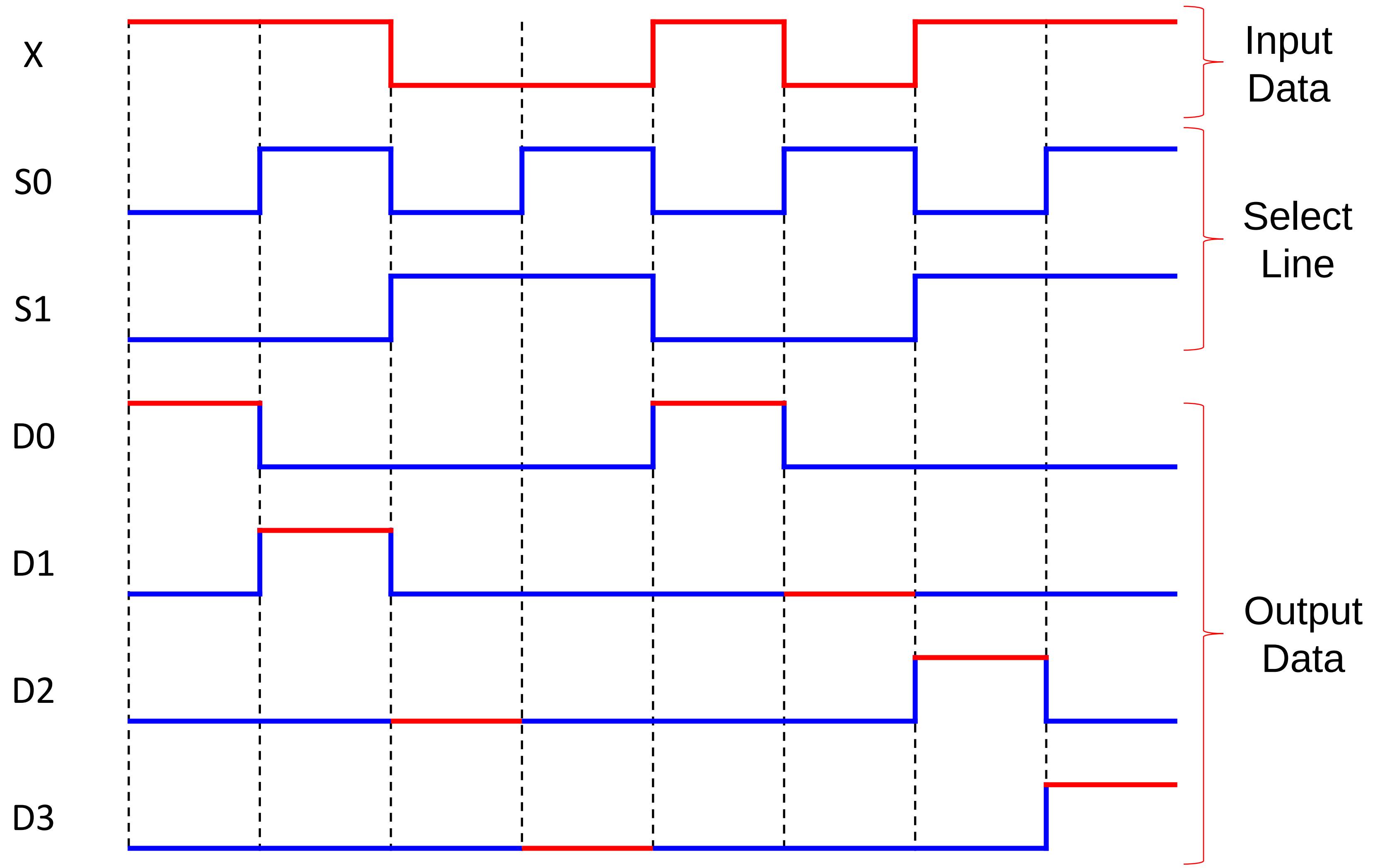
1-to-4 Demultiplexer (DeMUX)



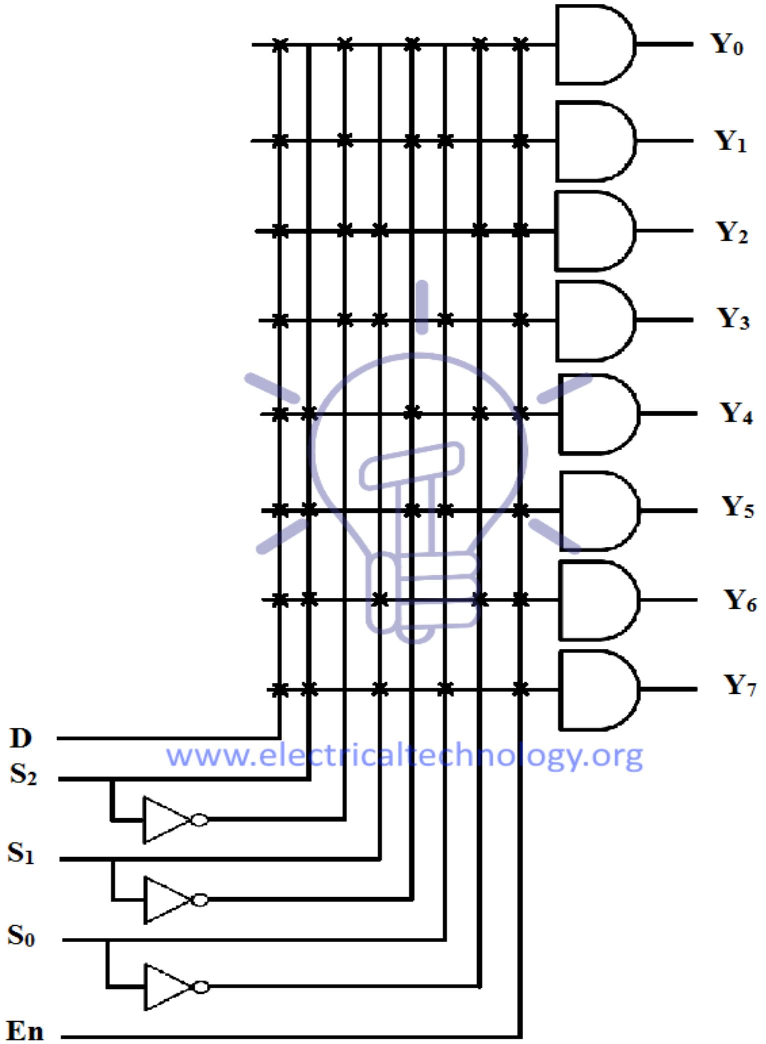


B	A	D0	D1	D2	D3
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X

1-to-4 De-Multiplexer Waveforms

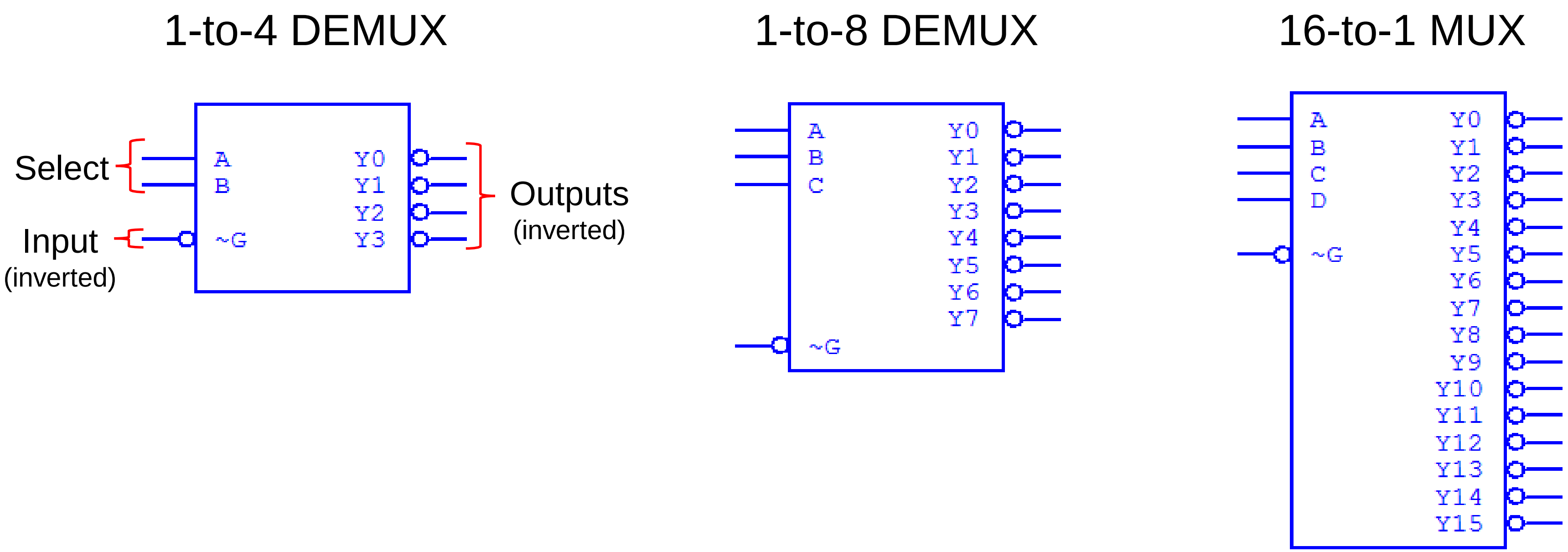


1:8 Demux with Enable Input



INPUT					OUTPUT							
En	D	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	X	X	X	X	0	0	0	0	0	0	0	0
1	X	0	0	0	D	0	0	0	0	0	0	0
1	X	0	0	1	0	D	0	0	0	0	0	0
1	X	0	1	0	0	0	D	0	0	0	0	0
1	X	0	1	1	0	0	0	D	0	0	0	0
1	X	1	0	0	0	0	0	0	D	0	0	0
1	X	1	0	1	0	0	0	0	0	D	0	0
1	X	1	1	0	0	0	0	0	0	0	D	0
1	X	1	1	1	0	0	0	0	0	0	0	D

Medium Scale Integration DEMUX

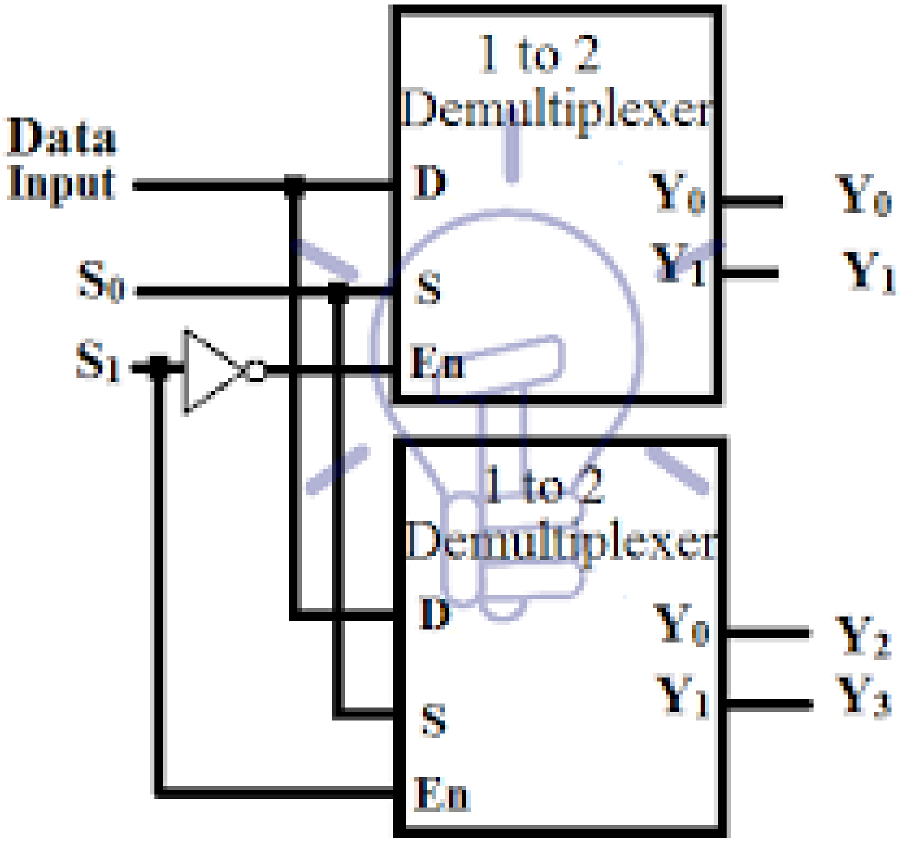


Note : Most Medium Scale Integrated (MSI) DEMUXs , like the three shown, have outputs that are inverted. This is done because it requires few logic gates to implement DEMUXs with inverted outputs rather than no-inverted outputs.

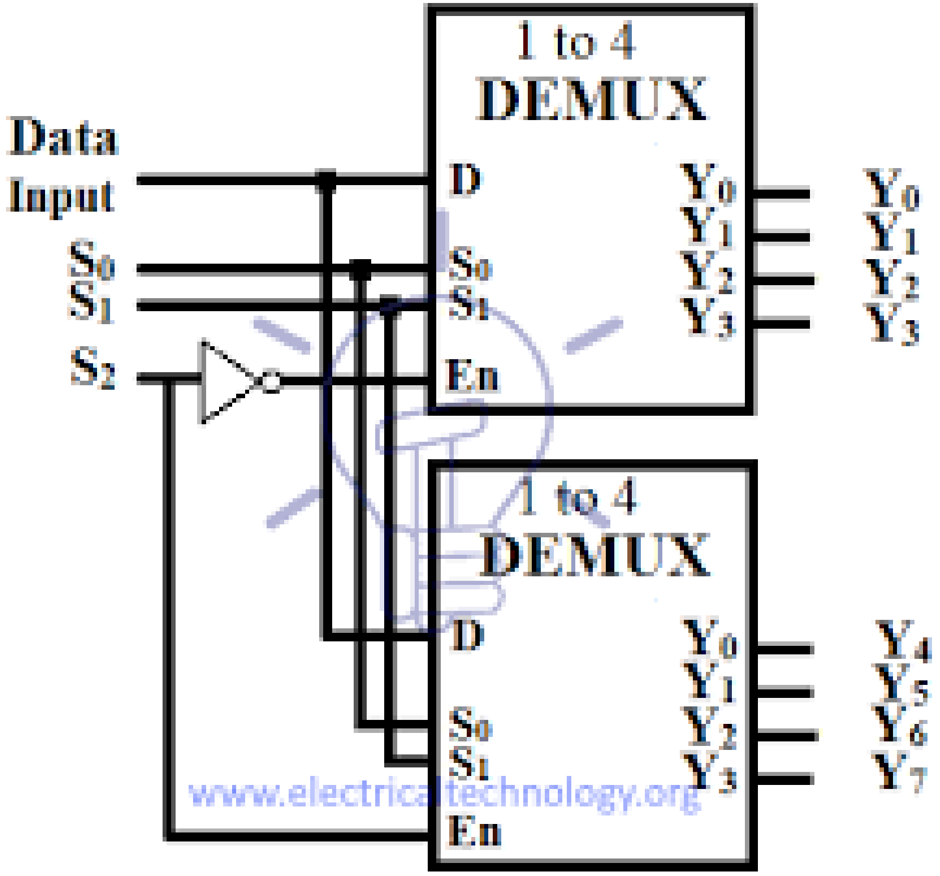
Demultiplxer Tree 1:4 using 1:2 Demux

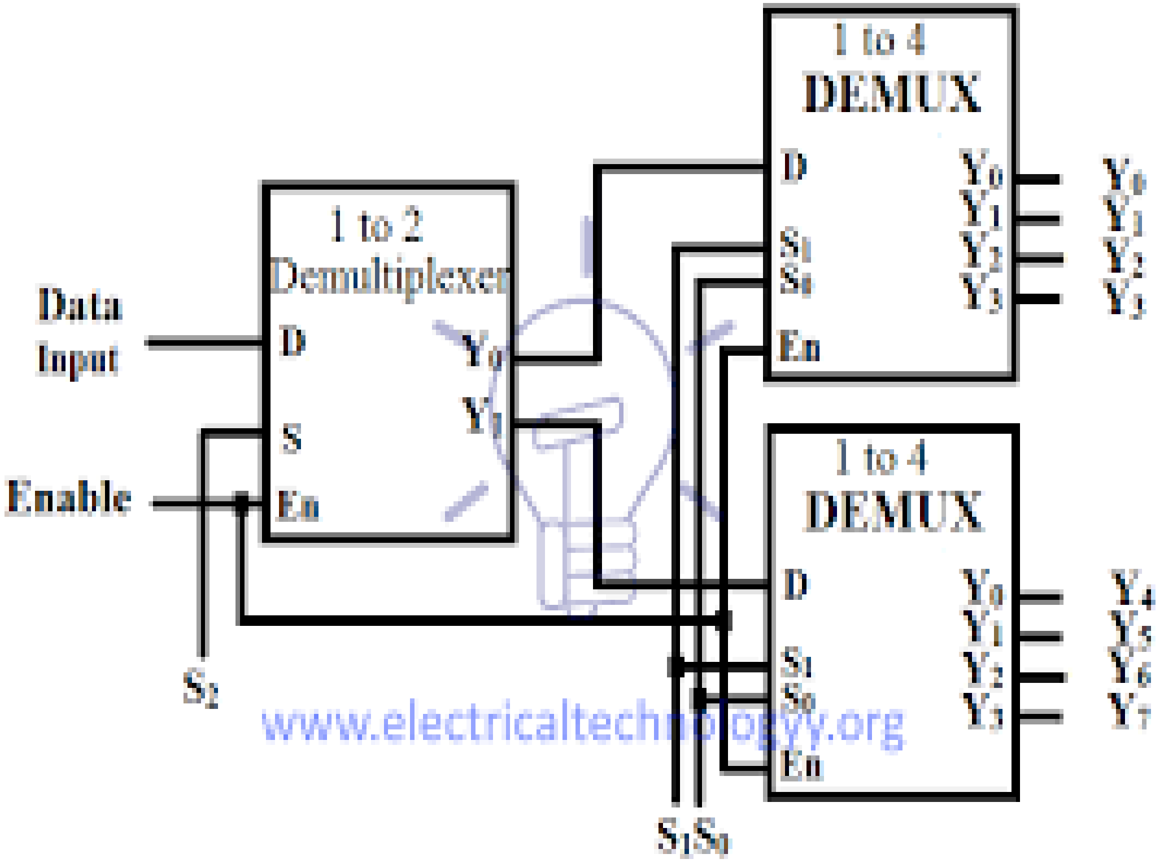
Truth Table

S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	D_{in}
0	1	0	0	D_{in}	0
1	0	0	D_{in}	0	0
1	1	D_{in}	0	0	0

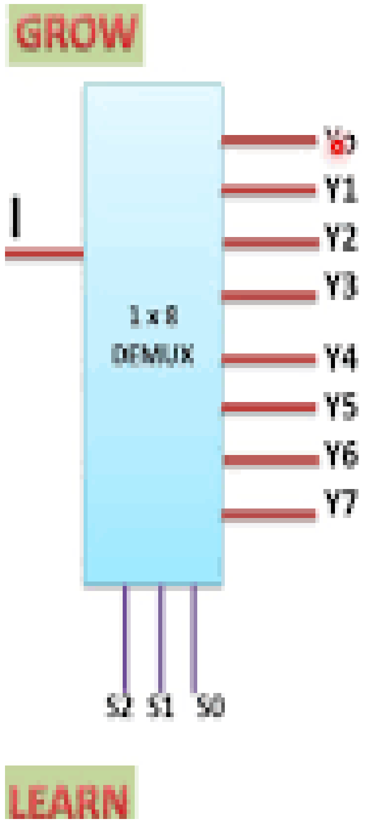


1:8 DeMUX Using 1:4



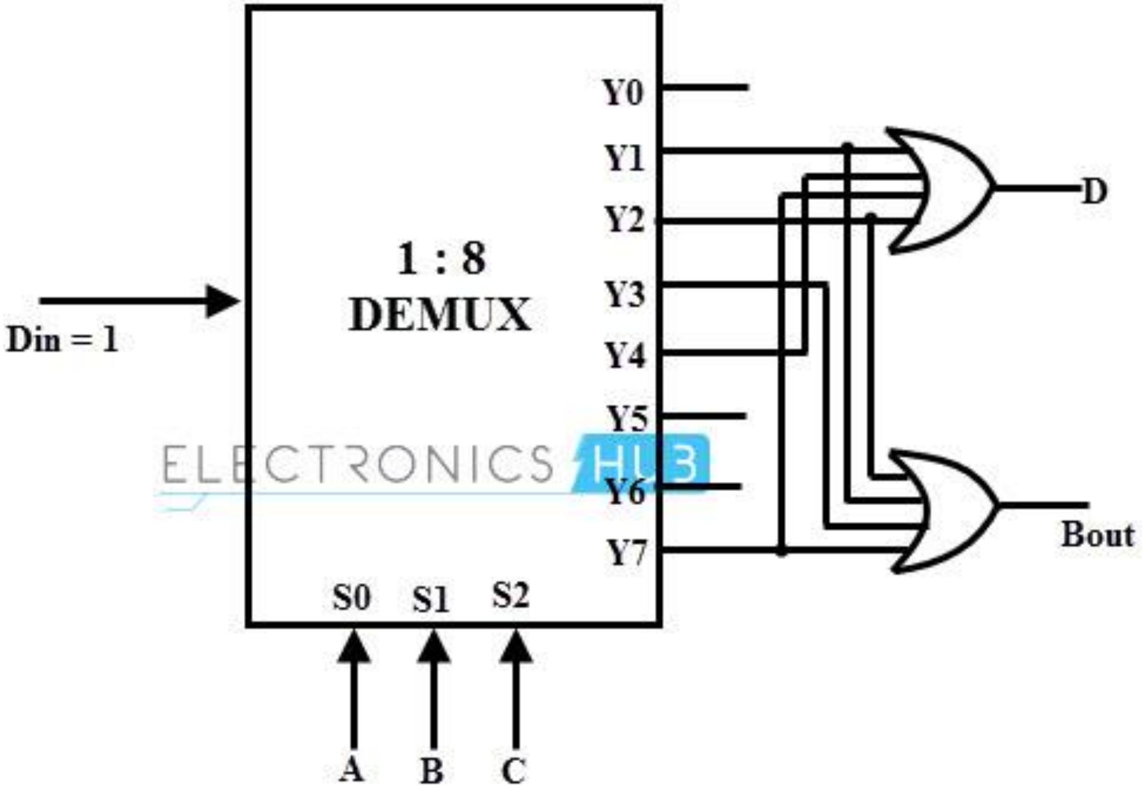


Full Adder using 1:8 De MUX



$D(x,y,z) = \Sigma m(1,2,4,7)$

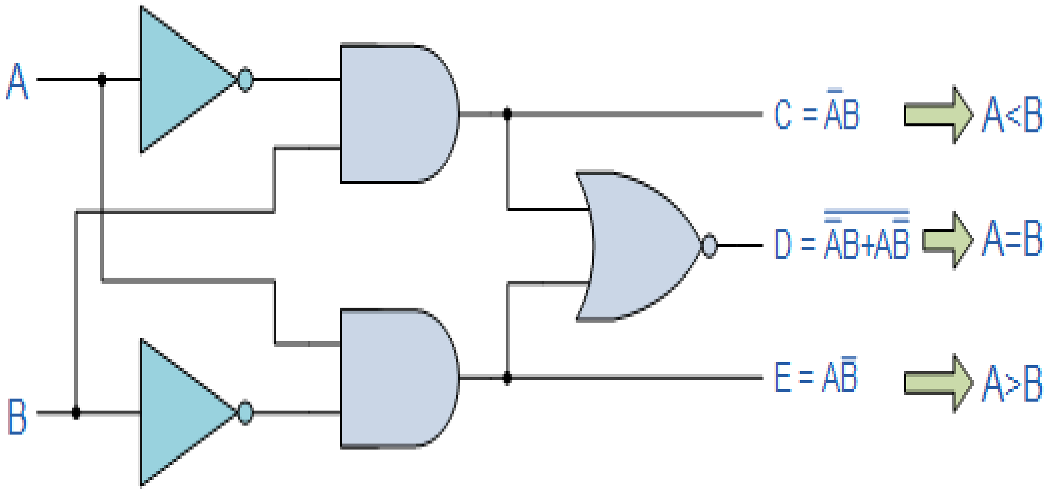
$B(x,y,z) = \Sigma m(1,2,3,7)$



Draw 1:64 demultiplexer using 1:16 demux and 1: 4 demux

Comparator

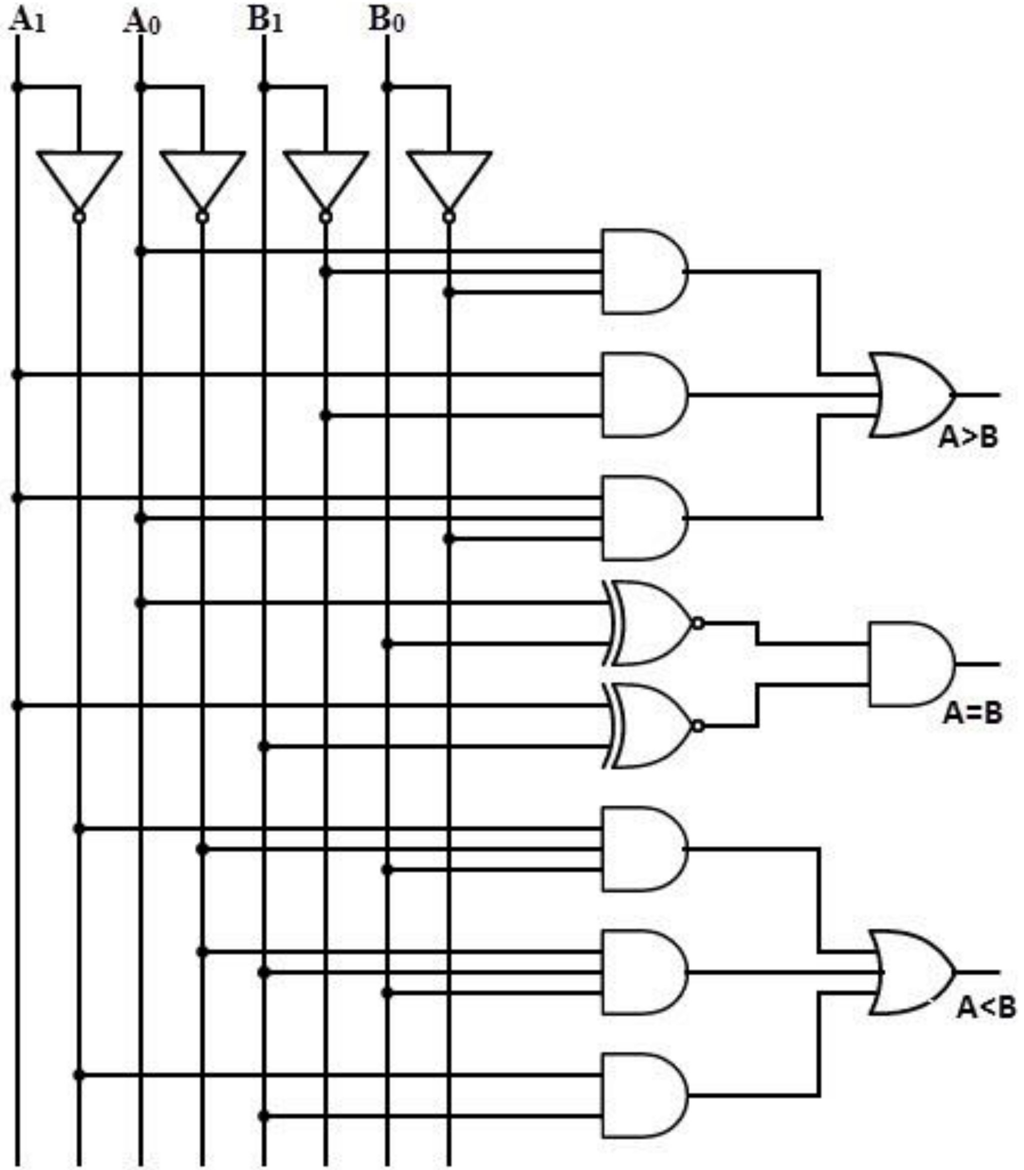
A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



2-Bit Magnitude Comparator

Table 1. Truth Table of 2-Bit Magnitude Comparator

INPUT				OUTPUT		
A1	A0	B1	B0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0



f(A>B)

A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

We get the equation as f(A>B)

$$= A_1\overline{B_1} + A_0\overline{B_1}\,\overline{B_0} + A_1A_0\overline{B_0}$$

K-Map for A<B:

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

K-Map for A=B:

A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

For A<B

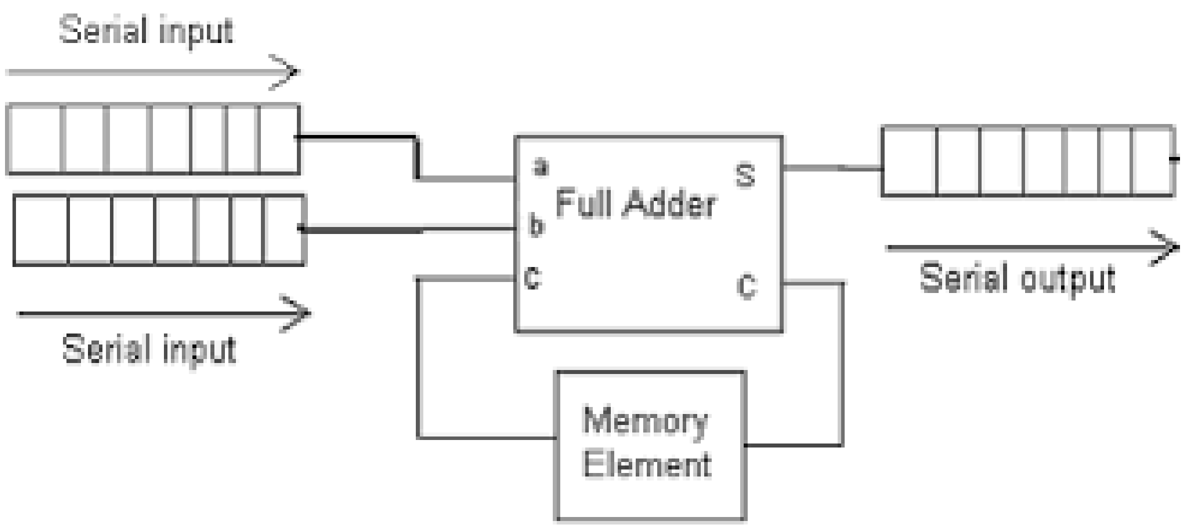
$$Y_1 = \overline{A_1}\,\overline{A_0}\,B_0 + \overline{A_1}\,B_1 + \overline{A_0}\,B_1\,B_0$$

For A=B

$$Y_2 = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}\,A_0\,\overline{B_1}\,B_0 + A_1A_0B_1B_0 + A_1\,\overline{A_0}\,B_1\,\overline{B_0}$$

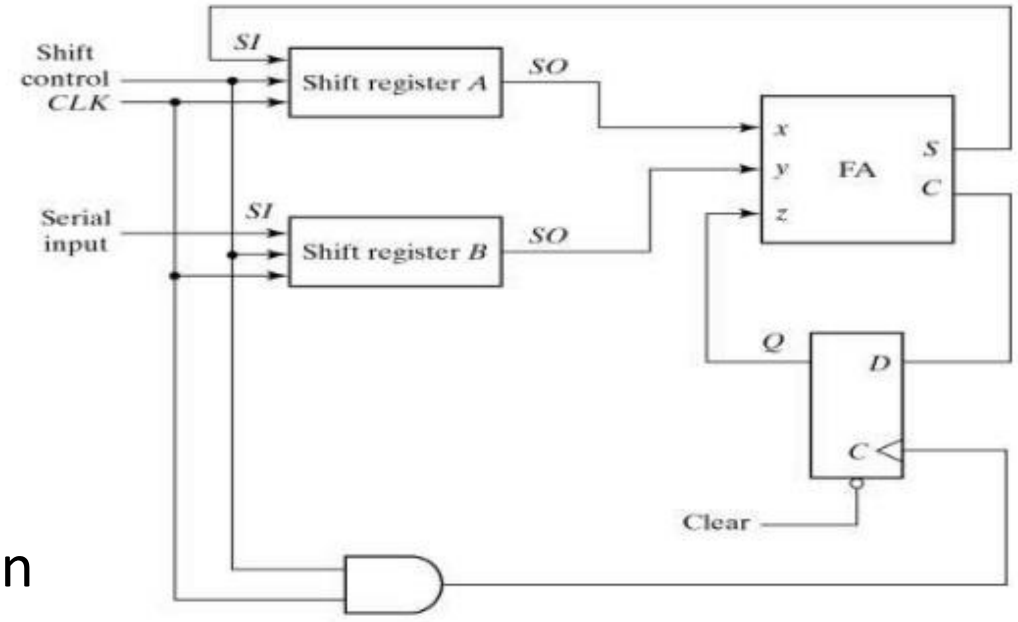
Binary Serial Adder

Let Register-1 hold the first number and register 2 holds the other no.
The D FF is cleared initially so Q=0 and Cin= 0.
The serial o/p (SO) of the two registers will provide the LSBs of the two number.
They will act bits a and b for full adder.
The FA will add these bits and produce sum and carry out Co.
Thus addition of LSB is complete.
Now a clock pulse is applied to both the shift registers.
Hence the two numbers are right shifted by one bit each.



The clock pulse gets applied to the D FF also and Q= Cin = Cout =0.
The adder adds the two bits available at the SO outputs of the two registers

Fig: Serial Adder



Serial Adder

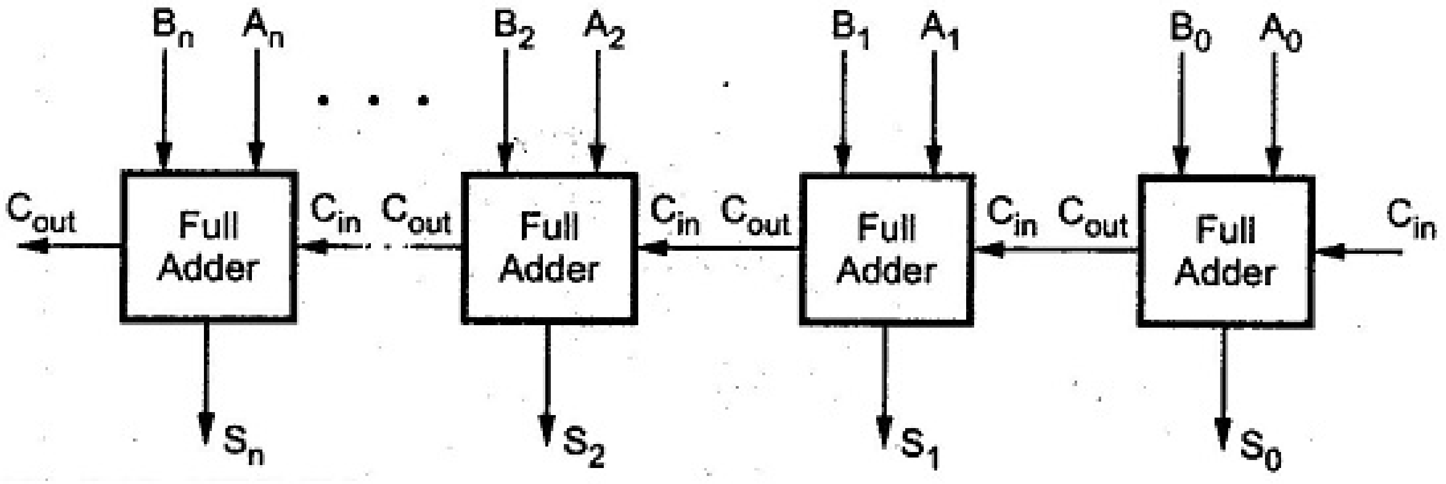
Advantages

The addition of a pair of bits only takes place at any instant of time
(n-1) number of clock cycles are required to complete the addition
The process of addition continues until the shift right control is disabled
Sum is available in the serial form

Drawback

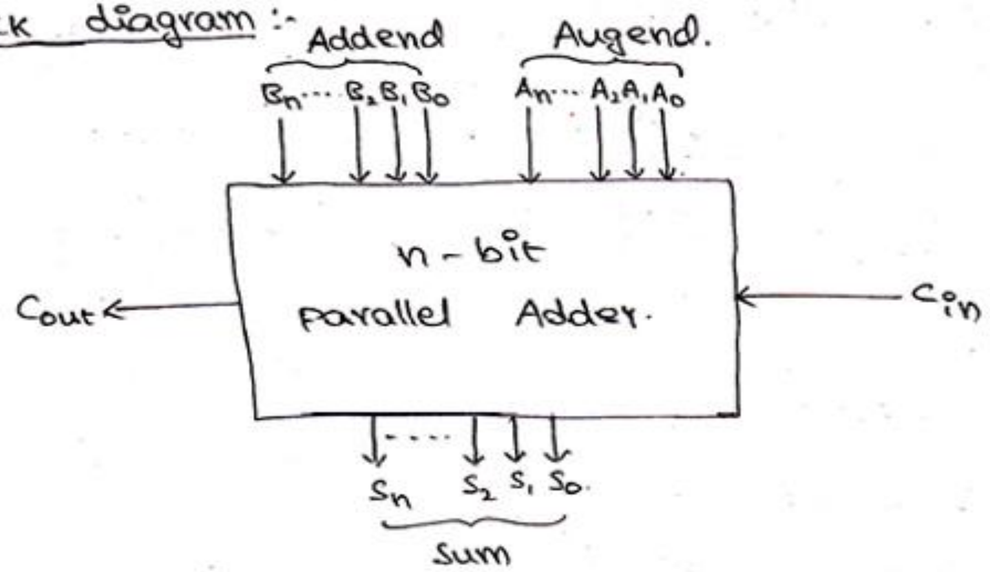
i) More time is required to complete the addition
ii) A complicated ckt is required
iii) The ckt contains more no of components
iv) The Sum and Carry are available in the serial form hence result can’t be observed at once.

Binary Parallel Adder



N-Bit Parallel Adder

Block diagram :-



Cascading of Adders

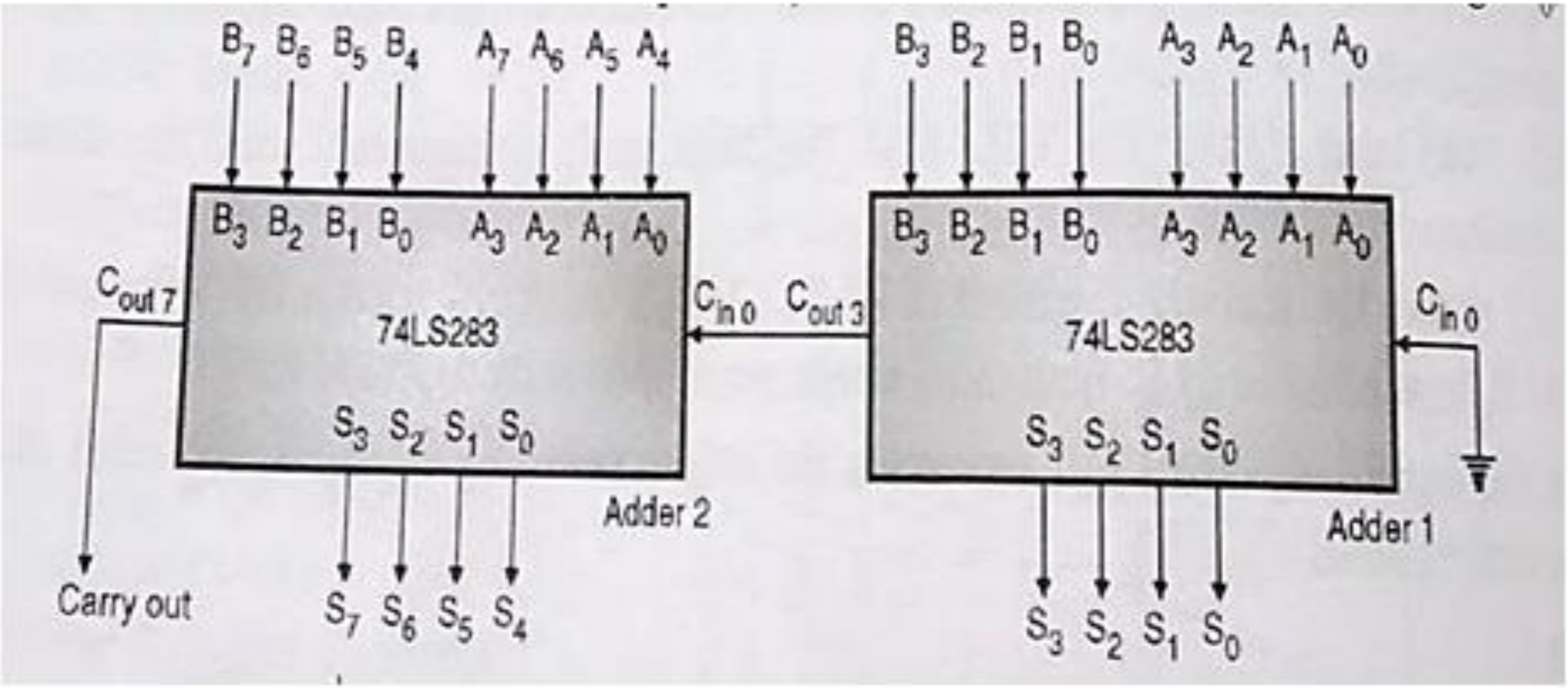
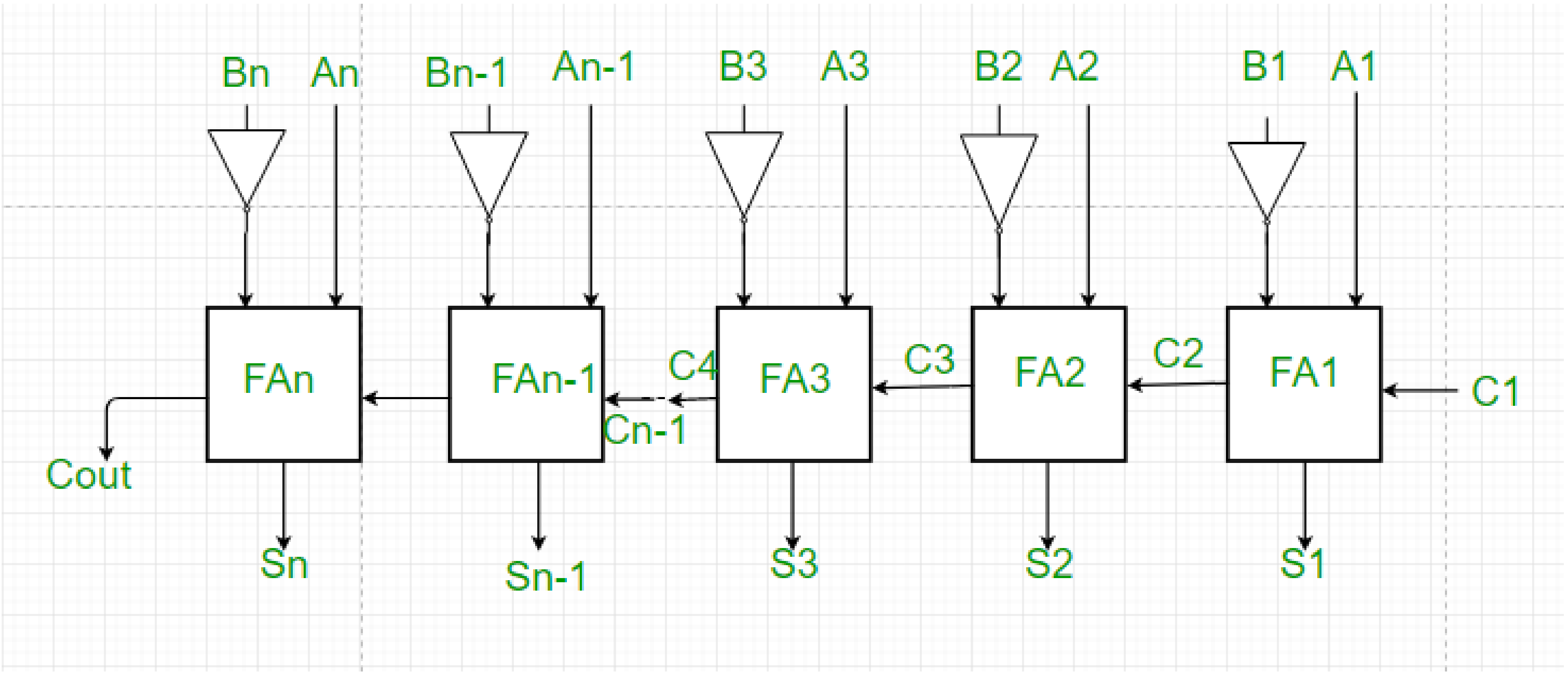


Fig2. Cascading of two IC 7483s

Binary Parallel Subtractor



Propagation delay in Parallel adder