ALTS

Anantha Lakshmi

INSTITUTE OF TECHNOLOGY AND SCIENCES

(Approved by A.I.C.T.E, New Delhi, Accredited by NAAC and Affiliated to J.N.T.University Ananthapuramu.)

Near SK University, Itikalapalli-515 721. (V), Ananthapuramu (Dist.)

Website : www.alits.ac.in

LECTURENOTES

ON

DIGITAL ELECTRONICS

B.TechIII-Isemester (R20)

ELECTRONICSANDCOMMUNICATIONENGINEERING

**ANANTHALAKSHMIINSTITUTEOF
TECHNOLOGY AND SCIENCES**

ANANTAPUR

UNIT-I

DIGITAL FUNDAMENTALS

NUMBERSYSTEMS&CODES

- Philosophyofnumbersystems
- Complementrepresentationofnegativenumbers
- Binaryarithmetic
- Binarycodes
- Errordetecting&errorcorrectingcodes:Hammingcodes

HISTORYOFTHENUMERALSYSTEMS:

A **numeral system** (or **system of numeration**) is a linguistic system and mathematical notation for representing numbers of a given set bysymbols in a consistent manner. For example, It allows the numeral "11" to be interpreted as the binary numeral for *three*, the decimal numeral for *eleven*, or other numbers in different bases. Ideally, a numeral system will:

- Representausefulsetofnumbers(e.g.allwholenumbers,integers,orrealnumbers)
- Giveeverynumberrepresentedauniquerepresentation(oratleast astandardrepresentation)
- Reflect the algebraic and arithmetic structure of the numbers.

For example, the usual decimal representation of whole numbers gives every whole number a unique representationasafinitesequenceofdigits,withtheoperationsofarithmetic(addition,subtraction, multiplicationanddivision)beingpresentasthestandardalgorithmsofarithmetic.However,when decimal representation is used for the rational or real numbers, the representation is no longer unique: many rational numbers have two numerals, a standard one that terminates, such as 2.31, andanotherthatrecurs,suchas2.309999999Numeralswhichterminatehavenonon-zero digits after a given

position. For example,numerals like **2.31 and 2.310 are taken** to be the same, except in the experimental sciences, where greater precision is denoted by the trailing zero.

The most commonly used system of numerals is known as Hindu-Arabic numerals.Great Indian mathematicians Aryabhatta of Kusumapura (5th Century) developed the place value notation. Brahmagupta(6thCentury) introducedthesymbolzero.

BINARY

Theancient Indian writer Pingala developed advanced mathematical concepts fordescribing prosody, and in doing so presented the first known description of a binary numeral system.A full setof8trigramsand64hexagrams,analogoustothe3-bitand6-bit binarynumerals,wereknown totheancientChineseintheclassictext*IChing*.Anarrangementofthehexagramsofthe*IChing*, ordered accordingtothevaluesofthecorrespondingbinarynumbers(from 0to63),andamethod for generating thesame, was developed by the Chinese scholar and philosopher Shao Yong in the 11th century.

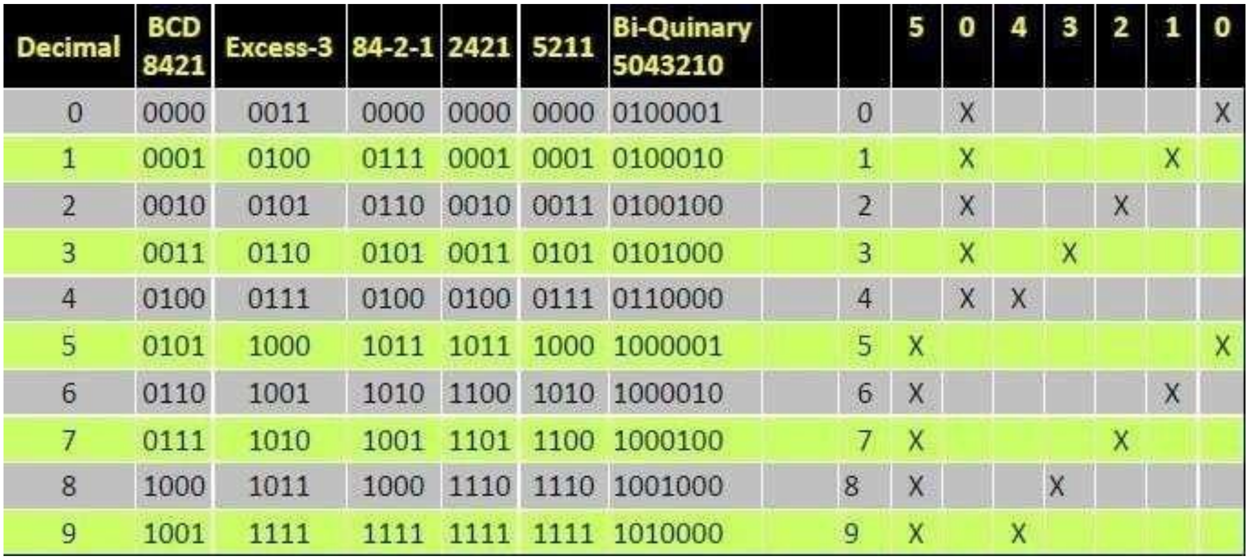
In 1854, British mathematician George Boole published a landmark paper detailing an algebraic systemoflogicthatwouldbecomeknownasBooleanalgebra.Hislogicalcalculuswastobecome instrumental in the design of digital electronic circuitry. In 1937, Claude Shannon produced his master's thesis at MIT that implemented Boolean algebra and binary arithmetic using electronic relaysandswitchesforthefirsttimeinhistory.Entitled*ASymbolicAnalysisofRelayandSwitching Circuits*, Shannon's thesis essentially founded practical digital circuit design.

Binarycodes

Binarycodesarecodeswhicharerepresentedinbinarysystemwithmodificationfromtheoriginal ones.

- WeightedBinarycodes
- NonWeightedCodes

Weighted binarycodes are those which obeythe positional weightingprinciples, each position of the number represents a specific weight. The binary counting sequence is an example.



Decimal	BCD	Excess-3	84-2-1	2421	5211	Bi-Quinary			5	0	4	3	2	1	0
	8421					5043210									
0	0000	0011	0000	0000	0000	0100001		0		X					X
1	0001	0100	0111	0001	0001	0100010		1		X				X	
2	0010	0101	0110	0010	0011	0100100		2		X			X		
3	0011	0110	0101	0011	0101	0101000		3		X		X			
4	0100	0111	0100	0100	0111	0110000		4		X	X				
5	0101	1000	1011	1011	1000	1000001		5	X						X
6	0110	1001	1010	1100	1010	1000010		6	X					X	
7	0111	1010	1001	1101	1100	1000100		7	X				X		
8	1000	1011	1000	1110	1110	1001000		8	X			X			
9	1001	1111	1111	1111	1111	1010000		9	X		X				

ReflectiveCode

Acodeissaidtobereflectivewhen codefor9iscomplementforthecodefor0,andsoisfor8 and 1 codes, 7 and 2, 6 and 3, 5 and 4. Codes 2421, 5211, and excess-3 are reflective, whereas the 8421 code is not.

SequentialCodes

Acodeissaidtobesequentialwhentwosubsequentcodes,seenasnumbersin binary

representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.

Nonweightedcodes

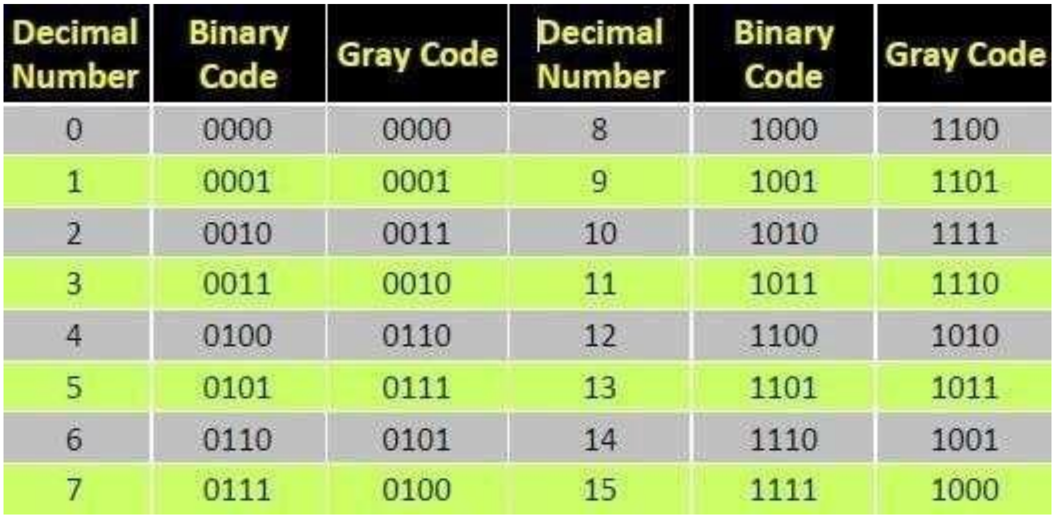
Non weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value. Ex: Excess-3 code

Excess-3Code

Excess-3 is a non weighted code used to express decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011(3).

GrayCode

The gray code belongs to a class of codes called minimum change codes, in which only one bit in the code changes when moving from one code to the next. The Gray code is non-weighted code, as the position of bit does not contain any weight. The gray code is a reflective digital code which has thespecialpropertythatanytwosubsequentnumberscodesdifferbyonlyonebit.Thisisalsocalled a unitdistance code. In digital Gray code has got a special place.



Decimal Number	Binary Code	Gray Code	Decimal Number	Binary Code	Gray Code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

BinarytoGray Conversion

- GrayCodeMSBisbinarycodeMSB.
- GrayCodeMSB-1istheXORofbinarycodeMSBandMSB-1.
- MSB-2bitofgraycodeisXORofMSB-1andMSB-2bitofbinary code.
- MSB-NbitofgraycodeisXORofMSB-N-1and MSB-Nbitofbinarycode.

Errordetectioncodes

1) Paritybits

A**parity bit** isabitthatisaddedtoa groupofsourcebitstoensurethatthenumberofsetbits(i.e., bitswithvalue1)intheoutcomeisevenorodd.Itisaverysimpleschemethatcanbeusedtodetect single or any other odd number (i.e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous.

2) Checksums

A **checksum** of a message is a modular arithmetic sum of message code words of a fixed word length (e.g.,byte values). The sum may be negated by means of a one's-complement prior to transmission to detect errorsresulting in all-zero messages.Checksum schemes include parity bits, check digits, and longitudinal redundancy checks. Some checksum schemes, such as the Luhn algorithm and the Verhoeff algorithm, are specifically designed to detect errorscommonly introduced by humans in writing down or remembering identification numbers.

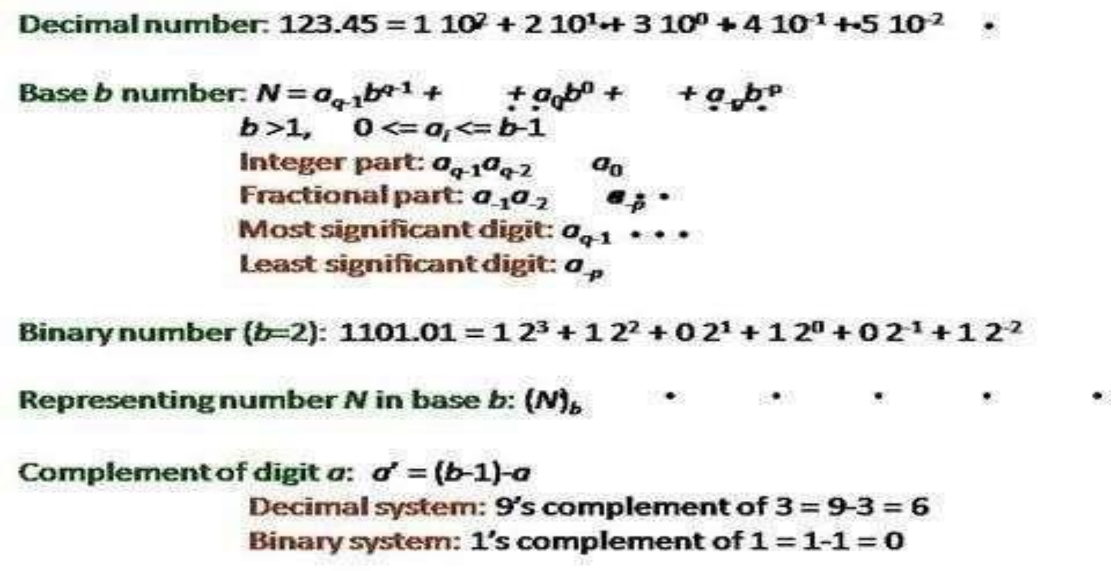
3) Cyclicredundancychecks(CRCs)

A **cyclic redundancy check (CRC)** is a single-burst-error-detecting cyclic code and non-secure hash function designed to detect accidental changes to digital data in computer networks. It is characterizedbyspecificationofaso-called*generatorpolynomial*,whichisusedasthedivisorina polynomial long division over a finite field, taking the input data as the dividend, and where the remainderbecomestheresult.Cycliccodeshavefavorablepropertiesinthattheyarewellsuitedfor detecting burst errors. CRCs are particularly easy to implement in hardware, and are therefore commonly used in digital networks and storage devices such as hard disk drives.Even parity is a specialcaseofacyclicredundancycheck,wherethesingle-bitCRCisgeneratedbythedivisor$x$+1.

NUMBERBASECONVERSIONS

Anynumberinonebasesystem canbeconvertedintoanotherbasesystemTypes

1) decimalto anybase
2) Anybasetodecimal
3) Anybaseto Anybase



Decimal number: $123.45 = 1\ 10^2 + 2\ 10^1 + 3\ 10^0 + 4\ 10^{-1} + 5\ 10^{-2}$

Base b number: $N = a_{q-1}b^{q-1} + \quad + a_0b^0 + \quad + a_{-p}b^{-p}$

$b > 1, \quad 0 <= a_i <= b-1$

Integer part: $a_{q-1}a_{q-2} \quad a_0$

Fractional part: $a_{-1}a_{-2} \quad a_{-p}$

Most significant digit: a_{q-1}

Least significant digit: a_{-p}

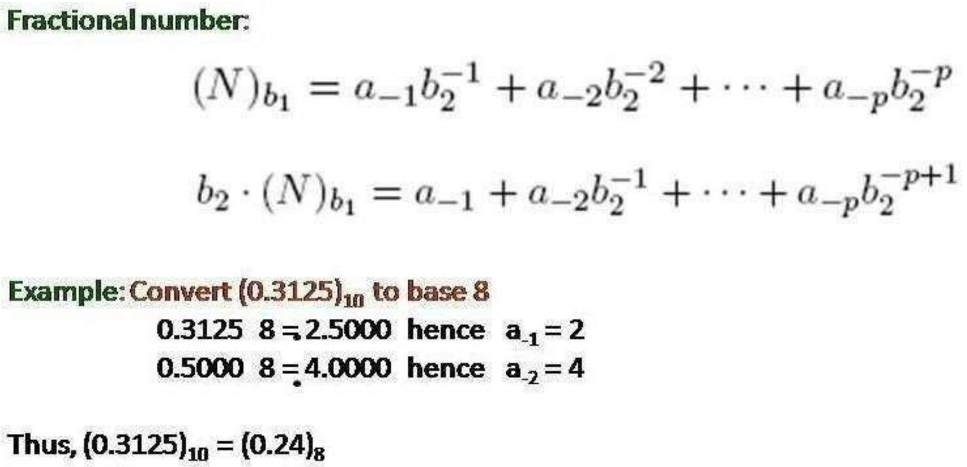
Binary number (b=2): $1101.01 = 1\ 2^3 + 1\ 2^2 + 0\ 2^1 + 1\ 2^0 + 0\ 2^{-1} + 1\ 2^{-2}$

Representing number N in base b: $(N)_b$

Complement of digit a: $a' = (b-1)-a$

Decimal system: 9's complement of 3 = 9-3 = 6

Binary system: 1's complement of 1 = 1-1 = 0



Fractional number:

$$(N)_{b_1} = a_{-1}b_2^{-1} + a_{-2}b_2^{-2} + \cdots + a_{-p}b_2^{-p}$$

$$b_2 \cdot (N)_{b_1} = a_{-1} + a_{-2}b_2^{-1} + \cdots + a_{-p}b_2^{-p+1}$$

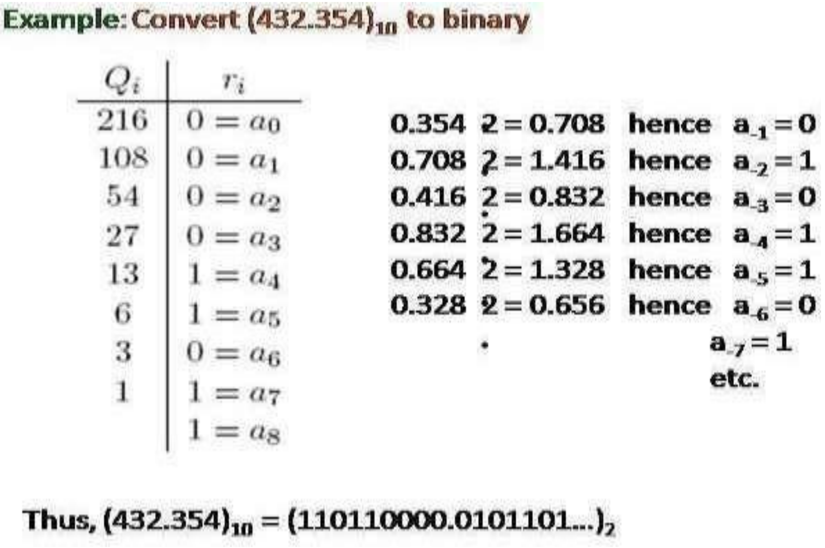
Example: Convert $(0.3125)_{10}$ to base 8

0.3125 8 = 2.5000 hence $a_{-1} = 2$

0.5000 8 = 4.0000 hence $a_{-2} = 4$

Thus, $(0.3125)_{10} = (0.24)_8$

DecimaltoBinary



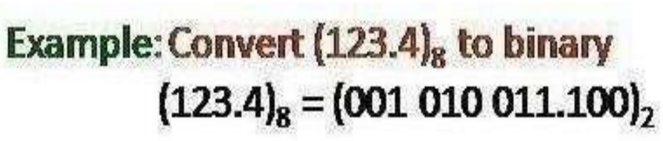
Example: Convert $(432.354)_{10}$ to binary

Q_i	r_i
216	$0 = a_0$
108	$0 = a_1$
54	$0 = a_2$
27	$0 = a_3$
13	$1 = a_4$
6	$1 = a_5$
3	$0 = a_6$
1	$1 = a_7$
	$1 = a_8$

0.354 2 = 0.708 hence $a_{-1} = 0$
0.708 2 = 1.416 hence $a_{-2} = 1$
0.416 2 = 0.832 hence $a_{-3} = 0$
0.832 2 = 1.664 hence $a_{-4} = 1$
0.664 2 = 1.328 hence $a_{-5} = 1$
0.328 2 = 0.656 hence $a_{-6} = 0$
. $a_{-7} = 1$
etc.

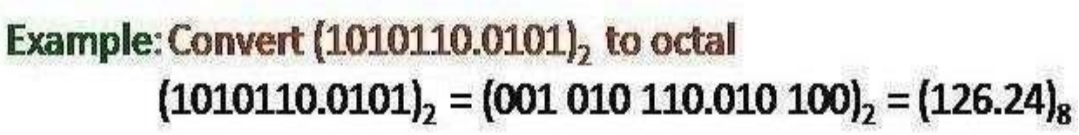
Thus, $(432.354)_{10} = (110110000.0101101...)_2$

OctalToBinary



Example: Convert $(123.4)_8$ to binary

$(123.4)_8 = (001\ 010\ 011.100)_2$



Example: Convert $(1010110.0101)_2$ to octal

$(1010110.0101)_2 = (001\ 010\ 110.010\ 100)_2 = (126.24)_8$

ErrorDetectionandCorrectionCodes

- Nocommunication channelorstoragedeviceiscompletelyerror-free
- Asthenumberofbitsperareaorthetransmissionrateincreases,more errors occur. • Impossible to detect or correct 100% of the errors

Hamming Codes

1. Oneofthemosteffectivecodesforerror-recovery
2. Usedinsituationswhererandomerrorsarelikelytooccur
3. Errordetectionandcorrectionincreasesinproportiontothenumberofparity bits (error- checking bits) added to the end of the information bits

   codeword=informationbits+paritybits

   Hammingdistance:thenumberofbitpositionsinwhichtwocodewordsdiffer.

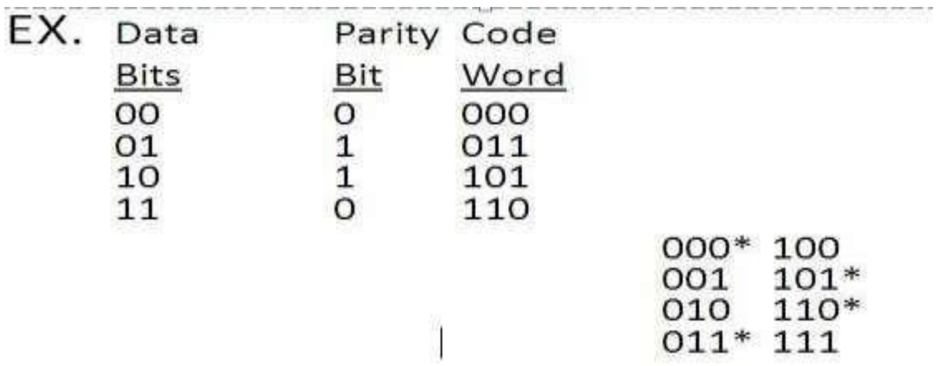
   10001001

   10110001

   * * *

   MinimumHammingdistanceorD(min):determinesitserrordetectingandcorrecting capability.

4. HammingcodescanalwaysdetectD(min)–1errors,butcanonlycorrecthalfofthoseerrors.



EX.

Data Bits	Parity Bit	Code Word
00	0	000
01	1	011
10	1	101
11	0	110

000*	100
001	101*
010	110*
011*	111

5. Singleparitybit canonlydetecterror, notcorrectit

6. Error-correctingcodesrequiremorethanasingle

   paritybit EX. 0 0 0 0 0

   0 1 0 11

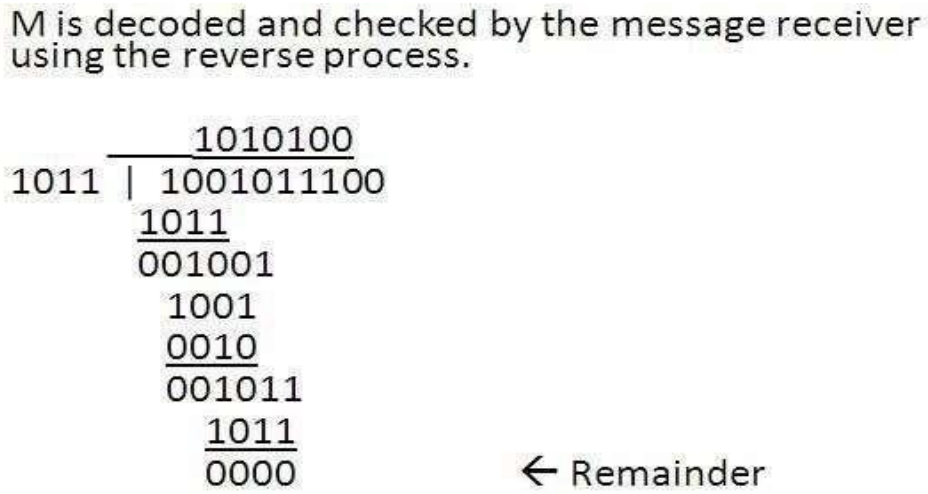
   1 0 1 10

1 1 1 0 1

MinimumHammingdistance=3

Candetectupto2errorsandcorrect1error
Cyclic Redundancy Check

1. LettheinformationbyteF= 1001011
2. Thesender andreceiveragreeonanarbitrarybinarypatternP.LetP=1011.
3. ShiftFto theleftby1lessthanthenumberofbits inP.Now,F=1001011000.
4. LetFbethedividendand P bethedivisor.Perform“modulo2division”.
5. Afterperformingthedivision,weignorethequotient.Wegot100forthe remainder, which becomes the actual CRC checksum.
6. AddtheremaindertoF,givingthe messageM: 1001011 + 100 =



M is decoded and checked by the message receiver using the reverse process.

```
            1010100
1011 | 1001011100
          1011
          001001
            1001
            0010
            001011
              1011
              0000          ← Remainder
```

1001011100 =M

BOOLEANALGEBRAANDTHEOREMS

- **FundamentalpostulatesofBooleanalgebra**
- **Basictheoremsandproperties**
- **Switchingfunctions**
- **CanonicalandStandardforms**
- **Algebraicsimplificationdigitallogicgates,propertiesofXORgates**
- **Universalgates**
- **MultilevelNAND/NORrealizations**

BooleanAlgebra: Booleanalgebra,likeanyotherdeductivemathematicalsystem,maybedefined with aset of elements, a set of operators, and a number of unproved axioms or postulates. A *set* of elementsisanycollectionofobjectshavingacommonproperty. If**S** isasetand***x***and***y***arecertain objects,then**x** Î**S**denotesthat***x*** isamemberofthe set**S**,and ***y*** Ï**S** denotes that***y*** isnotan element of **S**. A set with adenumerable number of elements is specified by braces: **A** = {1,2,3,4}, *i.e.* the elements of set **A** are thenumbers 1, 2, 3, and 4. A *binary operator* defined on a set S of elements isarulethatassignstoeachpairofelementsfromSauniqueelementfromS._Example:In$a*b=c$,
wesaythat * is abinaryoperatorifit specifies aruleforfinding *c*from thepair(*a*,*b*)and also if *a*, *b*, *c* Î S.

CLOSURE: The Boolean system is *closed* with respect to a binary operator if for every pair of Boolean values,it produces a Boolean result. For example, logical AND is closed in the Boolean system because it accepts only Boolean operands and produces only Boolean results.

A set *S* is closed with respect to a binary operator if, for every pair of elements of *S*, the binary operator specifies a rule for obtaining a unique element of *S*.

For example, the set of natural numbers N = {1, 2, 3, 4, … 9} is closed with respect to the binary operator plus(+)bytheruleofarithmeticaddition,sinceforany*a*, *b* ÎN weobtain aunique *c* ÎN by the operation *a* + *b* = c.

ASSOCIATIVELAW:

Abinaryoperator*ona set *S*issaidtobeassociativewhenever$(x*y)*z = x*(y*z)$forall$x,y,$ z Î S, forall Boolean values x, y and z.

COMMUTATIVELAW:

Abinaryoperator *onaset *S*is saidtobecommutativewheneverx**y*=*y***x*forall*x*, *y*,*z*є S

IDENTITYELEMENT:

A set ***S*** is said to have an identityelement with respect to a binary operation * on S if there exists an element *e є* S with the property*e * x* = *x * e* = *x* for every*x* є S

BASICIDENTITIESOFBOOLEANALGEBRA

- *Postulate 1(Definition)*: A Boolean algebra is a closed algebraic system containing a set *K* of two or more elements and the two operators · and + which refer to logical AND and logical OR •*x + 0 =x*
- *x·0 =0*
- *x+1 =1*
- *x·1 =1*
- *x+x=x*
- *x· x=x*
- *x+x'=x*
- *x· x'=0*
- *x+y=y+x*
- *xy= yx*
- *x+(y+z)= (x+y)+z*
- *x(yz)=(xy)z*
- *x (y+z)=xy+ xz*
- *x+yz=(x +y)(x+z)*
- *(x+y)' =x'y'*
- *(xy)'=x'+y'*
- *(x')'=x*

DeMorgan'sTheorem

(a)$(a+b)'=a'b'$
(b) $(ab)' = a' + b'$

GeneralizedDeMorgan's

Theorem (a)$(a+b+\ldots z)'$

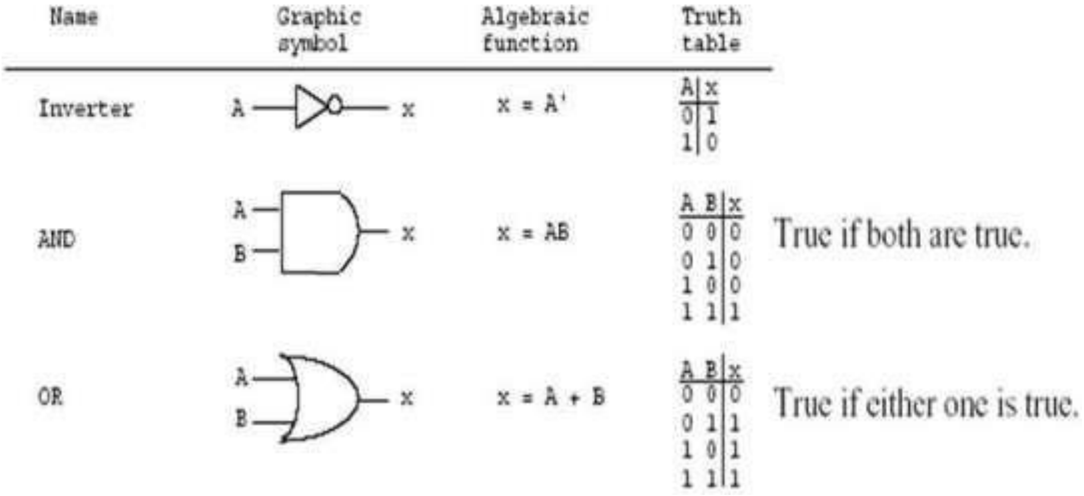
$= a'b' \ldots z'$

(b)$(a.b\ldots z)'= a'+b' + \ldots z'$

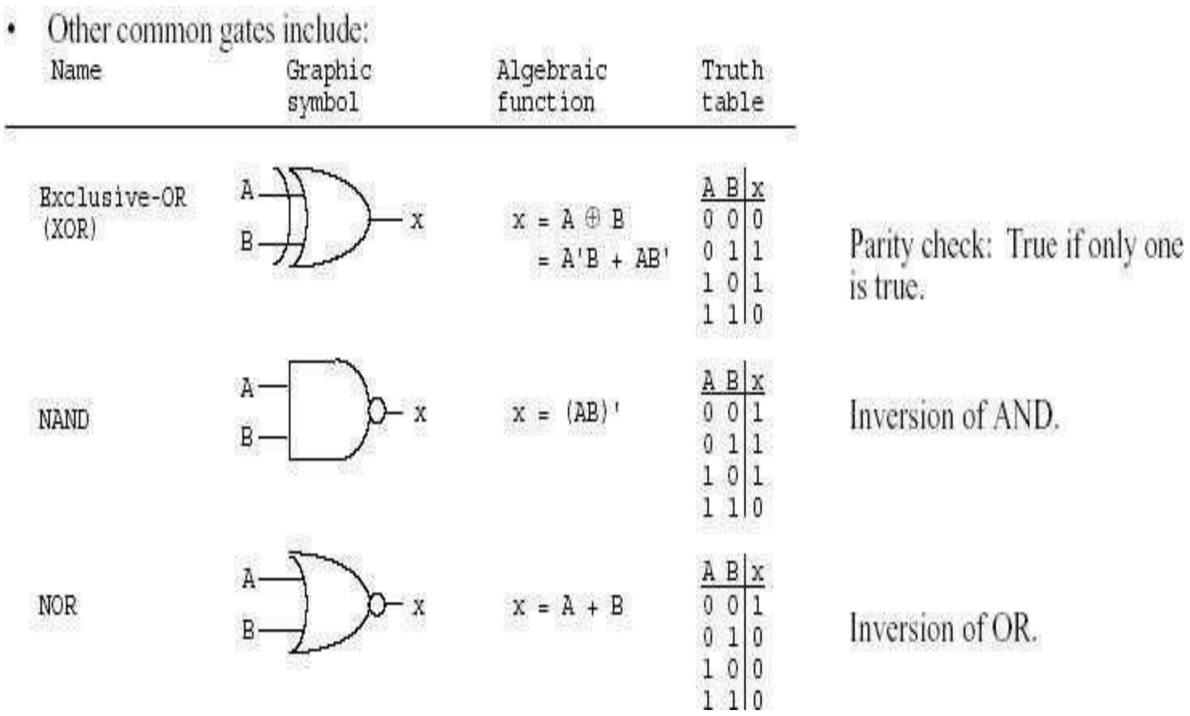
LOGICGATES

Formallogic:Informallogic,astatement(proposition)isadeclarativesentencethatiseither true(1) or false (0). It is easier to communicate with computers using formal logic.

- Booleanvariable:Takes onlytwovalues–eithertrue(1)orfalse(0).Theyareusedasbasicunits of formal logic.
- Booleanalgebra:Dealswithbinaryvariablesandlogicoperationsoperatingonthosevariables.
- Logicdiagram:Composedofgraphicsymbolsforlogicgates.Asimplecircuitsketchthat represents inputs and outputs of Boolean functions.



Name	Graphic symbol	Algebraic function	Truth table	
Inverter	A x	x = A'	A | x 0 | 1 1 | 0	
AND	A B x	x = AB	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	True if both are true.
OR	A B x	x = A + B	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	True if either one is true.



- Other common gates include:

Name	Graphic symbol	Algebraic function	Truth table	
Exclusive-OR (XOR)		$x = A \oplus B$ $= A'B + AB'$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	Parity check: True if only one is true.
NAND		$x = (AB)'$	A B | x 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	Inversion of AND.
NOR		$x = A + B$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	Inversion of OR.

Minimization of switching functions is to obtain logic circuits with least circuit complexity. This goal is very difficult since how a minimal function relates to the implementation technology is important. For example, If we are building a logic circuit that uses discrete logic made of small scale Integration ICs(SSIs) like 7400 series, inwhich basic buildingblockareconstructed and are availableforuse.Thegoalofminimizationwouldbetoreducethenumberof ICsandnotthelogic gates.Forexample,Ifwerequiretwo6andgatesand5Orgates,wewouldrequire2ANDICs(each has 4 AND gates) and one OR IC. (4 gates). On the other hand if the same logic could be implemented with only 10 nand gates, we require only 3 ICs. Similarly when we design logic on Programmable device, we may implement the design with certain number of gates and remaining gates may not be used.

Whatevermaybethe criteriaof minimization wewould beguided bythefollowing:

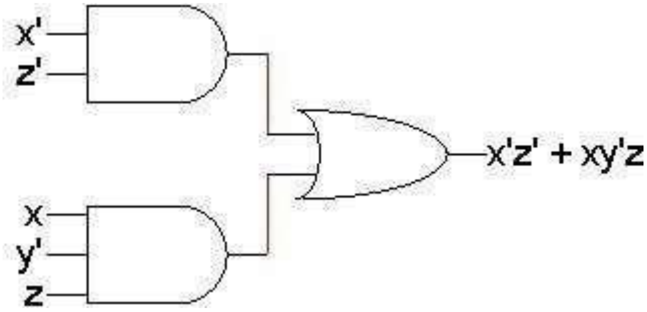
- Booleanalgebrahelpsussimplifyexpressionsandcircuits
- KarnaughMap:AgraphicaltechniqueforsimplifyingaBooleanexpressionintoeitherform:
  - o minimal sum of products

    (MSP) o minimalproductof

    sums (MPS)
- Goalofthesimplification.

- Thereareaminimalnumberofproduct/sumtermsof Eachtermhasaminimal number of literals

- Circuit-wise,thisleadstoa*minimal*two-levelimplementation



- Atwo-variablefunctionhasfourpossibleminterms.Wecanre-arrangethesemintermsintoa Karnaugh map

x	Y	Minterm
0	0	x’y’
0	1	x’y
1	0	xy’
1	1	Xy

X	0	1
0	x’y’	x’y
1	xy’	xy

- Nowwecaneasilyseewhichmintermscontaincommonliterals
  - Mintermsontheleftandrightsidescontainy’andyrespectively
  - Mintermsinthetopandbottomrowscontainx’andxrespectively

		Y	
		0	1
X	0	x’y’	x’y
	1	xy’	xy

	y’	y
x’	x’y’	x’y
x	xy’	xy

K-mapSimplification

- Imagineatwo-variablesumofmintermsx’y’+x’y
- BothofthesemintermsappearinthetoprowofaKarnaughmap,whichmeansthattheyboth contain the literal x’

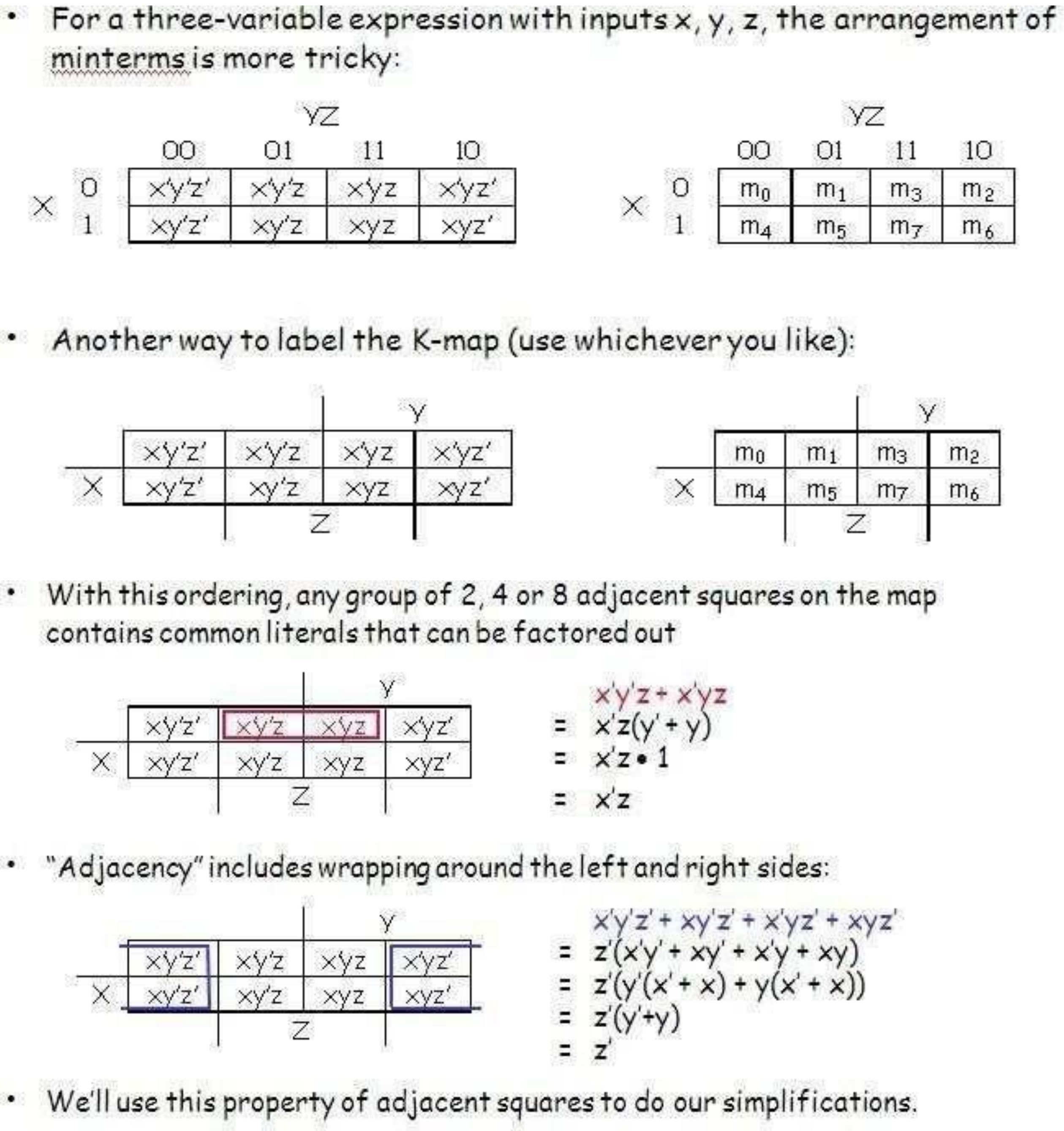
		y
	x’y’	x’y
x	xy’	xy

- WhathappensifyousimplifythisexpressionusingBooleanalgebra?
- x’y’+x’y=x’(y’+y)[Distributive]

=x’+1 [y+y’=1]

=x’ [x+1=x]

AThree-VariableKarnaughMap



- For a three-variable expression with inputs x, y, z, the arrangement of minterms is more tricky:

		YZ			
		00	01	11	10
X	0	x'y'z'	x'y'z	x'yz	x'yz'
	1	xy'z'	xy'z	xyz	xyz'

		YZ			
		00	01	11	10
X	0	m_0	m_1	m_3	m_2
	1	m_4	m_5	m_7	m_6

- Another way to label the K-map (use whichever you like):

			Y	
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z	Z	

			Y	
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z	Z	

- With this ordering, any group of 2, 4 or 8 adjacent squares on the map contains common literals that can be factored out

			Y	
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z	Z	

$$\begin{aligned} & x'y'z + x'yz \\ = \; & x'z(y' + y) \\ = \; & x'z \bullet 1 \\ = \; & x'z \end{aligned}$$

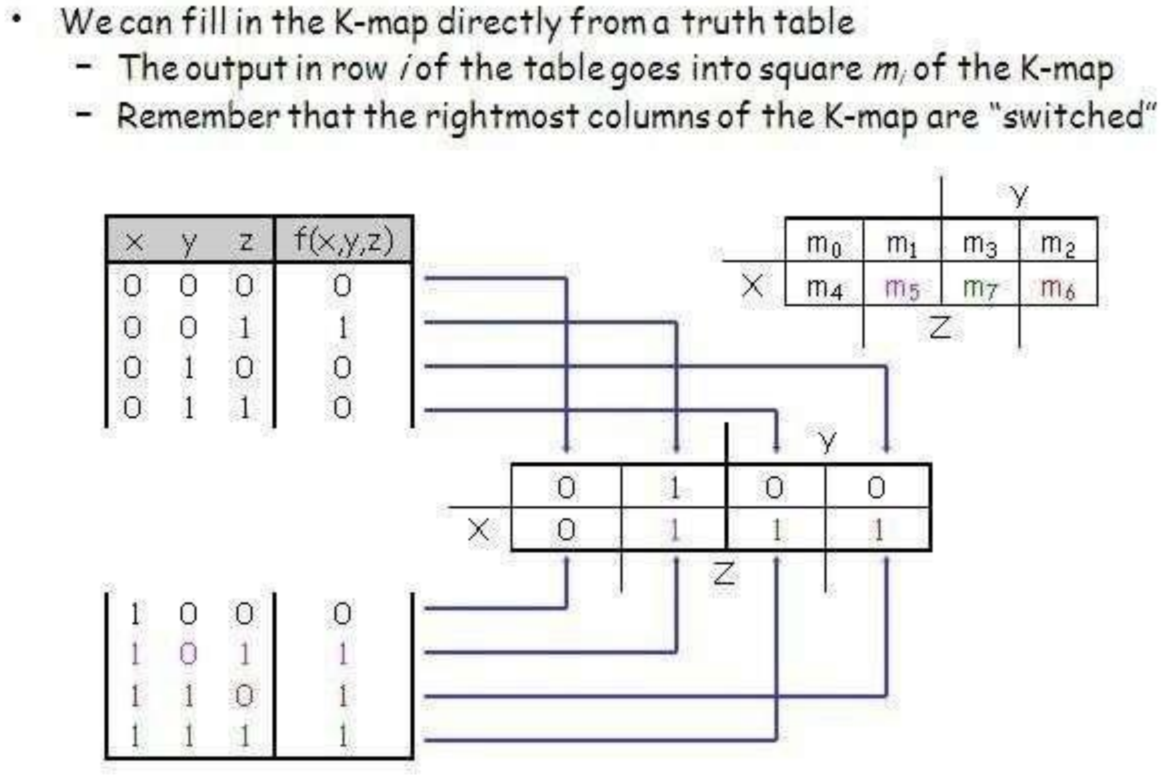
- "Adjacency" includes wrapping around the left and right sides:

			Y	
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z	Z	

$$\begin{aligned} & x'y'z' + xy'z' + x'yz' + xyz' \\ = \; & z'(x'y' + xy' + x'y + xy) \\ = \; & z'(y'(x' + x) + y(x' + x)) \\ = \; & z'(y'+y) \\ = \; & z' \end{aligned}$$

- We'll use this property of adjacent squares to do our simplifications.

K-mapsFromTruthTables



- We can fill in the K-map directly from a truth table
  - The output in row *i* of the table goes into square m_i of the K-map
  - Remember that the rightmost columns of the K-map are "switched"

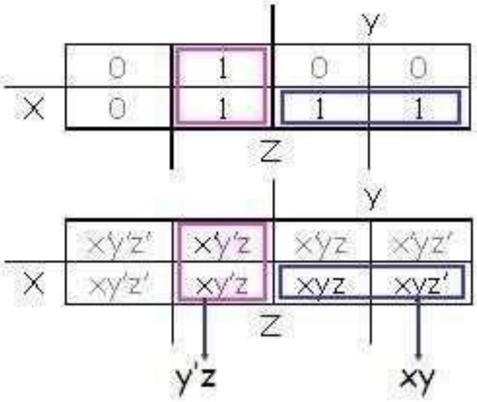
x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6

	0	1	0	0
X	0	1	1	1

ReadingtheMSPfromtheK-map

- You can find the minimal SoP expression
  - Each rectangle corresponds to one product term
  - The product is determined by finding the common literals in that rectangle

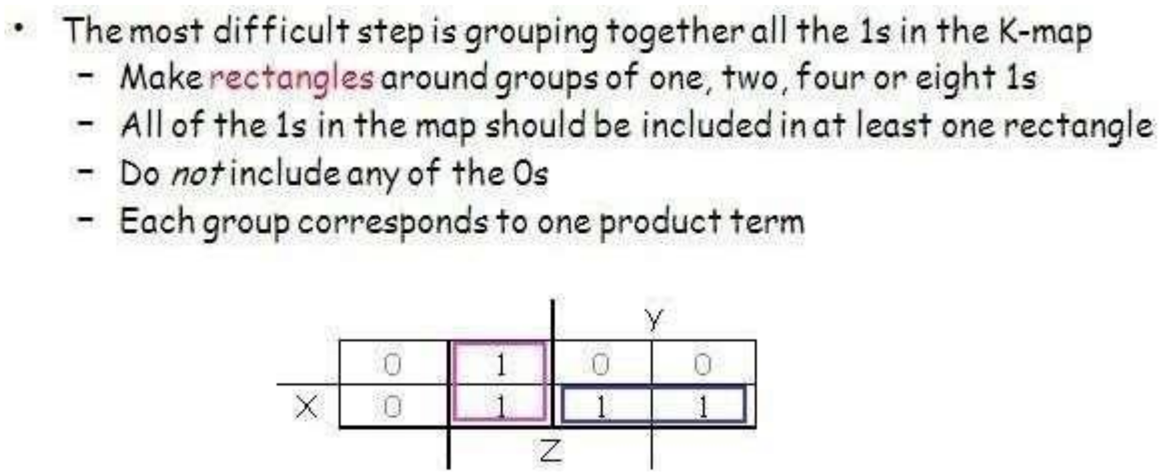


	0	1	0	0
X	0	1	1	1

	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'

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

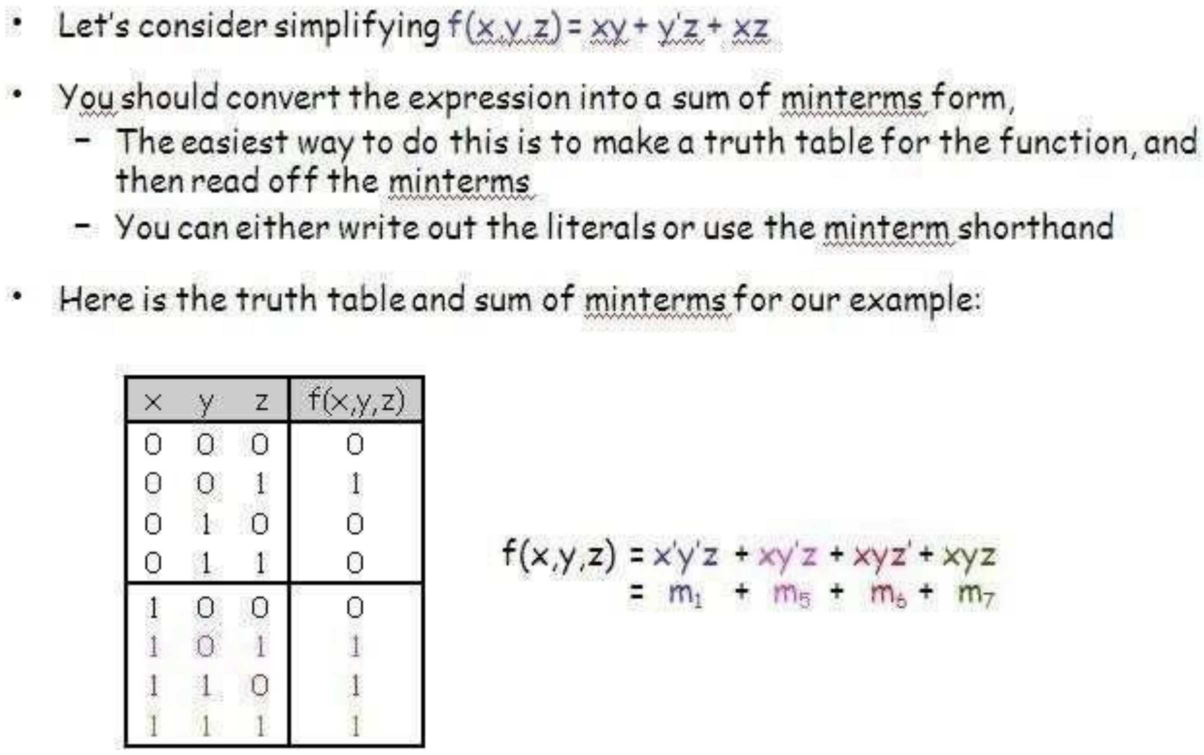
GroupingtheMintermsTogether



- The most difficult step is grouping together all the 1s in the K-map
  - Make rectangles around groups of one, two, four or eight 1s
  - All of the 1s in the map should be included in at least one rectangle
  - Do *not* include any of the 0s
  - Each group corresponds to one product term

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

K-mapSimplificationofSoPExpressions

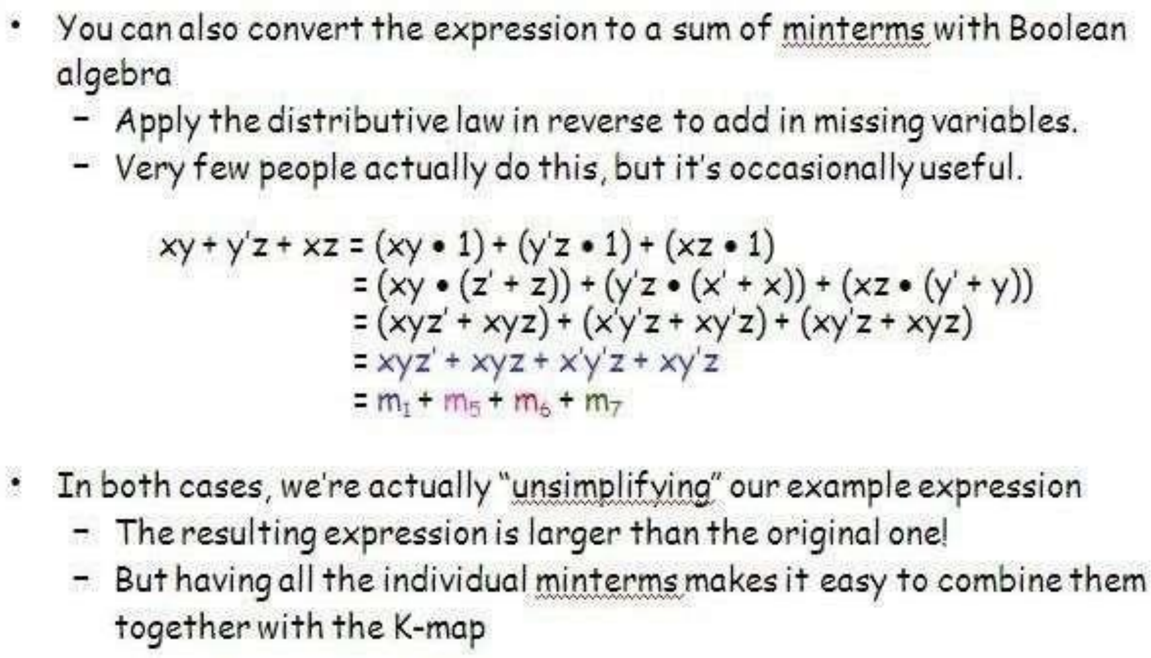


- Let's consider simplifying $f(x,y,z) = xy + y'z + xz$
- You should convert the expression into a sum of minterms form,
  - The easiest way to do this is to make a truth table for the function, and then read off the minterms
  - You can either write out the literals or use the minterm shorthand
- Here is the truth table and sum of minterms for our example:

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

$$\begin{aligned} f(x,y,z) &= x'y'z + xy'z + xyz' + xyz \\ &= m_1 + m_5 + m_6 + m_7 \end{aligned}$$

UnsimplifyingExpressions



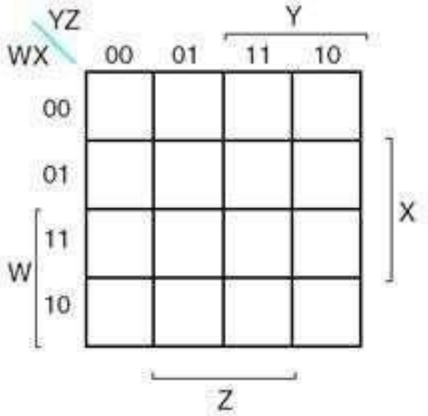
- You can also convert the expression to a sum of minterms with Boolean algebra
  - Apply the distributive law in reverse to add in missing variables.
  - Very few people actually do this, but it's occasionally useful.

$$\begin{aligned} xy + y'z + xz &= (xy \bullet 1) + (y'z \bullet 1) + (xz \bullet 1) \\ &= (xy \bullet (z' + z)) + (y'z \bullet (x' + x)) + (xz \bullet (y' + y)) \\ &= (xyz' + xyz) + (x'y'z + xy'z) + (xy'z + xyz) \\ &= xyz' + xyz + x'y'z + xy'z \\ &= m_1 + m_5 + m_6 + m_7 \end{aligned}$$

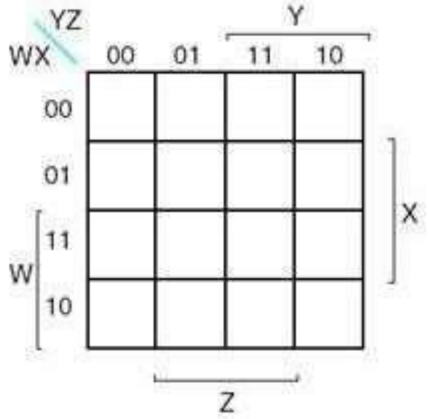
- In both cases, we're actually "unsimplifying" our example expression
  - The resulting expression is larger than the original one!
  - But having all the individual minterms makes it easy to combine them together with the K-map

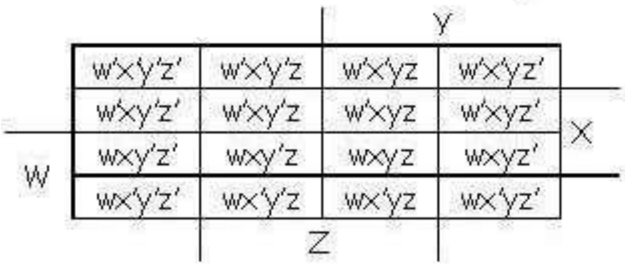
Four-variableK-maps–f(W,X,Y,Z)

- We can do four-variable expressions too!
  - The minterms in the third and fourth columns, *and* in the third and fourth rows, are switched around.
  - Again, this ensures that adjacent squares have common literals

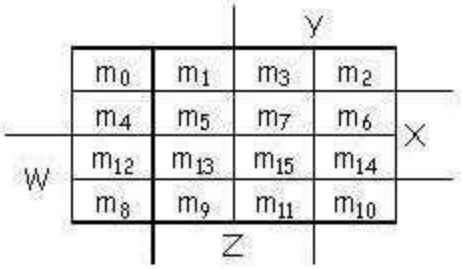


- Grouping minterms is similar to the three-variable case, but:
  - You can have rectangular groups of 1, 2, 4, 8 or 16 minterms
  - You can wrap around *all four* sides





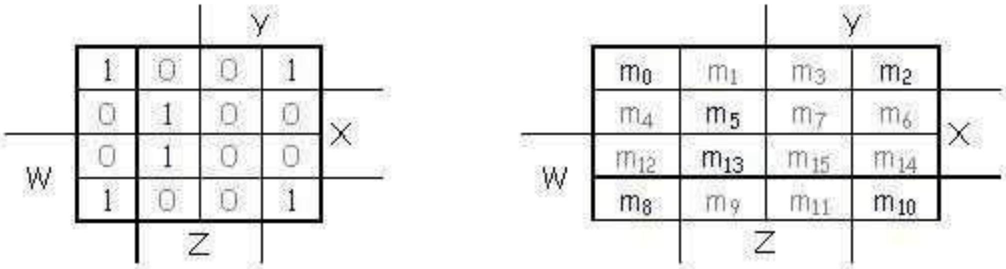
			Y		
	w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'	
	w'xy'z'	w'xy'z	w'xyz	w'xyz'	X
W	wxy'z'	wxy'z	wxyz	wxyz'	
	wx'y'z'	wx'y'z	wx'yz	wx'yz'	
		Z			



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

Simplifym0+m2+m5+m8+m10+m13

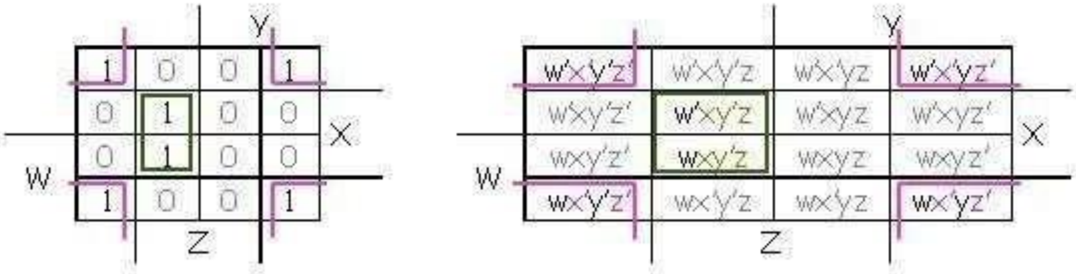
- The expression is already a sum of minterms, so here's the K-map:



1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1

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}

- We can make the following groups, resulting in the MSP x'z' + xy'z

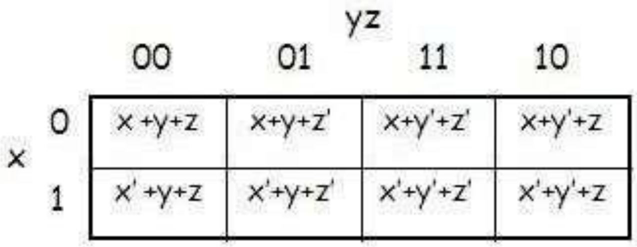


1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1

w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'
w'xy'z'	w'xy'z	w'xyz	w'xyz'
wxy'z'	wxy'z	wxyz	wxyz'
wx'y'z'	wx'y'z	wx'yz	wx'yz'

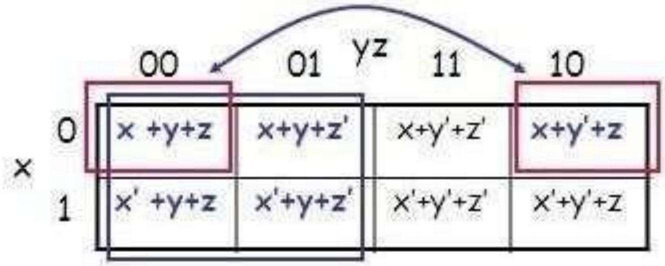
PoSOptimization

- Maxterms are grouped to find minimal PoS expression

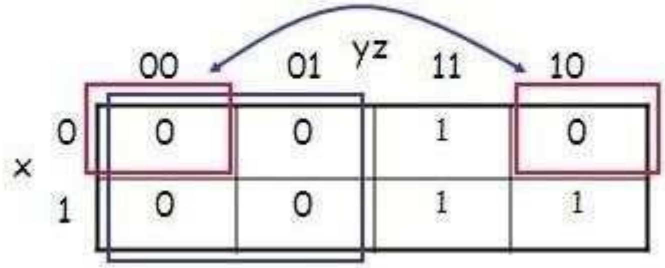


x \ yz	00	01	11	10
0	x +y+z	x+y+z'	x+y'+z'	x+y'+z
1	x' +y+z	x'+y+z'	x'+y'+z'	x'+y'+z

- F(W,X,Y,Z)= ∏ M(0,1,2,4,5)



x \ yz	00	01	11	10
0	x +y+z	x+y+z'	x+y'+z'	x+y'+z
1	x' +y+z	x'+y+z'	x'+y'+z'	x'+y'+z



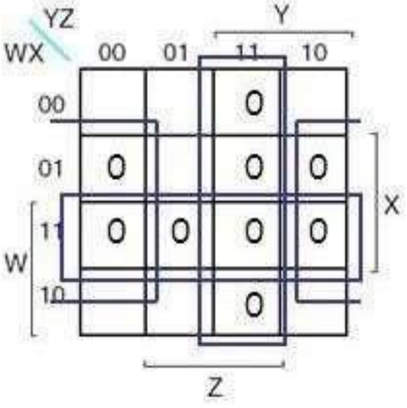
x \ yz	00	01	11	10
0	0	0	1	0
1	0	0	1	1

F(W,X,Y,Z)= Y . (X + Z)

PoSOptimizationfromSoP

F(W,X,Y,Z)= Σm(0,1,2,5,8,9,10)

= ∏ M(3,4,6,7,11,12,13,14,15)



WX \ YZ	00	01	11	10
00			0	
01	0		0	0
11	0	0	0	0
10			0	

F(W,X,Y,Z)= (W' + X')(Y' + Z')(X' + Z)

Or,

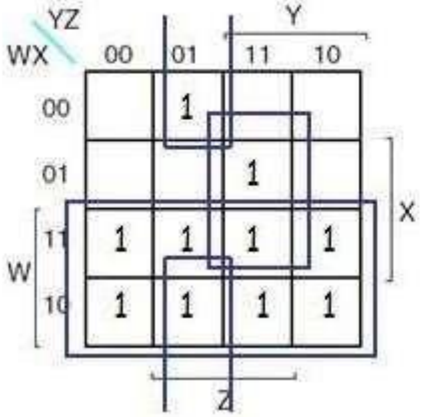
F(W,X,Y,Z)= X'Y' + X'Z' + W'Y'Z

Which one is the minimal one?

SoPOptimizationfromPoS

$F(W,X,Y,Z) = \prod M(0,2,3,4,5,6)$

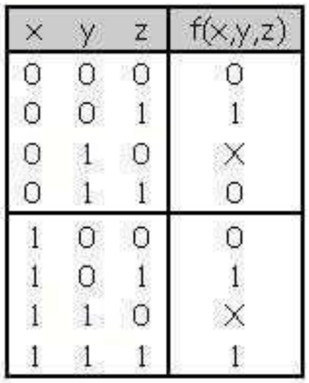
$= \sum m(1,7,8,9,10,11,12,13,14,15)$



$F(W,X,Y,Z) = W + XYZ + X'Y'Z$

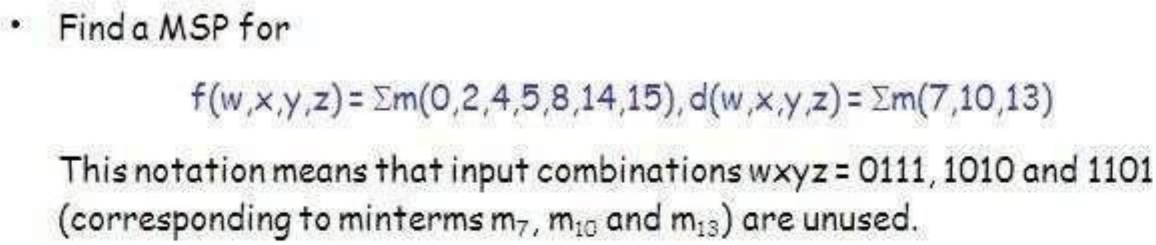
Don’tcare

- You don't always need all 2^n input combinations in an n-variable function
  - If you can guarantee that certain input combinations never occur
  - If some outputs aren't used in the rest of the circuit
- We mark don't-care outputs in truth tables and K-maps with Xs.



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

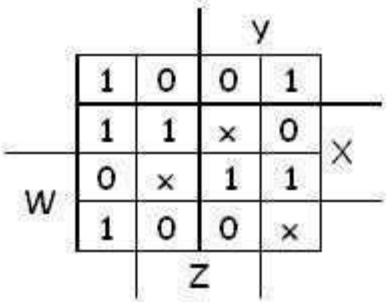
- Within a K-map, each X can be considered as either 0 or 1. You should pick the interpretation that allows for the most simplification.



- Find a MSP for

$$f(w,x,y,z) = \Sigma m(0,2,4,5,8,14,15), d(w,x,y,z) = \Sigma m(7,10,13)$$

This notation means that input combinations wxyz = 0111, 1010 and 1101 (corresponding to minterms m_7, m_{10} and m_{13}) are unused.



1	0	0	1
1	1	x	0
0	x	1	1
1	0	0	x

K-mapSummary

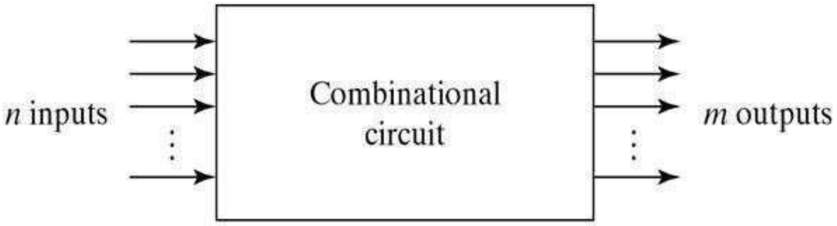
- K-mapsareanalternativetoalgebraforsimplifyingexpressions
  - TheresultisaMSP/MPS,whichleadstoaminimaltwo-levelcircuit
  - It’seasytohandledon’t-careconditions
  - K-mapsarereallyonlygoodformanualsimplificationofsmallexpressions...
  - Thingstokeepinmind:
  - RememberthecorrectorderofmintermsZ/maxtermsontheK-map
  - Whengrouping, youcanwraparoundallsidesoftheK-map,and your groups can overlap
  - Makeasfewrectanglesaspossible,butmakeeach ofthemaslargeas possible.This leads to fewer, but simpler, product terms
  - Theremaybemorethanonevalidsolution

UNIT-II

COMBINATIONALCIRCUITS

CombinationalLogic

- Logiccircuitsfordigitalsystemsmaybecombinationalorsequential.
- Acombinationalcircuitconsistsofinputvariables,logicgates,andoutputvariables.



Forninputvariables,thereare2^{n}possiblecombinationsofbinaryinputvariables.Foreach possible input Combination ,there is one and only one possible output combination.A combinational circuit can be described by m Boolean functions one for each output variables.Usually the input s comes from flip-flops and outputs goto flip-flops.

DesignProcedure:

1. Theproblemisstated
2. Thenumberofavailableinputvariablesandrequiredoutputvariables isdetermined.
3. Theinputandoutputvariablesareassignedlettersymbols.
4. Thetruthtablethatdefinestherequiredrelationshipbetweeninputsandoutputsis derived.
5. ThesimplifiedBooleanfunctionforeachoutputisobtained.
6. Thelogicdiagramisdrawn.

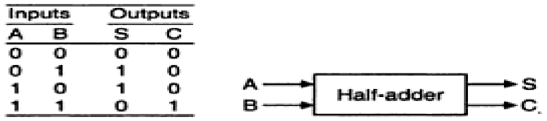
Adders:

Digital computers perform variety of information processing tasks,the one is arithmetic operations.Andthemost basicarithmeticoperationistheadditionoftwobinarydigits.i.e,4basic possible operations are:

0+0=0,0+1=1,1+0=1,1+1=10

Thefirstthreeoperationsproduceasumwhoselengthisonedigit,butwhenaugendsandaddend bits areequalto1,thebinarysumconsistsof twodigits.Thehighersignificantbitofthisresultis calledacarry.Acombinationalcircuitthatperformstheadditionoftwobitsiscalledahalf-adder. One that performs the addition of 3 bits (two significant bits & previous carry) is called a full adder.& 2 half adder can employ as a full-adder.

The Half Adder: A Half Adder is a combinational circuit with two binary inputs (augends and addend bits and two binary outputs (sum and carry bits.) It adds the two inputs (A and B) and produces thesum (S)and thecarry(C)bits. It is an arithmeticoperation of addition oftwo single bit words.



Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) Truth table

A
B
Half-adder
S
C

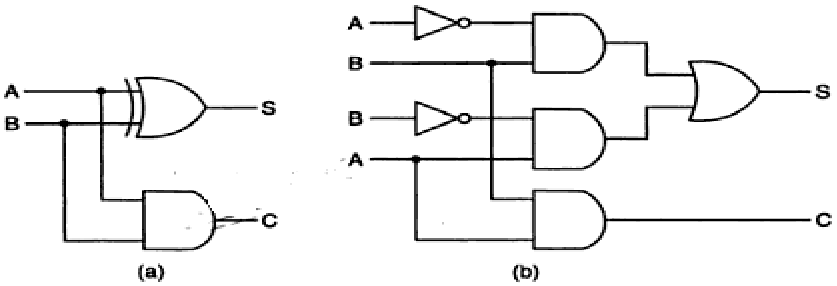
(b) Block diagram

TheSum(S) bit and the carry(C) bit, accordingto the rules of binaryaddition, the sum (S) is the X-OR of A and B (It represents the LSB of the sum). Therefore,

$$S=A'B+AB'$$

Thecarry(C)istheANDofAand B(itis0unlessboththeinputsare1).Therefore, C=AB

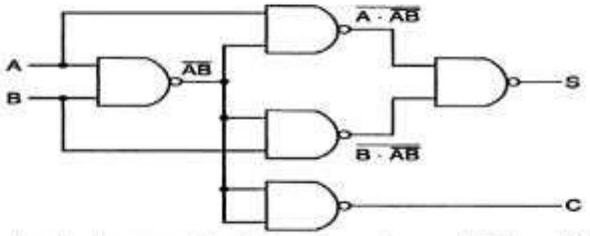
Ahalf-addercanberealizedbyusingoneX-ORgateandoneAND gatea



Logicdiagramsofhalf-adder

NANDLOGIC:

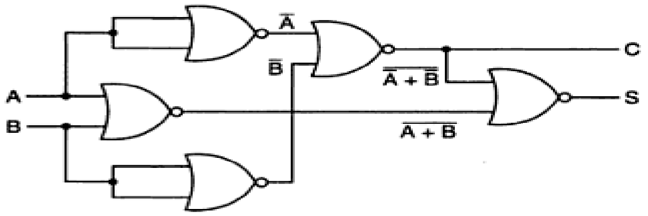
$$S = A\bar{B} + \bar{A}B = A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B}$$
$$= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B})$$
$$= A \cdot \overline{AB} + B \cdot \overline{AB}$$
$$= \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$
$$C = AB = \overline{\overline{AB}}$$


A
B
$\overline{AB}$
$\overline{A \cdot \overline{AB}}$
$\overline{B \cdot \overline{AB}}$
S
C

Logic diagram of a half-adder using only 2-input NAND gates.

NORLogic:

$$S = A\bar{B} + \bar{A}B = A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B}$$
$$= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B})$$
$$= (A + B)(\bar{A} + \bar{B})$$
$$= \overline{\overline{A + B} + \overline{\bar{A} + \bar{B}}}$$
$$C = AB = \overline{\overline{AB}} = \overline{\bar{A} + \bar{B}}$$

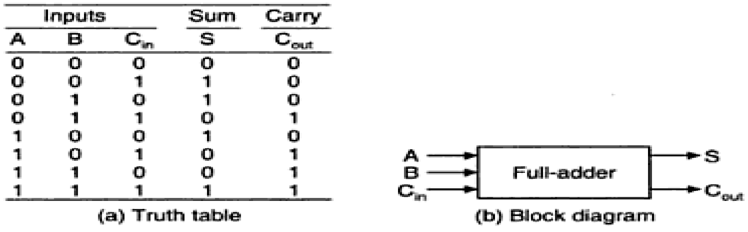


Logic diagram of a half-adder using only 2-input NOR gates.

TheFull Adder:

AFull-adderisacombinationalcircuitthataddstwobitsandacarryandoutputsasumbit andacarrybit.Toaddtwobinarynumbers,each havingtwoormorebits,theLSBscanbeadded byusinga half-adder. The carryresulted from the addition of the LSBs is carried over to the next significantcolumnandaddedtothetwobitsinthatcolumn.So,inthesecondandhighercolumns, thetwodatabitsofthatcolumnandthecarrybitgeneratedfromtheadditioninthepreviouscolumn need to be added.

The full-adder adds the bits A and B and the carry from the previous column called the carry-in C_{in} and outputs thesum bit S and thecarrybit called thecarry-out C_{out} . ThevariableS givesthevalueoftheleastsignificantbitofthesum.ThevariableC$_{out}$givestheoutputcarry.The eight rows under the input variables designate all possible combinations of 1s and 0s that these variablesmayhave.The1sand0sfortheoutputvariablesaredeterminedfromthearithmeticsum of the input bits. When all the bits are 0s , the output is 0. TheS output is equal to 1 when only1 input is equal to 1 or when all the inputs are equal to 1. The Cout has a carryof 1 if two or three inputs are equal to 1.



Inputs			Sum	Carry
A	B	C_{in}	S	C_{out}
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

(a) Truth table

(b) Block diagram

Full-adder.

Fromthetruthtable,acircuitthatwillproducethecorrectsumandcarrybitsinresponsetoevery possible combination of A,B and Cin is described by

$$S = A\overline{B}C\overline{in} + ABC\overline{in} + \overline{A}BCin + A\overline{B}C\overline{in}$$

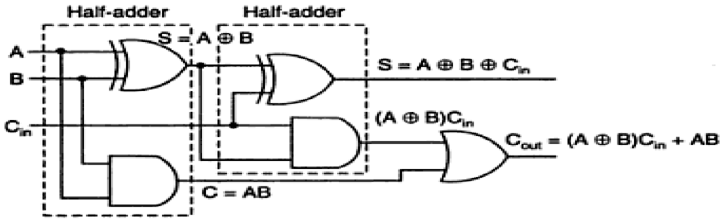
$$Cout = A\underline{B}Cin + ABC\underline{in} + ABCin + \underline{A}BCin$$

and

$$S = A \oplus B \oplus Cin$$

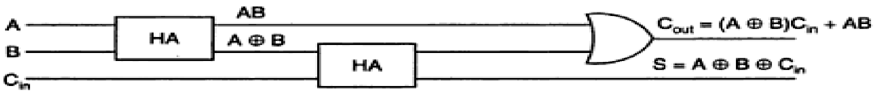
$$Cout = ACin + BCin + AB$$

The sum term of the full-adder is the X-OR of A,B, and Cin, i.e, the sum bit the modulo sumofthedatabitsinthatcolumnandthecarryfromthepreviouscolumn.Thelogicdiagramof the full-adder using two X-OR gates and two AND gates (i.e, Two half adders) and one OR gate is



Logic diagram of a full-adder using two half-adders.

The block diagram of a full-adder using two half-adders is ·



A
B
C_{in}
HA
AB
$A \oplus B$
HA
$C_{out} = (A \oplus B)C_{in} + AB$
$S = A \oplus B \oplus C_{in}$

Block diagram of a full-adder using two half-adders.

Even though a full-adder can be constructed using two half-adders, the disadvantage is that the bits must propagate through several gates in accession, which makes the total propagation delay greater than that of the full-adder circuit using AOI logic.

TheFull-adder neither can also be realized usinguniversal logic,i.e., either onlyNAND gatesor only NOR gates as

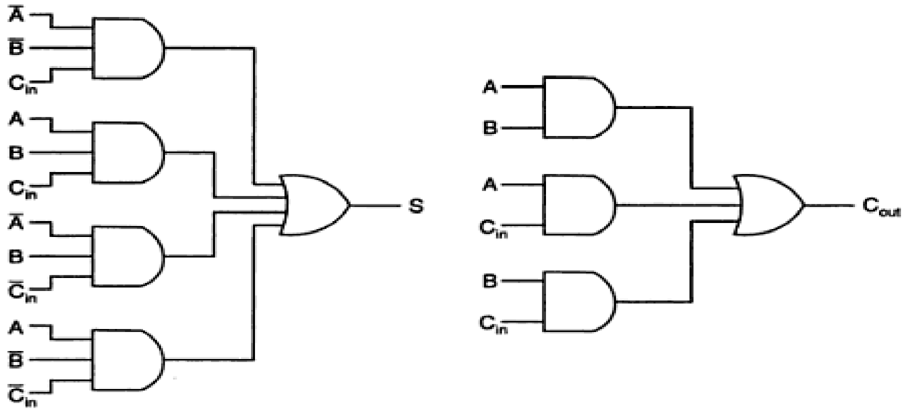
$$A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$

Then

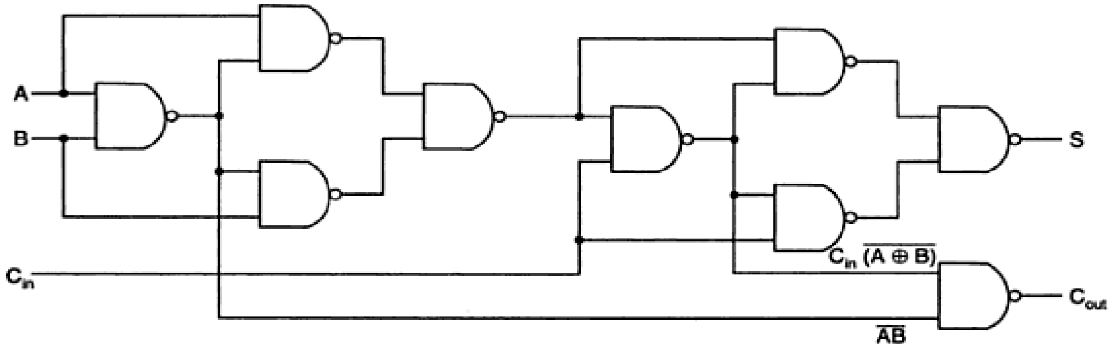
$$S = A \oplus B \oplus C_{in} = \overline{\overline{(A \oplus B) \cdot \overline{(A \oplus B)C_{in}}} \cdot \overline{C_{in} \cdot \overline{(A \oplus B)C_{in}}}}$$

NANDLogic:

$$C_{out} = C_{in}(A \oplus B) + AB = \overline{\overline{C_{in}(A \oplus B)} \cdot \overline{AB}}$$



Sum and carry bits of a full-adder using AOI logic.



Logic diagram of a full-adder using only 2-input NAND gates.

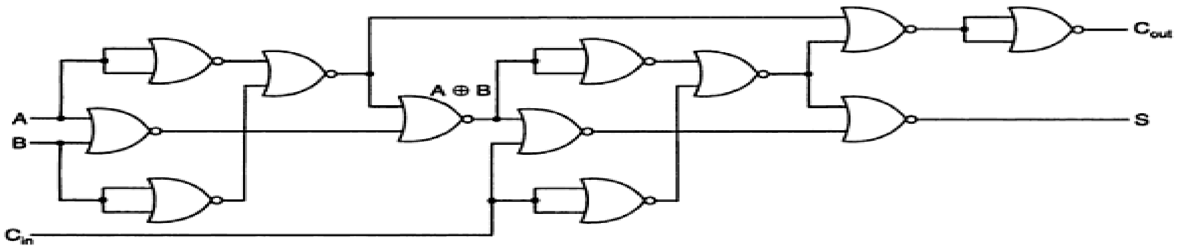
NORLogic:

$$A \oplus B = \overline{\overline{(A + B)} + \overline{\overline{A} + \overline{B}}}$$

Then

$$S = A \oplus B \oplus C_{in} = \overline{\overline{(A \oplus B) + C_{in}} + \overline{\overline{(A \oplus B)} + \overline{C}_{in}}}$$

$$C_{out} = AB + C_{in}(A \oplus B) = \overline{\overline{A} + \overline{B}} + \overline{\overline{C}_{in} + \overline{A \oplus B}}$$



Logic diagram of a full-adder using only 2-input NOR gates.

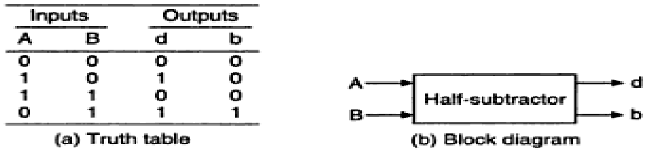
Subtractors:

Thesubtractionof twobinarynumbersmaybeaccomplishedbytakingthecomplementof thesubtrahendandaddingittotheminuend.Bythis,thesubtractionoperationbecomesanaddition operation and instead of having a separate circuit for subtraction, the adder itself can be used to perform subtraction. This results in reduction of hardware. In subtraction, each subtrahend bit of thenumberissubtractedfromitscorrespondingsignificantminuendbittoformadifferencebit.If the minuend bit is smaller than the subtrahend bit, a 1 is borrowed from the next significant position., that has been borrowed must be conveyed to the next higher pair of bits by means of a signal coming out (output) of a given stage and going into (input) the next higher stage.

TheHalf-Subtractor:

AHalf-subtractorisacombinationalcircuitthatsubtractsonebitfromtheotherandproduces thedifference. It also has an output to specifyifa 1 has been borrowed. . It is used to subtract the LSB of the subtrahend from the LSB of the minuend when one binarynumber is subtracted from the other.

A Half-subtractor is a combinational circuit with two inputs A and B and two outputs d and b. d indicatesthedifferenceandbistheoutputsignalgeneratedthatinformsthenextstagethata1has beenborrowed.WhenabitBissubtractedfromanotherbitA,adifferencebit(d)andaborrowbit (b)resultaccordingtotherulesgivenas



Inputs		Outputs	
A	B	d	b
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

(a) Truth table

(b) Block diagram

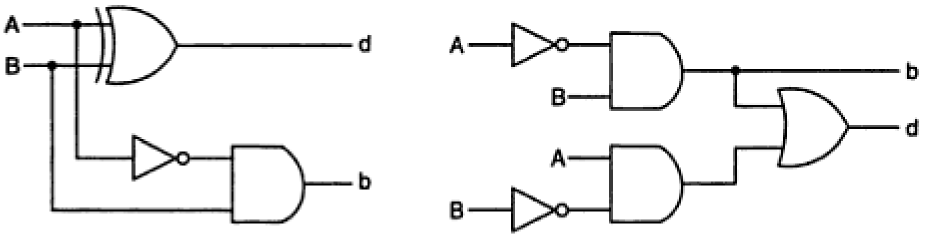
Half-subtractor.

TheoutputborrowbisaOaslongasA≥B.Itisa1forA=0andB=1.Thedoutputistheresultof the arithmetic operation2b+A-B.

A circuit that produces the correct difference and borrow bits in response to every possible combination of the two 1-bit numbers is , therefore ,

d=A*B*+*A* $A \oplus B$ andb=*A*B

That is, the difference bit is obtained by X-OR ing the two inputs, and the borrow bit is obtained by ANDing the complement of the minuend with the subtrahend.Note that logic for this exactly the same as the logic for output S in the half-adder.



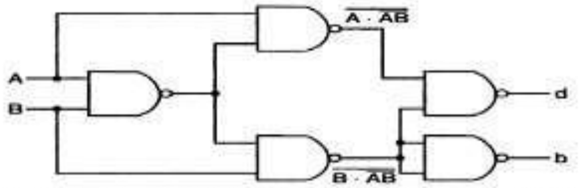
Logic diagrams of a half-subtractor.

A half-substractor can also be realized using universal logic either using only NAND gates or using NOR gates as:

NANDLogic:

$$d = A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$

$$b = \bar{A}B = B(\bar{A} + \bar{B}) = B(\overline{AB}) = \overline{\overline{B \cdot \overline{AB}}}$$


$\overline{A \cdot \overline{AB}}$
A
B
d
b
$\overline{B \cdot \overline{AB}}$

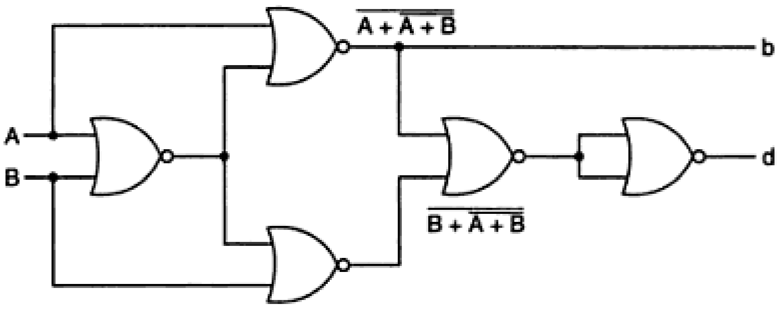
Logic diagram of a half-subtractor using only 2-input NAND gates.

NORLogic:

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

$$= \bar{B}(A + B) + \bar{A}(A + B) = \overline{B + \overline{A + B}} + \overline{A + \overline{A + B}}$$

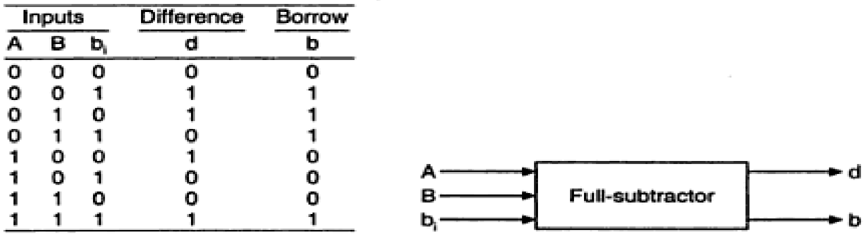
$$d = \bar{A}B = \bar{A}(A + B) = \overline{\overline{\bar{A}(A + B)}} = \overline{A + (\overline{A + B})}$$



Logic diagram of a half-subtractor using only 2-input NOR gates.

TheFull-Subtractor:

Thehalf-subtractorcanbeonlyforLSBsubtraction.IFthereisaborrowduringthesubtraction of the LSBs, it affects the subtraction in the next higher column; the subtrahend bit is subtracted from the minuend bit, considering the borrow from that column used for the subtraction in the preceding column. Such a subtraction is performed by a full-subtractor. It subtracts one bit (B) fromanotherbit(A),whenalreadythereisaborrowb$_i$ fromthiscolumnforthesubtractioninthe preceding column, and outputs the difference bit (d) and the borrow bit(b) required from the next d and b. The two outputs present the difference and output borrow. The 1s and 0s for the output variables are determined from the subtraction of A-B-b$_i$.



Inputs			Difference	Borrow
A	B	b_i	d	b
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

A
B
b_i
Full-subtractor
d
b

(a) Truth table

(b) Block diagram

Full-subtractor.

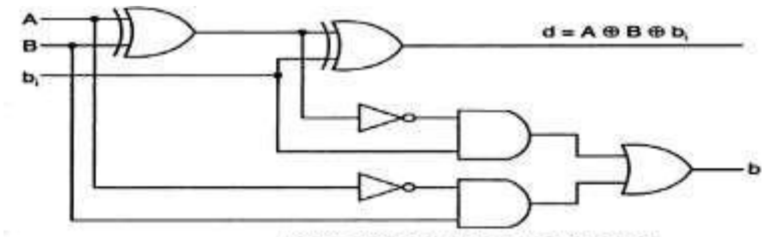
Fromthetruthtable,acircuitthatwillproducethecorrectdifferenceandborrowbitsinresponse to every possiblecombinations of A,B and b$_i$ is

$$d = \bar{A}\bar{B}b_i + \bar{A}B\,\bar{b}_i + A\bar{B}\,\bar{b}_i + ABb_i$$
$$= b_i(AB + \bar{A}\bar{B}) + \bar{b}_i(A\bar{B} + \bar{A}B)$$
$$= b_i(\overline{A \oplus B}) + \bar{b}_i(A \oplus B) = A \oplus B \oplus b_i$$

and

$$b = \bar{A}\bar{B}b_i + \bar{A}B\,\bar{b}_i + \bar{A}Bb_i + ABb_i = \bar{A}B(b_i + \bar{b}_i) + (AB + \bar{A}\bar{B})b_i$$
$$= \bar{A}B + (\overline{A \oplus B})b_i$$

Afull-subtractorcanberealizedusingX-ORgatesandAOIgatesas



Logic diagram of a full-subtractor.

ThefullsubtractorcanalsoberealizedusinguniversallogiceitherusingonlyNANDgatesor using NOR gates as:

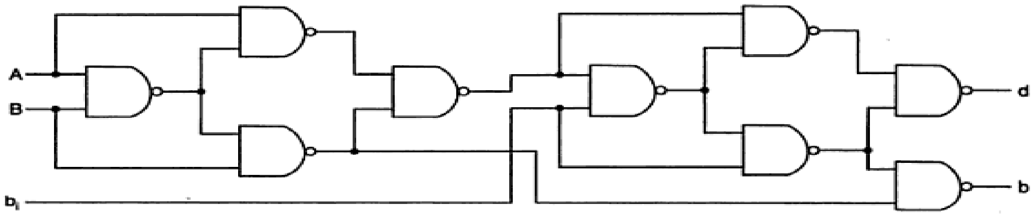
NANDLogic:

$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}} = \overline{\overline{(A \oplus B)\overline{(A \oplus B)b_i}} \cdot \overline{b_i\overline{(A \oplus B)b_i}}}$$

$$b = \overline{A}B + b_i(\overline{A \oplus B}) = \overline{\overline{\overline{A}B + b_i(\overline{A \oplus B})}}$$

$$= \overline{\overline{\overline{A}B} \cdot \overline{b_i(\overline{A \oplus B})}} = \overline{\overline{B(\overline{A} + \overline{B})} \cdot \overline{b_i(\overline{b_i} + (\overline{A \oplus B}))]}}$$

$$= \overline{\overline{B \cdot \overline{AB}} \cdot \overline{b_i[\overline{b_i \cdot (A \oplus B)}]}}$$



Logic diagram of a full-subtractor using only 2-input NAND gates.

NORLogic:

$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}}$$

$$= \overline{(A \oplus B)b_i + (\overline{A \oplus B})\overline{b_i}}$$

$$= \overline{[(A \oplus B) + (\overline{A \oplus B})\overline{b_i}][b_i + (\overline{A \oplus B})\overline{b_i}]}$$

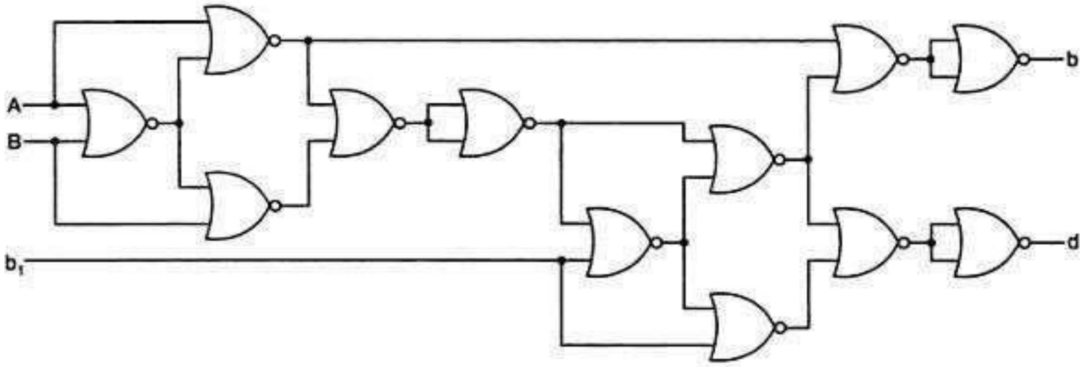
$$= \overline{(A \oplus B) + \overline{(A \oplus B) + b_i}} + \overline{b_i + \overline{(A \oplus B) + b_i}}$$

$$= \overline{\overline{\overline{(A \oplus B) + \overline{(A \oplus B) + b_i}} + \overline{b_i + \overline{(A \oplus B) + b_i}}}}$$

$$b = \overline{A}B + b_i(\overline{A \oplus B})$$

$$= \overline{A}(A + B) + (\overline{A \oplus B})[(A \oplus B) + b_i]$$

$$= \overline{\overline{\overline{A + \overline{(A + B)}} + \overline{(A \oplus B) + \overline{(A \oplus B) + b_i}}}}$$

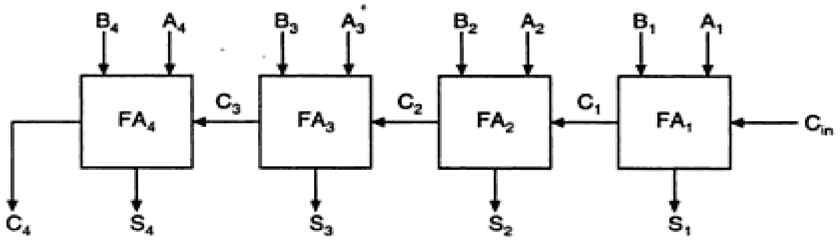


Logic diagram of a full subtractor using only 2-input NOR gates.

BinaryParallelAdder:

A binary parallel adder is a digital circuit that adds two binary numbers in parallel form and produces the arithmetic sum of those numbers in parallel form. It consists of full adders connected in a chain , with the output carry from each full-adder connected to the input carry of the next full-adder in the chain.

The interconnection of four full-adder (FA) circuits to provide a 4-bit parallel adder. The augendsbitsofAandaddendbitsofBaredesignatedbysubscriptnumbersfromrighttoleft,with subscript 1 denoting the lower –order bit. The carries are connected in a chain through the full- adders. The input carryto the adder is Cin and the output carryis C4. The S output generates the required sum bits. When the 4-bit full-adder circuit is enclosed within an IC package, it has four terminals for the augends bits, four terminals for the addend bits, four terminals for the sum bits, and two terminals for the input and output carries. AN n-bit parallel adder requires n-full adders. It canbeconstructedfrom4-bit,2-bitand1-bit fulladder ICsbycascadingseveralpackages. The outputcarryfromonepackagemustbeconnectedtotheinputcarryoftheonewiththenexthigher –order bits. The 4-bit full adder is a typical example of an MSIfunction.



Logic diagram of a 4-bit binary parallel adder.

Ripplecarryadder:

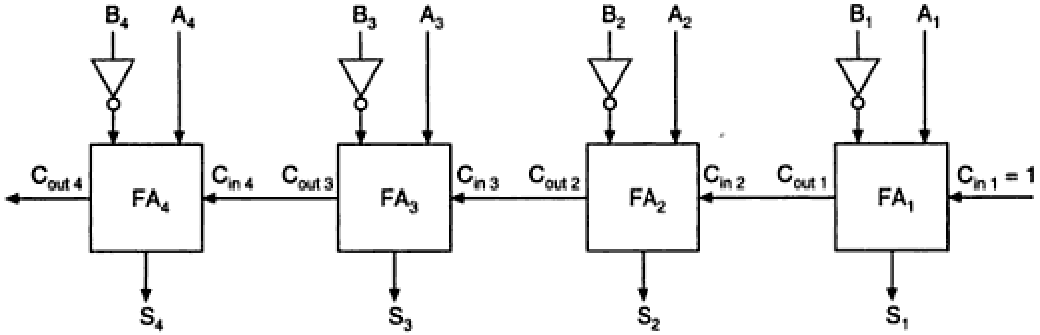
Intheparalleladder,thecarry–outofeachstageisconnectedtothecarry-inofthe next stage. The sum and carry-out bits of any stage cannot be produced, until sometime after the carry-in of that stage occurs. This is due to the propagation delays in the logic circuitry, whichleadtoatimedelayintheadditionprocess.Thecarrypropagationdelayforeachfull-adder is the time between the application of the carry-in and the occurrence of the carry-out.

The 4-bit parallel adder, the sum (S_1) and carry-out (C_1) bits given by FA_1 are not valid, until after the propagation delay of FA_1. Similarly, the sum S_2 and carry-out (C_2) bits given by FA_2 are not valid until after the cumulative propagation delayof two full adders (FA_1 and FA_2) , and soon.Ateachstage,thesumbitisnotvaliduntilafterthecarrybitsinallthEprecedingstagesare valid. Carrybits must propagate or ripple through all stages before the most significant sum bit is valid.Thus,thetotalsum(theparalleloutput)isnotvaliduntilafterthecumulativedelayofallthe adders.

The parallel adder in which the carry-out of each full-adder is the carry-in to the next most significant adder is called a ripple carry adder.. The greater the number of bits that a ripple carry addermustadd,thegreaterthetimerequiredforittoperformavalidaddition. Iftwonumbersare added such that no carries occurbetween stages, thentheadd timeis simplythepropagation time through a single full-adder.

4-BitParallelSubtractor:

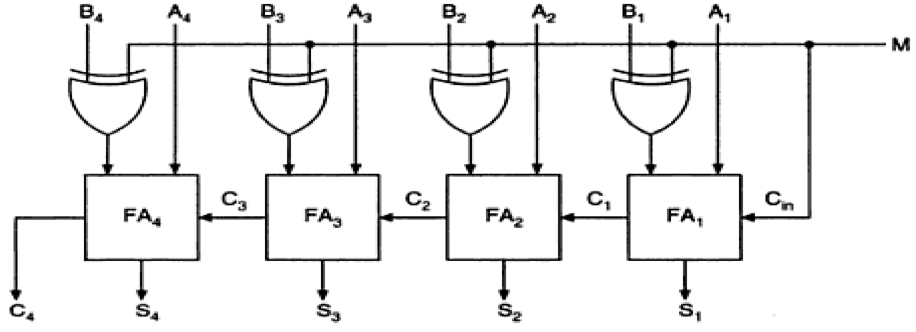
The subtraction of binary numbers can be carried out most conveniently by means of complements,thesubtractionA-BcanbedonebytakingthE2‘scomplementofBandadding ittoA.The2‘scomplement can beobtainedbytakingthe1‘s complement andadding1tothe least significant pair of bits. The 1‘s complement can be implemented with inverters as



Logic diagram of a 4-bit parallel subtractor.

Binary-AdderSubtractor:

A 4-bit adder-subtractor, the addition and subtraction operations are combined into onecircuitwithonecommonbinaryadder.ThisisdonebyincludinganX-ORgatewitheachfull- adder. The mode input M controls the operation. When M=0, the circuit is an adder, and when M=1,thecircuitbecomesasubtractor.EachX-OR gatereceivesinputMandone oftheinputsof B.WhenM=0, $B \oplus 0 = B$ .Thefull-adderreceivesthevalueofB,theinputcarryis0 and the circuit performsA+B. when $B \oplus 1 = B'$ andC1=1.TheBinputsarecomplementedand a1isthroughtheinputcarry.The circuitperformstheoperationAplusthe2‘scomplementofB.



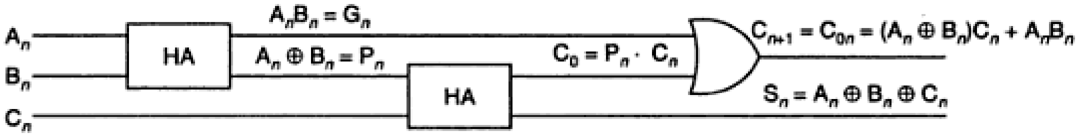
Logic diagram of a 4-bit binary adder-subtractor.

TheLook-Ahead–CarryAdder:

In parallel-adder,the speed with which an addition can be performed is governed by the time required for the carries to propagate or ripple through all of the stages of the adder. The look-ahead carry adder speeds up the process by eliminating this ripple carry delay. It examines all the input bits simultaneously and also generates the carry-in bits for all the stages simultaneously.

The method of speeding up the addition process is based on the two additional functions of the full-adder, called the carry generate and carrypropagate functions.

Consideronefulladderstage;saythenthstageofaparalleladderasshown infig.we knowthatismadebytwohalfaddersandthatthehalfaddercontainsanX-OR gatetoproducethe sum and an AND gate to produce the carry. If both the bits An and Bn are 1s, a carry has to be generatedinthisstageregardlessofwhethertheinputcarryCin isa0ora1.Thisiscalledgenerated carry,expressedasGn=An.Bnwhichhastoappearattheoutputthroughthe ORgateasshownin fig.


A_n
B_n
C_n
HA
HA
$A_nB_n = G_n$
$A_n \oplus B_n = P_n$
$C_0 = P_n \cdot C_n$
$C_{n+1} = C_{0n} = (A_n \oplus B_n)C_n + A_nB_n$
$S_n = A_n \oplus B_n \oplus C_n$

A full adder (nth stage of a parallel adder).

Thereisanotherpossibilityofproducinga carryout.X-ORgateinsidethehalf-adder

at theinputproduces an intermediary sum bit- call itPn–which isexpressedas $P_n = A_n \oplus B_n$. Next Pn and Cn areadded usingthe X-OR gate inside the second half adder to produce thefinal sum bitand $S_n = P_n \oplus C_n$ where $P_n = A_n \oplus B_n$ andoutputcarryC_0=P_n.C_n=($A_n \oplus B_n$)$C_n$which becomes carry for the (n+1) thstage.

ConsiderthecaseofbothPnandCnbeing1.TheinputcarryCnhastobepropagated to the output only if Pn is 1. If Pn is 0, even if Cn is 1, the and gate in the second half-adder will inhibitCn.thecarryout ofthenthstageis1wheneitherGn=1orPn.Cn=1orbothGnandPn.Cn are equal to 1.

Forthefinalsumandcarryoutputsofthenthstage,wegetthefollowingBoolean expressions.

$$S_n = P_n \oplus C_n \text{ where } P_n = A_n \oplus B_n$$
$$C_{on} = C_{n+1} = G_n + P_n C_n \text{ where } G_n = A_n \cdot B_n$$

Observetherecursivenatureof the expressionfor theoutputcarry at the nth stage which becomes the input carry for the (n+1)st stage .it is possible to express the output carryof a higher significant stage is the carry-out of the previous stage.

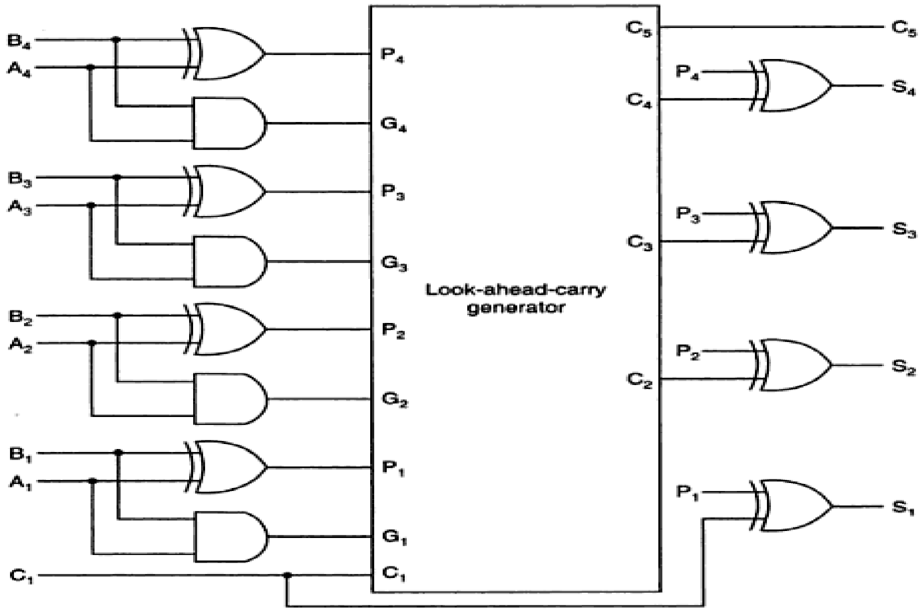
Basedonthese,theexpressionforthecarry-outsofvariousfulladdersare as follows,

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$
$$C_3 = G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$
$$C_4 = G_3 + P_3 \cdot C_3 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

The general expression for *n* stages designated as 0 through ($n - 1$) would be

$$C_n = G_{n-1} + P_{n-1} \cdot C_{n-1} = G_{n-1} + P_{n-1} \cdot G_{n-2} + P_{n-1} \cdot P_{n-2} \cdot G_{n-3} + \ldots + P_{n-1} \cdot \ldots P_0 \cdot C_0$$

Observethatthefinaloutputcarryisexpressedasafunctionofthe input variables in SOP form. Which is two level AND-OR or equivalent NAND-NAND form. Observe that the full look-ahead scheme requires the use of OR gate with (n+1) inputs and AND gates with number of inputs varying from 2 to (n+1).



Logic diagram of a 4-bit look-ahead-carry adder.

2'scomplementAdditionandSubtractionusingParallelAdders:

Most modern computers use the 2‘s complement system to represent negative numbers andtoperformsubtractionoperationsofsignednumberscanbeperformedusingonlytheaddition operation ,if we use the 2‘s complement form to represent negative numbers.

The circuit shown can perform both addition and subtraction in the 2‘s complement. This adder/subtractorcircuitiscontrolledbythecontrolsignalADD/SUB‘.WhentheADD/SUB‘level is HIGH, the circuit performs the addition of the numbers stored in registers A and B. When the ADD/Sub‘levelisLOW,thecircuitsubtractthenumberinregisterBfromthenumberinregister A.Theoperationis:

WhenADD/SUB‘isa1:

1. AND gates 1,3,5 and 7 are enabled , allowing B0,B1,B2and B3 to pass to the OR gates 9,10,11,12 . AND gates 2,4,6 and 8 are disabled , blocking B0‘,B1‘,B2‘, and B3‘ from reaching the OR gates 9,10,11 and 12.

2. ThetwolevelsB0toB3passthroughtheORgatestothe4-bitparalleladder,tobeadded to the bits A0 to A3. The sum appears at the output S0 toS3

3. Add/SUB‘=1causesnocarryintotheadder.

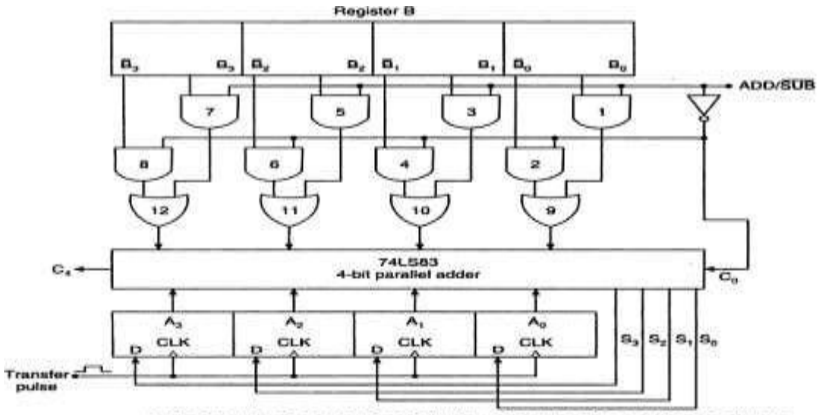
When ADD/SUB‘ is a 0:

1. AND gates 1,3,5 and 7 are disabled , allowing B0,B1,B2and B3 from reaching the OR gates9,10,11,12.ANDgates2,4,6and8areenabled,blockingB0‘,B1‘,B2‘,andB3‘from reaching the OR gates.

2. ThetwolevelsB$_0$'toB$_3$'passthroughtheORgatestothe4-bitparalleladder,tobeadded to the bits A$_0$ to A$_3$.The C$_0$ is now 1.thus the number in register B is converted to its 2's complement form.

3. Thedifference appearsattheoutputS$_0$ toS$_3$.

Adders/Subtractors used for adding and subtracting signed binary numbers. In computers , the output is transferred into the register A (accumulator) so that the result of the addition or subtraction always end up stored in the register A This is accomplished by applying a transfer pulse to the CLK inputs of register A.



Logic diagram of a parallel adder/subtractor using 2's complement system.

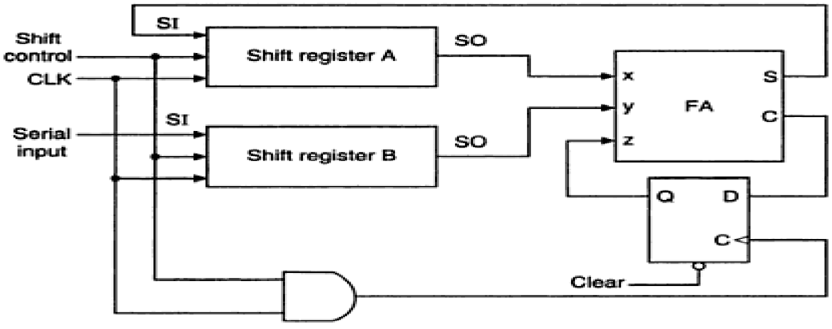
SerialAdder:

A serial adderis used to add binarynumbers in serial form. Thetwo binarynumbers to be addedseriallyarestoredintwoshiftregistersAandB.Bitsareaddedonepair atatimethrough a single full adder (FA) circuit as shown. The carry out of the full-adder is transferred to a D flip-flop.Theoutputofthisflip-flopisthenusedasthecarryinputforthenext pairofsignificant bits. The sum bit from the S output of the full-adder could be transferred to a third shift register. By shifting the sum into A while the bits of A are shifted out, it is possible to use one register for storing both augend and the sum bits. The serial input register B can be used to transfer a new binarynumber while the addend bits are shifted out during theaddition.

Theoperationoftheserialadderis:

Initiallyregister A holds the augend, register B holds the addend and the carryflip-flop is cleared to 0. The outputs (SO) of A and B provide a pair of significant bits for the full-adder at x and y. The shift control enables both registers and carry flip-flop , so, at the clock pulse both registersareshiftedoncetotheright,thesumbitfromSenterstheleftmostflip-flopofA,andthe output carryis transferred into flip-flopQ. Theshift control enablestheregistersforanumberof clock pulses equal to the number of bits of the registers. For each succeeding clock pulse a new sum bit is transferred to A, a new carry is transferred to Q, and both registers are shifted once to theright.Thisprocesscontinuesuntiltheshift controlis disabled.Thustheadditionis accomplished by passing each pair of bits together with the previous carry through a single full adder circuit and transferring the sum, one bitat a time, into registerA.

Initially,registerAandthecarryflip-floparecleared to0andthenthefirstnumberisadded from B. While B is shifted through the full adder, a second number is transferred to it through its serial input. Thesecond numberis then added to the content of register A while a third number is transferredseriallyintoregisterB.Thiscanberepeatedtoformtheadditionoftwo,three,ormore numbers and accumulate their sum in register A.



Logic diagram of a serial adder.

DifferencebetweenSerialandParallelAdders:

Theparalleladderregisterswithparallelload,whereastheserialadderuses shiftregisters. The number of full adder circuits in the parallel adder is equal to the number of bits in the binary numbers, whereas the serial adder requires only one full adder circuit and a carry flip- flop. Excluding the registers, the parallel adder is a combinational circuit, whereas the serial adder is a sequential circuit. The sequential circuit in the serial adder consists of a full-adder and a flip-flop that stores the output carry.

BCDAdder:

TheBCDaddition process:

1. Add the 4-bit BCD code groups for each decimal digit position using ordinary binary addition.

2. For those positions where the sum is 9 or less, the sum is in proper BCD form and no correction is needed.

3. When the sum of two digits is greater than 9, a correction of 0110 should be added to thatsum,toproducetheproperBCDresult.Thiswillproduceacarrytobeaddedtothe next decimalposition.

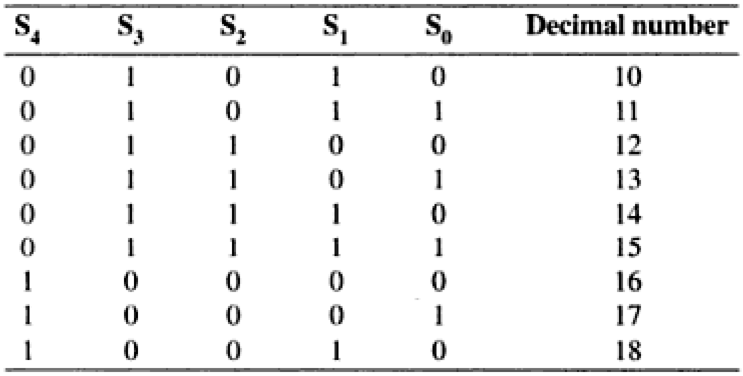
ABCDaddercircuitmustbeabletooperateinaccordancewiththeabovesteps. Inotherwords, the circuit must be able to do the following:

1. Addtwo4-bitBCDcodegroups,usingstraightbinaryaddition.

2. Determine,ifthesumof thisadditionisgreaterthan1101(decimal9);ifitis,add0110 (decimal 6) to this sum and generate a carry to the nextdecimalposition.

The first requirement is easily met by using a 4- bit binary parallel adder such as the 74LS83 IC .For example , if the two BCD code groups A3A2A1A0and B3B2B1B0 are applied to a 4-bit parallel adder,theadder will outputS4S3S2S1S0, whereS4 is actuallyC4 , thecarry–out ofthe MSB bits.

The sum outputs S4S3S2S1S0 can range anywhere from 00000 to 100109when both the BCD code groups are 1001=9). The circuitry for a BCD adder must include the logic needed to detectwheneverthesumisgreaterthan01001,sothatthecorrectioncanbeaddedin.Thosecases ,wherethesumisgreaterthan1001arelistedas:



S_4	S_3	S_2	S_1	S_0	Decimal number
0	1	0	1	0	10
0	1	0	1	1	11
0	1	1	0	0	12
0	1	1	0	1	13
0	1	1	1	0	14
0	1	1	1	1	15
1	0	0	0	0	16
1	0	0	0	1	17
1	0	0	1	0	18

LetusdefinealogicoutputXthatwillgoHIGHonlywhenthesumisgreaterthan01001(i.e, forthecases intable). If examinethesecases,seethatXwillbeHIGHfor eitherofthefollowing conditions:

1. Whenever S4=1(sumgreaterthan15)

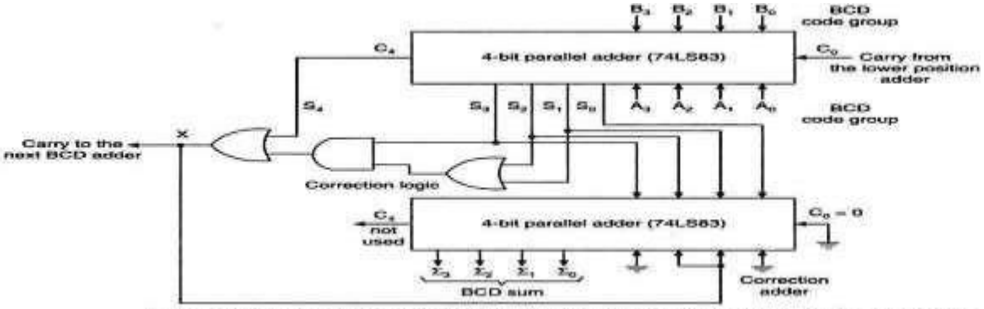
2. WheneverS3=1andeitherS2orS1orbothare1(sum10to 15)

This condition can be expressedas

$$X=S4+S3(S2+S1)$$

Whenever X=1, it is necessary to add the correction factor 0110 to the sum bits, and to generate a carry. The circuit consists of three basic parts. The two BCD code groups A3A2A1A0 andB3B2B1B0areaddedtogetherintheupper4-bitadder,toproducethesumS4S3S2S1S0.The logic gates shown implement the expression for X. The lower 4-bit adder will add the correction 0110tothesumbits,onlywhenX=1,producingthefinalBCDsumoutputrepresentedby ∑3∑2∑1∑0.TheXisalsothecarry-outthatisproducedwhenthesumisgreaterthan01001.When X=0, there is no carryand no addition of 0110. In such cases, ∑3∑2∑1∑0= S3S2S1S0.

Two or more BCD adders can be connected in cascade when two or more digit decimal numbers are to be added. The carry-out of the first BCD adder is connected as the carry-in of the secondBCDadder,thecarry-outofthesecondBCDadderisconnectedasthecarry-inofthethird BCD adder and so on.



Logic diagram of a BCD adder using two 4-bit adders and a correction-detector circuit.

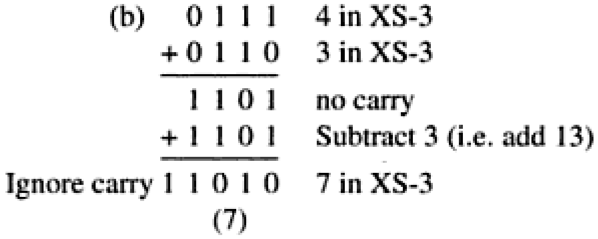
EXCESS-3(XS-3)ADDER:

ToperformExcess-3additions,

1. Addtwoxs-3codegroups
2. If carry=1,add0011(3) tothesumofthosetwocodegroups
   Ifcarry=0,subtract0011(3)i.e.,add1101(13indecimal)tothesumofthosetwocode groups.
   Ex: Add 9 and 5

```
                  1100          9inXs-3
                 +1000                 5in xs-3
                 ——–– – –——
         1        0100                 thereisacarry
+0011             0011                 add3toeachgroup
-----------       -----------
0100              0111                 14in xs-3
  (1)               (4)
```

EX:



```
(b)       0 1 1 1    4 in XS-3
        + 0 1 1 0    3 in XS-3
        ---------
          1 1 0 1    no carry
        + 1 1 0 1    Subtract 3 (i.e. add 13)
        ---------
Ignore carry 1 1 0 1 0    7 in XS-3
              (7)
```

Implementation of xs-3 adder using 4-bit binary adders is shown. The augend (A_3 $A_2A_1A_0$)andaddend($B_3B_2B_1B_0$)inxs-3areaddedusingthe4-bitparalleladder.Ifthecarryis a1, then 0011(3)is added to thesum bits$S_3S_2S_1S_0$ oftheupperadder inthelower4-bit parallel adder.Ifthecarryisa0,then1101(3)isaddedtothesumbits(Thisisequivalenttosubtracting 0011(3) from the sum bits. The correct sum in xs-3 is obtained

Excess-3(XS-3)Subtractor:

ToperformExcess-3subtraction,

1. Complementthesubtrahend
2. Addthecomplementedsubtrahend totheminuend.
3. Ifcarry=1,resultispositive.Add3andendaroundcarrytotheresult.Ifcarry=0,the result isnegative. Subtract 3, i.e,and take the 1‘s complement of theresult.

```
Ex:   Perform9-4
        1100          9inxs-3
       +1000          Complementof4nXs-3
      --------------------
(1)     0100          Thereisacarry
       +0011          Add0011(3)
      --------------------
        0111
           1          Endaroundcarry
      --------------------
        1000          5 in xs-3
```

The minuend and the 1‘s complement of the subtrahend in xs-3 are added in the upper 4-bit parallel adder. If the carry-out from the upper adder is a 0, then 1101 is added to the sum bits oftheupperadderinthelower adderandthesumbitsofthelower adderarecomplementedto get theresult. Ifthecarry-outfromtheupperadderisa1,then3=0011isaddedtothesumbitsofthe lower adder and the sum bits of the lower adder give the result.

BinaryMultipliers:

In binary multiplication by the paper and pencil method, is modified somewhat in digital machines because a binary adder can add only two binary numbers at a time.

In abinarymultiplier,insteadofaddingallthepartialproductsattheend,theyareaddedtwo atatimeandtheirsumaccumulatedinaregister(theaccumulatorregister). Inaddition,whenthe multiplier bit is a 0,0s are not written down and added because it does not affect the final result. Instead, the multiplicand isshifted left by onebit.

Themultiplicationof1110by1001usingthisprocessis

```
Multiplicand 1110
Multiplier          1001
                    1110        TheLSBofthemultiplierisa1;writedownthemultiplicand;
                                shiftthe multiplicand one position to the left(1 1 1 0 0 )
                    1110        The secondmultiplierbit is a0;writedowntheprevious result
                                1110; shiftthe multiplicand to the leftagain (1 1 10
                                0 0)
```

+1110000 The fourth multiplier bit is a 1 write down the new multiplicand add it to the first partial product to obtain the finalproduct.

1111110

This multiplication process can be performed by the serial multiplier circuit , which multiplies two 4-bit numbers to produce an 8-bit product. The circuit consists of following elements

X register: A 4-bit shiftregister that stores the multiplier --- it will shiftright on the fallingedge of the clock. Note that 0s are shifted in from the left.

B register:An8-bitregisterthatstoresthemultiplicand;itwillshiftleftonthefallingedgeofthe clock. Note that 0s are shifted in from theright.

Aregister: An8-bitregister,i.e,theaccumulatorthataccumulatesthepartial products.

Adder: An8-bitparalleladderthatproducesthesumofAandBregisters.TheadderoutputsS7 through S0 are connected to the D inputs of the accumulator so that the sum can be transferred to the accumulator only when a clock pulse gets through the AND gate.

Thecircuitoperationcanbedescribedbygoingthrougheachstepinthemultiplicationof1110by 1001. The complete process requires 4 clock cycles.

1. **Before the first clock pulse***:* Prior to the occurrence of the first clock pulse, the register A is loadedwith00000000,theregisterBwiththemultiplicand00001110,andtheregisterXwiththe multiplier 1001. Assume that each of these registers is loaded using its asynchronous inputs(i.e., PRESET and CLEAR). The output of the adder will bethe sum of A and B,i.e., 00001110.
2. **FirstClockpulse***:*SincetheLSBof themultiplier(X0)isa1,thefirstclockpulsegetsthrough theANDgateanditspositivegoingtransitiontransfersthesumoutputsintotheaccumulator.The subsequent negative going transition causes the X and B registers to shift right and left, respectively. This produces a new sum of A andB.
3. **Second Clock Pulse:** Thesecond bitoftheoriginalmultiplierisnowinX0.Sincethisbitisa 0, the second clock pulse is inhibited from reaching the accumulator. Thus, the sum outputs are not transferred into the accumulator and the number in the accumulator does not change. The negativegoingtransitionoftheclockpulsewillagainshifttheXandBregisters.Againanewsum isproduced.
4. **Third Clock Pulse**:ThethirdbitoftheoriginalmultiplierisnowinX0;sincethisbitisa0,the third clock pulse is inhibited from reaching the accumulator. Thus, the sum outputs are not transferredintotheaccumulatorandthenumberintheaccumulatordoesnotchange.Thenegative going transition of the clock pulse will again shift the X and B registers. Again a new sum is produced.
5. **Fourth Clock Pulse:** ThelastbitoftheoriginalmultiplierisnowinX0,andsinceitisa1,the positive going transition of the fourth pulse transfers the sum into the accumulator. The accumulatornowholdsthefinalproduct.Thenegativegoingtransitionof theclockpulseshiftsX and B again. Note that, X isnow 0000, sinceall the multiplier bits have been shifted out.

Codeconverters:

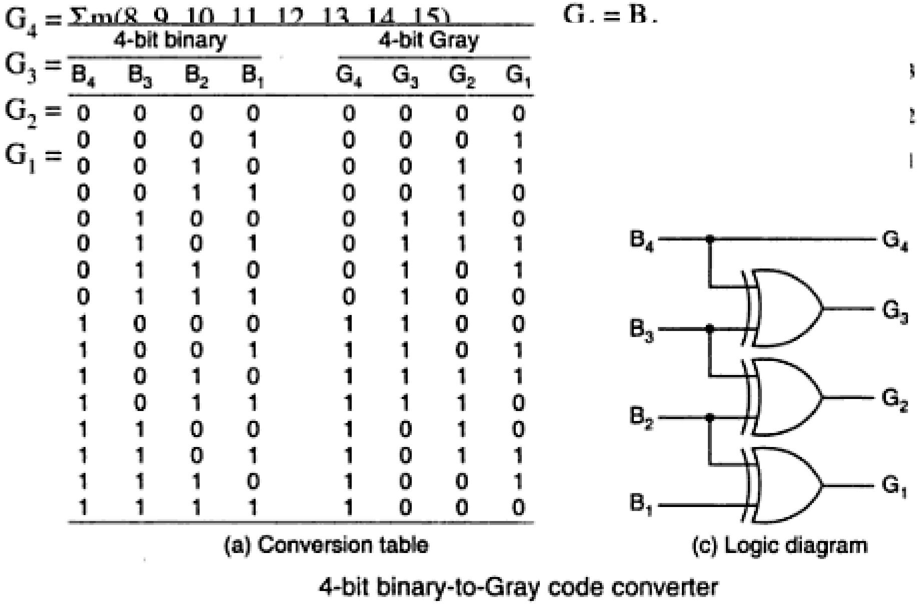
Theavailabilityofalargevarietyofcodesforthesamediscreteelementsofinformation resultsintheuseofdifferentcodesbydifferentdigitalsystems.Itissometimesnecessarytouse the output of one system as the input to another. A conversion circuit must be inserted between the two systems if each uses different codes for the same information. Thus a code converter is a logic circuit whose inputs are bit patterns representing numbers (or character) in one cod and whose outputs are the corresponding representation in a different code. Code converters are usually multiple output circuits.

To convert from binary code A to binary code B, the input lines must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B. A combinational circuit performs this transformation by means of logic gates.

For example, a binary –to-gray code converter has four binaryinput lines B4, B3,B2,B1 and fourgraycodeoutputlinesG4,G3,G2,G1.Whentheinputis0010,forinstance,theoutputshould be0011andsoforth.Todesign acodeconverter,weuseacodetabletreatingitasatruthtableto express each output as a Boolean algebraic function of all the inputs. Inthisexample,ofbinary–to-graycodeconversion,wecantreatthebinarytothe graycodetable as four truth tables to derive expressions for G4, G3, G2, and G1. Each of these four expressions would,ingeneral,containallthefourinputvariablesB4,B3,B2,andB1.Thus,thiscodeconverter is actually equivalent to four logic circuits, one for each of the truth tables.

The logic expression derived for the code converter can be simplified using the usual techniques, including_don‘tcares‘ifpresent.Eveniftheinputisanunweightedcode,thesamecellnumbering method which we used earlier can be used, but the cell numbers --must correspond to the input combinations as if they were an 8-4-2-1 weighted code.

Designofa4-bitbinarytograycodeconverter:



$G_4 = \Sigma m(8\ 9\ 10\ 11\ 12\ 13\ 14\ 15)$ $G. = B.$

$G_3 =$

$G_2 =$

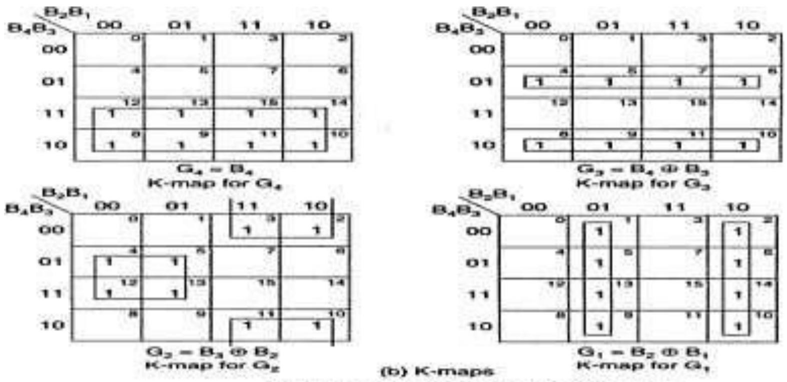
$G_1 =$

4-bit binary				4-bit Gray			
B_4	B_3	B_2	B_1	G_4	G_3	G_2	G_1
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	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

(a) Conversion table

(c) Logic diagram

4-bit binary-to-Gray code converter



$G_4 = B_4$ K-map for G_4

$G_3 = B_4 \oplus B_3$ K-map for G_3

$G_2 = B_3 \oplus B_2$ K-map for G_2

(b) K-maps

$G_1 = B_2 \oplus B_1$ K-map for G_1

4-bit binary-to-Gray code converter.

Designofa4-bitgraytoBinarycodeconverter:

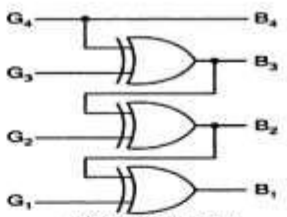
$$B_4 = \Sigma\, m(12, 13, 15, 14, 10, 11, 9, 8) = \Sigma\, m(8, 9, 10, 11, 12, 13, 14, 15)$$
$$B_3 = \Sigma\, m(6, 7, 5, 4, 10, 11, 9, 8) = \Sigma\, m(4, 5, 6, 7, 8, 9, 10, 11)$$
$$B_2 = \Sigma\, m(3, 2, 5, 4, 15, 14, 9, 8) = \Sigma\, m(2, 3, 4, 5, 8, 9, 14, 15)$$
$$B_1 = \Sigma\, m(1, 2, 7, 4, 13, 14, 11, 8) = \Sigma\, m(1, 2, 4, 7, 8, 11, 13, 14)$$

$$B_4 = G_4$$
$$B_3 = \bar{G}_4 G_3 + G_4 \bar{G}_3 = G_4 \oplus G_3$$
$$B_2 = \bar{G}_4 G_3 \bar{G}_2 + \bar{G}_4 \bar{G}_3 G_2 + G_4 \bar{G}_3 \bar{G}_2 + G_4 G_3 G_2$$
$$= \bar{G}_4 (G_3 \oplus G_2) + G_4 (\overline{G_3 \oplus G_2}) = G_4 \oplus G_3 \oplus G_2 = B_3 \oplus G_2$$
$$B_1 = \bar{G}_4 \bar{G}_3 \bar{G}_2 G_1 + \bar{G}_4 \bar{G}_3 G_2 \bar{G}_1 + \bar{G}_4 G_3 G_2 G_1 + \bar{G}_4 G_3 \bar{G}_2 \bar{G}_1 + G_4 G_3 \bar{G}_2 G_1 + G_4 G_3 G_2 \bar{G}_1 + G_4 \bar{G}_3 G_2 G_1 + G_4 \bar{G}_3 \bar{G}_2 \bar{G}_1$$

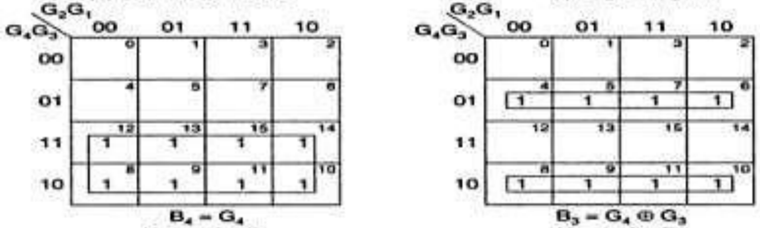
$$= \bar{G}_4 \bar{G}_3 (G_2 \oplus G_1) + G_4 G_3 (G_2 \oplus G_1) + \bar{G}_4 G_3 (\overline{G_2 \oplus G_1}) + G_4 \bar{G}_3 (\overline{G_2 \oplus G_1})$$
$$= (G_2 \oplus G_1)(\overline{G_4 \oplus G_3}) + (\overline{G_2 \oplus G_1})(G_4 \oplus G_3)$$
$$= G_4 \oplus G_3 \oplus G_2 \oplus G_1$$

4-bit Gray				4-bit binary			
G_4	G_3	G_2	G_1	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

(a) Conversion table

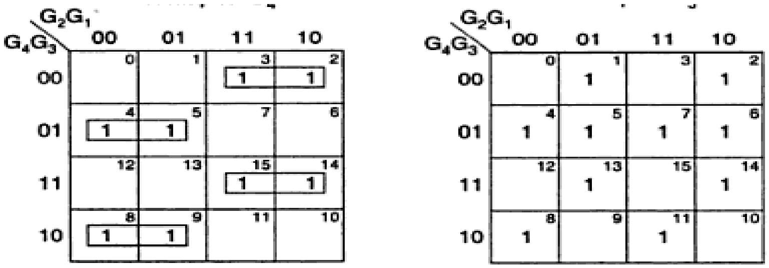


(c) Logic diagram



$B_4 = G_4$ K-map for B_4

$B_3 = G_4 \oplus G_3$ K-map for B_3



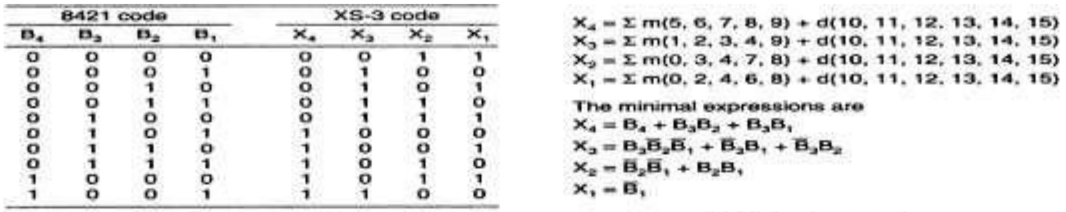
$B_2 = G_4 \oplus G_3 \oplus G_2$
K-map for B_2

$B_1 = G_4 \oplus G_3 \oplus G_2 \oplus G_1$
K-map for B_1

(b) K-maps

4-bit Gray-to-binary code converter.

Designofa4-bitBCDtoXS-3 codeconverter:



8421 code				XS-3 code			
B_4	B_3	B_2	B_1	X_4	X_3	X_2	X_1
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

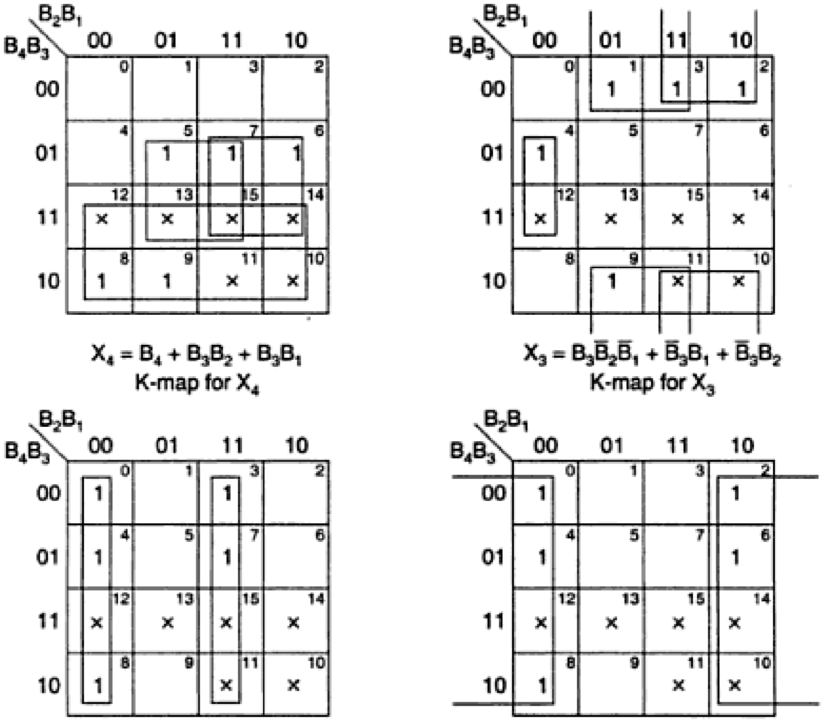
(a) Conversion table

$X_4 = \Sigma m(5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$
$X_3 = \Sigma m(1, 2, 3, 4, 9) + d(10, 11, 12, 13, 14, 15)$
$X_2 = \Sigma m(0, 3, 4, 7, 8) + d(10, 11, 12, 13, 14, 15)$
$X_1 = \Sigma m(0, 2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$

The minimal expressions are
$X_4 = B_4 + B_3B_2 + B_3B_1$
$X_3 = B_3\bar{B}_2\bar{B}_1 + \bar{B}_3B_1 + \bar{B}_3B_2$
$X_2 = \bar{B}_2\bar{B}_1 + B_2B_1$
$X_1 = \bar{B}_1$

(b) Minimal expressions

4-bit BCD-to-XS-3 code converter



$X_4 = B_4 + B_3B_2 + B_3B_1$
K-map for X_4

$X_3 = B_3\bar{B}_2\bar{B}_1 + \bar{B}_3B_1 + \bar{B}_3B_2$
K-map for X_3

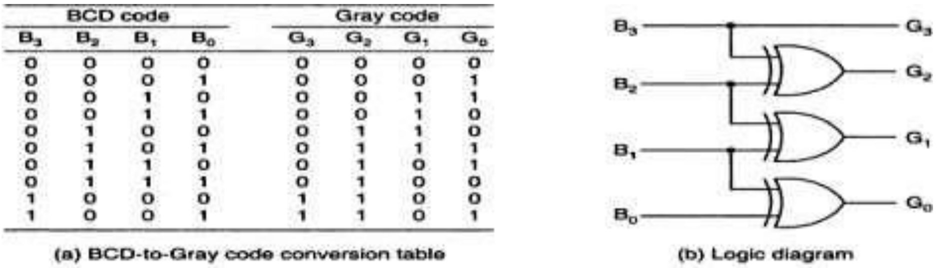
$X_2 = \bar{B}_2\bar{B}_1 + B_2B_1$
K-map for X_2

$X_1 = \bar{B}_1$
K-map for X_1

(c) K-maps

4-bit BCD-to-XS-3 code converter.

Design of aBCDtograycodeconverter:

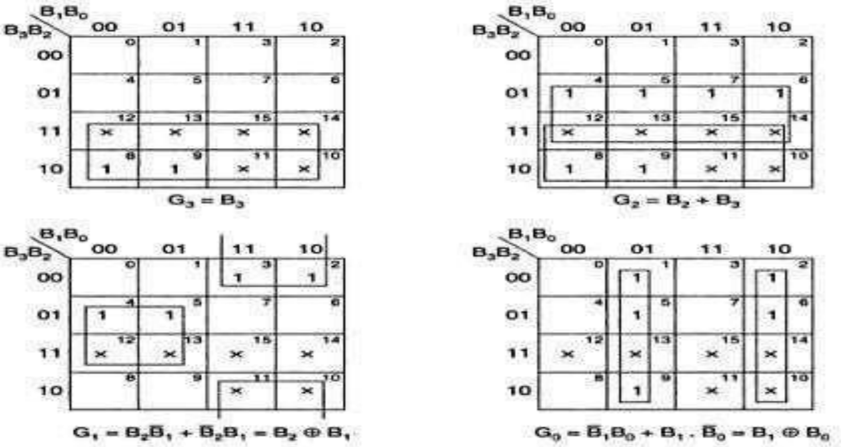


BCD code				Gray code			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1

(a) BCD-to-Gray code conversion table

(b) Logic diagram

BCD-to-Gray code converter.



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

$G_3 = B_3$

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

$G_2 = B_2 + B_3$

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

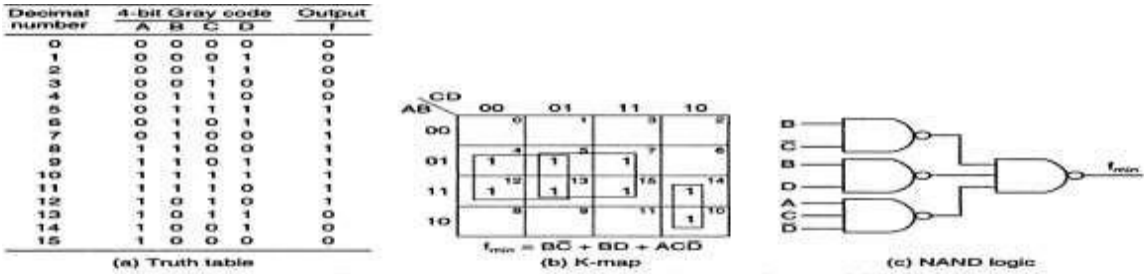
$G_1 = B_2\bar{B}_1 + \bar{B}_2B_1 = B_2 \oplus B_1$

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

$G_0 = \bar{B}_1B_0 + B_1 . \bar{B}_0 = B_1 \oplus B_0$

K-maps for a BCD-to-Gray code converter.

Designof aSOPcircuittoDetecttheDecimalnumbers5through12ina4-bitgraycode Input:



Decimal number	4-bit Gray code				Output
	A	B	C	D	f
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	1	0
3	0	0	1	0	0
4	0	1	1	0	0
5	0	1	1	1	1
6	0	1	0	1	1
7	0	1	0	0	1
8	1	1	0	0	1
9	1	1	0	1	1
10	1	1	1	1	1
11	1	1	1	0	1
12	1	0	1	0	1
13	1	0	1	1	0
14	1	0	0	1	0
15	1	0	0	0	0

(a) Truth table

$AB \backslash CD$	00	01	11	10
00				
01	1	1	1	
11	1	1	1	1
10				1

$f_{min} = B\bar{C} + BD + AC\bar{D}$

(b) K-map

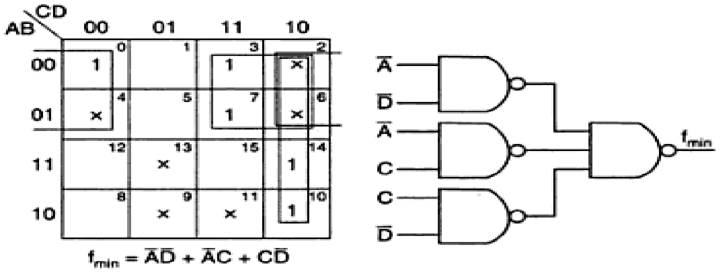
(c) NAND logic

Truth table, K-map and logic diagram for the SOP circuit.

DesignofaSOPcircuittodetectthedecimalnumbers0,2,4,6,8ina4-bit5211BCDcode input:

Decimal number	A	B	C	D	Output f
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	1	1
3	0	1	0	1	0
4	0	1	1	1	1
5	1	0	0	0	0
6	1	0	1	0	1
7	1	1	0	0	0
8	1	1	1	0	1
9	1	1	1	1	0

(a) Truth table



(b) K-map (c) Logic diagram

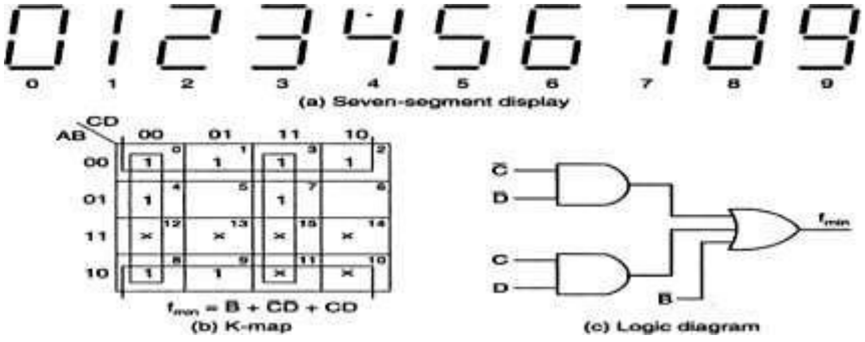
Truth table, K-map and logic diagram for the SOP circuit.

DesignofaCombinationalcircuittoproducethe2'scomplementofa4-bitbinarynumber:

Input				Output			
A	B	C	D	E	F	G	H
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

(a) Conversion table

Conversion table and K-maps for the circuit



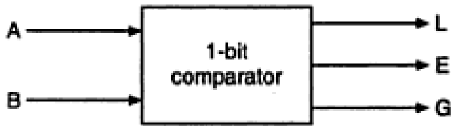
(a) Seven-segment display

(b) K-map

(c) Logic diagram

Comparators:

$$\text{EQUALITY} = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$



Block diagram of a 1-bit comparator.

The logic for a 1-bit magnitude comparator: Let the 1-bit numbers be $A = A_0$ and $B = B_0$.
If $A_0 = 1$ and $B_0 = 0$, then $A > B$.
Therefore,

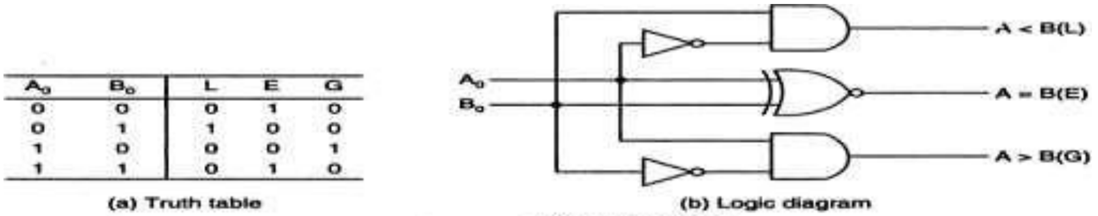
$$A > B: G = A_0\overline{B}_0$$

If $A_0 = 0$ and $B_0 = 1$, then $A < B$.
Therefore,

$$A < B: L = \overline{A}_0B_0$$

If A_0 and B_0 coincide, i.e. $A_0 = B_0 = 0$ or if $A_0 = B_0 = 1$, then $A = B$.
Therefore,

$$A = B : E = A_0 \odot B_0$$



A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

(a) Truth table

(b) Logic diagram

1-bit comparator.

1.MagnitudeComparator:

1-bitMagnitudeComparator:

The logic for a 2-bit magnitude comparator: Let the two 2-bit numbers be $A = A_1 A_0$ and $B = B_1 B_0$.

1. If $A_1 = 1$ and $B_1 = 0$, then $A > B$ or
2. If A_1 and B_1 coincide and $A_0 = 1$ and $B_0 = 0$, then $A > B$. So the logic expression for $A > B$ is

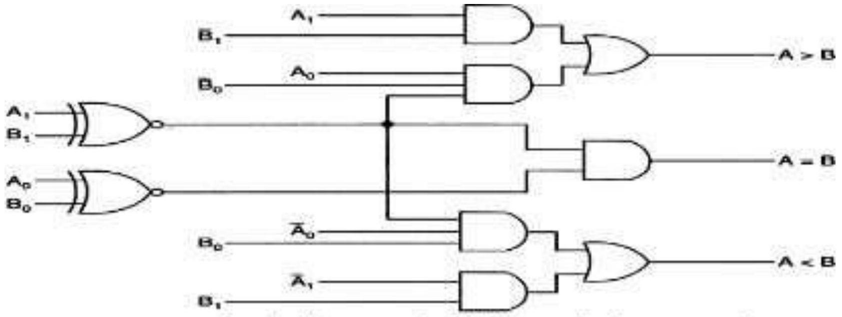
$$A > B : G = A_1\overline{B}_1 + (A_1 \odot B_1)A_0\overline{B}_0$$

1. If $A_1 = 0$ and $B_1 = 1$, then $A < B$ or
2. If A_1 and B_1 coincide and $A_0 = 0$ and $B_0 =1$, then $A< B$. So the expression for $A < B$ is

$$A < B : L = \overline{A}_1B_1 + (A_1 \odot B_1)\overline{A}_0B_0$$

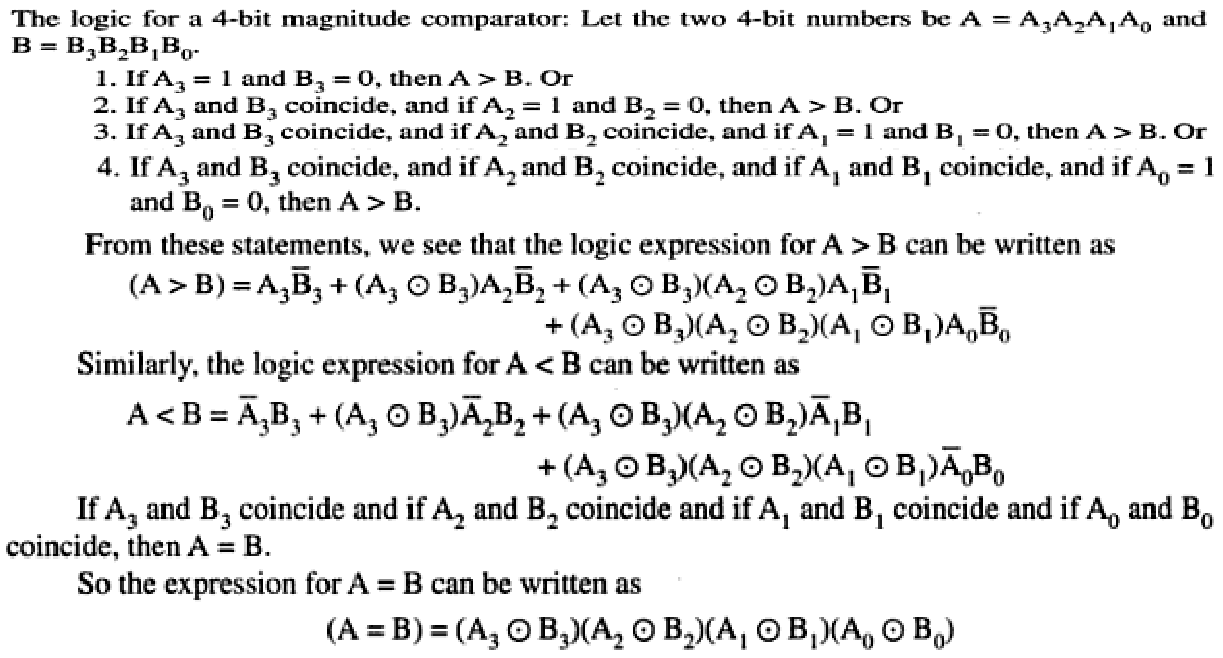
If A_1 and B_1 coincide and if A_0 and B_0 coincide then $A = B$. So the expression for $A = B$ is

$$A = B : E = (A_1 \odot B_1)(A_0 \odot B_0)$$



Logic diagram of a 2-bit magnitude comparator.

4-BitMagnitudeComparator:



The logic for a 4-bit magnitude comparator: Let the two 4-bit numbers be $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$.

1. If $A_3 = 1$ and $B_3 = 0$, then $A > B$. Or
2. If A_3 and B_3 coincide, and if $A_2 = 1$ and $B_2 = 0$, then $A > B$. Or
3. If A_3 and B_3 coincide, and if A_2 and B_2 coincide, and if $A_1 = 1$ and $B_1 = 0$, then $A > B$. Or
4. If A_3 and B_3 coincide, and if A_2 and B_2 coincide, and if A_1 and B_1 coincide, and if $A_0 = 1$ and $B_0 = 0$, then $A > B$.

From these statements, we see that the logic expression for $A > B$ can be written as

$$(A > B) = A_3\overline{B}_3 + (A_3 \odot B_3)A_2\overline{B}_2 + (A_3 \odot B_3)(A_2 \odot B_2)A_1\overline{B}_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)A_0\overline{B}_0$$

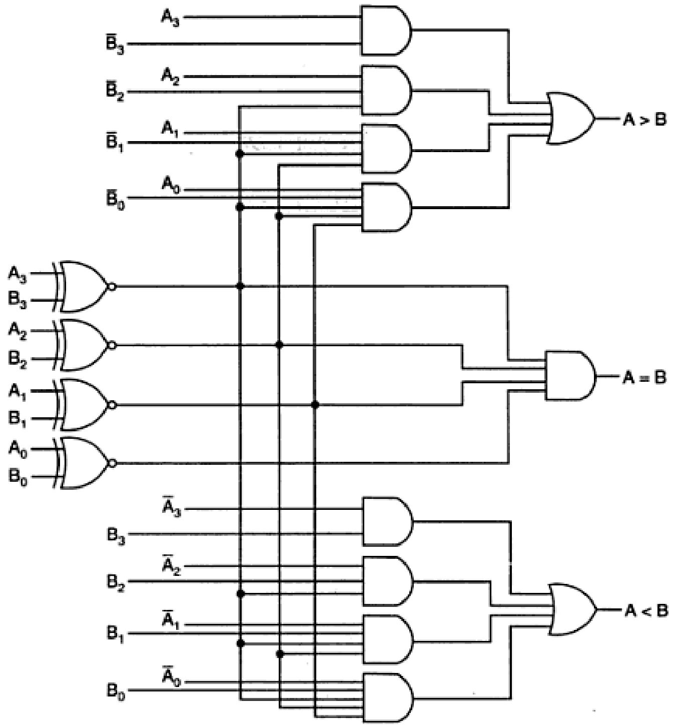
Similarly, the logic expression for $A < B$ can be written as

$$A < B = \overline{A}_3B_3 + (A_3 \odot B_3)\overline{A}_2B_2 + (A_3 \odot B_3)(A_2 \odot B_2)\overline{A}_1B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\overline{A}_0B_0$$

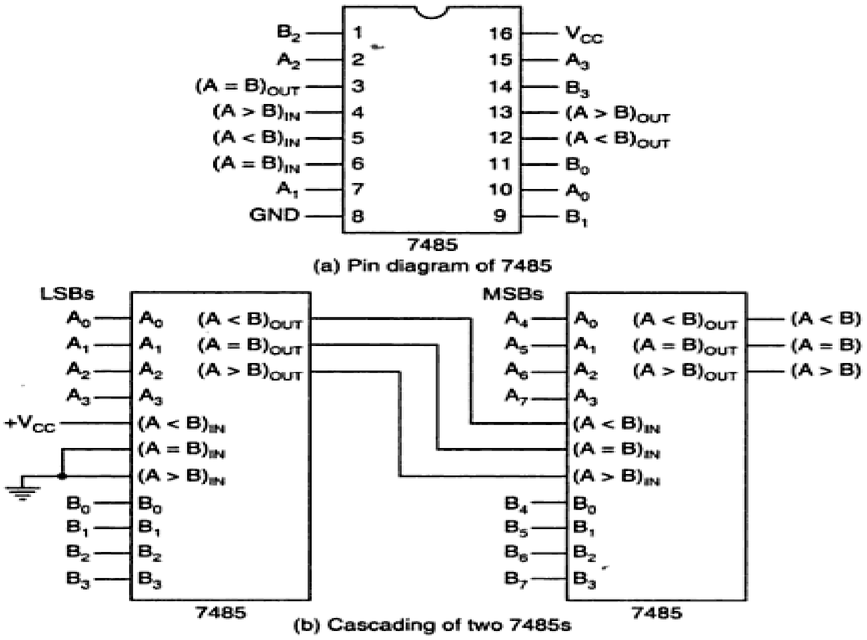
If A_3 and B_3 coincide and if A_2 and B_2 coincide and if A_1 and B_1 coincide and if A_0 and B_0 coincide, then $A = B$.

So the expression for $A = B$ can be written as

$$(A = B) = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$



ICComparator:

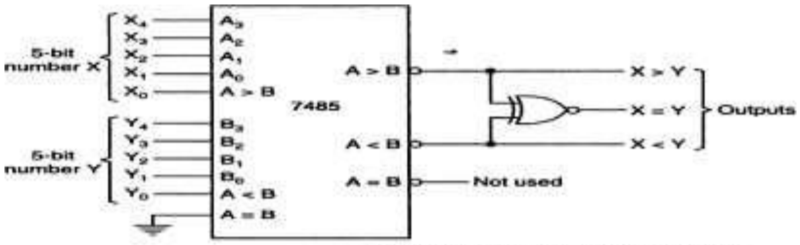


(a) Pin diagram of 7485

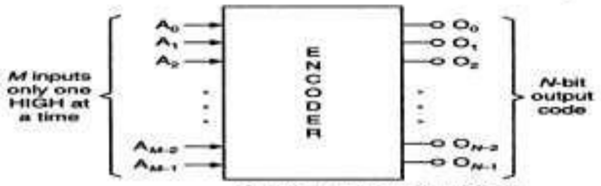
(b) Cascading of two 7485s

Pin diagram and cascading of 7485 4-bit comparators.

ENCODERS:

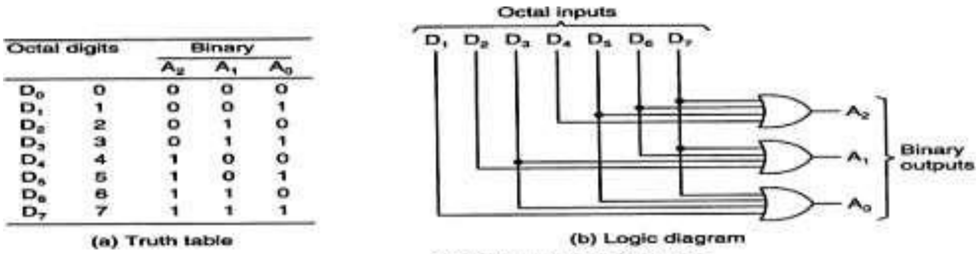


Use of 7485 as a 5-bit comparator.



Block diagram of encoder.

OctaltoBinaryEncoder:



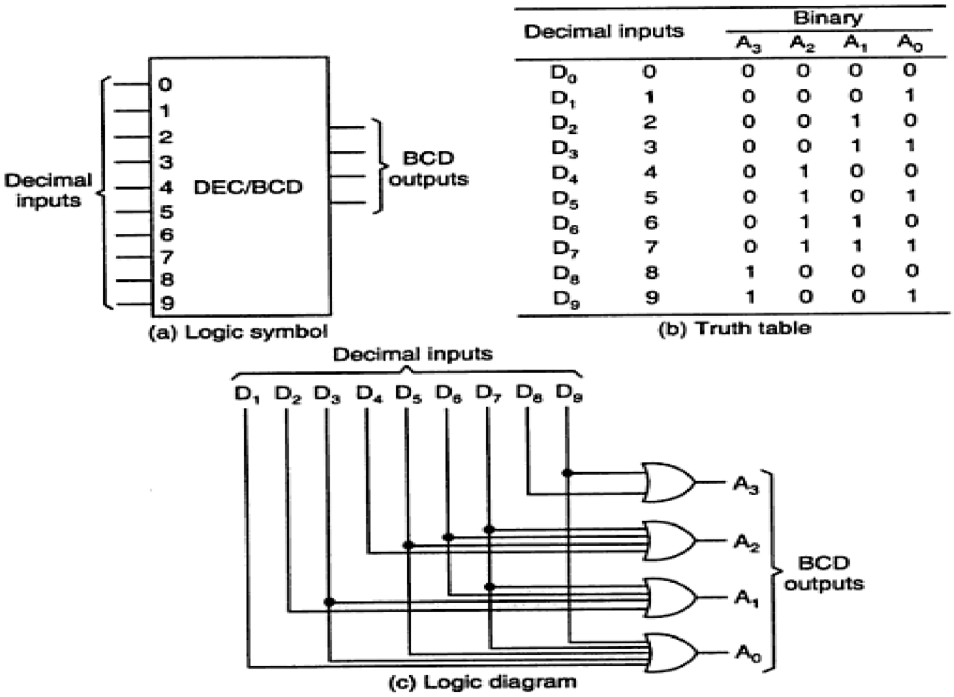
Octal digits		Binary		
		A_2	A_1	A_0
D_0	0	0	0	0
D_1	1	0	0	1
D_2	2	0	1	0
D_3	3	0	1	1
D_4	4	1	0	0
D_5	5	1	0	1
D_6	6	1	1	0
D_7	7	1	1	1

(a) Truth table

(b) Logic diagram

Octal-to-binary encoder.

DecimaltoBCDEncoder:



(a) Logic symbol

Decimal inputs		Binary			
		A_3	A_2	A_1	A_0
D_0	0	0	0	0	0
D_1	1	0	0	0	1
D_2	2	0	0	1	0
D_3	3	0	0	1	1
D_4	4	0	1	0	0
D_5	5	0	1	0	1
D_6	6	0	1	1	0
D_7	7	0	1	1	1
D_8	8	1	0	0	0
D_9	9	1	0	0	1

(b) Truth table

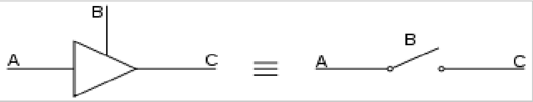
(c) Logic diagram

Decimal-to-BCD encoder.

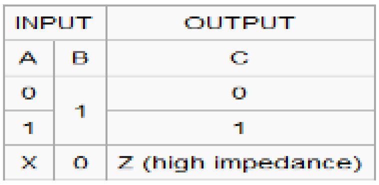
Tristatebus system:

In digital electronics**three-state**, **tri-state**, or **3-state**logic allows an output port to assume a highimpedancestateinadditiontothe0and1logiclevels,effectivelyremovingtheoutputfromthecircuit.This allows multiple circuits to share the same output line or lines (such as a bus which cannot listen to more than one device at a time).

Three-state outputs are implemented in many registers,bus drivers, and flip-flops in the 7400 and 4000 seriesaswellasinothertypes,butalsointernallyinmanyintegratedcircuits.Othertypicalusesareinternal andexternalbusesinmicroprocessors,computermemory, andperipherals. Manydevicesarecontrolledby anactive-lowinputcalledOE(OutputEnable)whichdictateswhethertheoutputsshouldbeheldinahigh-impedance state or drive their respective loads (to either 0- or 1-level).



A tristate buffer can be thought of as a switch. If *B* is on, the switch is closed. If B is off, the switch is open.



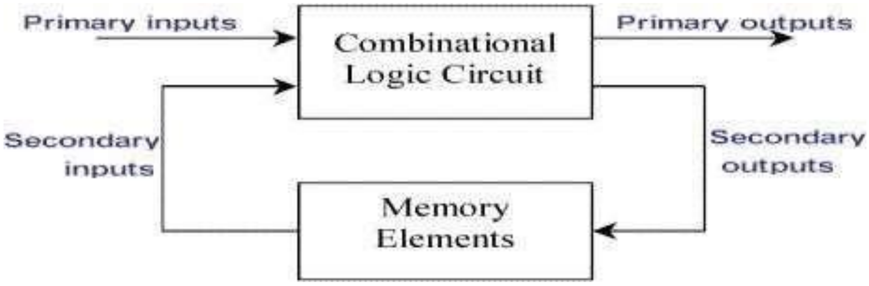
INPUT		OUTPUT
A	B	C
0	1	0
1		1
X	0	Z (high impedance)

UNIT III
SEQUENTIALCIRCUITS

Classificationofsequentialcircuits: Sequentialcircuitsmaybeclassifiedastwotypes.

1. Synchronoussequentialcircuits
2. Asynchronoussequentialcircuits

Combinational logic refers to circuits whose output is strictly depended on the present value of theinputs.Assoonasinputsarechanged,theinformationaboutthepreviousinputsislost,thatis, combinational logics circuits have no memory. Although every digital system is likely to have combinationalcircuits,mostsystemsencounteredinpracticealsoincludememoryelements,which require that the system be described in terms of sequential logic. Circuits whose output depends not only on the present input value but also the past input value are known as **sequential logic circuits**. The mathematical model of a sequential circuit is usually referred to as a **sequential machine**.



Comparisonbetweencombinationalandsequentialcircuits

Combinationalcircuit	**Sequentialcircuit**
1.Incombinationalcircuits,the output variables at any instant of time are dependent only on the present input Variables	1.insequentialcircuitstheoutputvariablesat anyinstant of time are dependent not onlyon the present input variables, but also on the present state
2.memoryunitisnotrequiresin combinational circuit	2.memoryunitisrequiredtostorethepast history of the input variables
3.thesecircuitsarefasterbecause the delaybetween the i/p and o/p	3. sequential circuits are slower than combinationalcircuitsduetopropagation delay of gates only
4. easytodesign	4.comparativelyhardtodesign

Levelmodeandpulsemodeasynchronoussequentialcircuits:

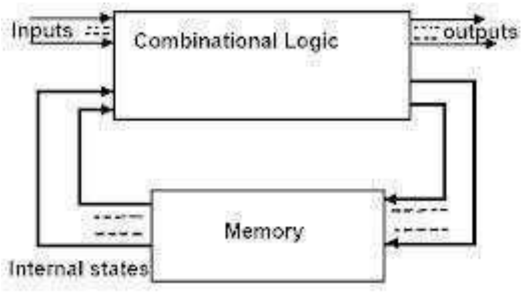


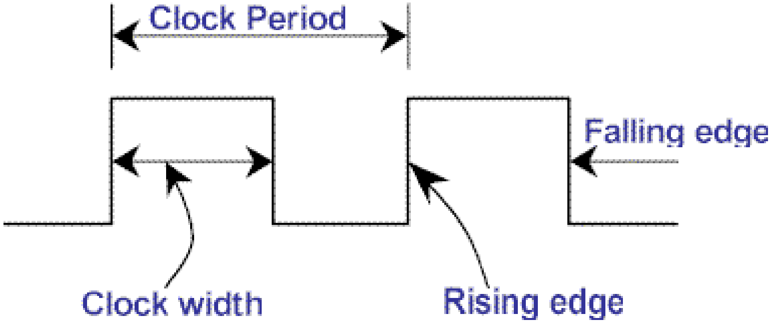
Figure 1: Asynchronous Sequential Circuit

Fig shows a block diagram of an asynchronous sequential circuit. It consists of a combinational circuit and delay elements connected to form the feedbackloops. The present state and next state variables in asynchronous sequential circuits called secondary variables and excitation variables respectively..

Therearetwotypesofasynchronouscircuits:fundamentalmodecircuitsandpulsemode circuits.

SynchronousandAsynchronousOperation:

Sequentialcircuitsaredividedintotwomaintypes:**synchronous**and**asynchronous**.Their classificationdependson thetimingof theirsignals.***Synchronous*** sequentialcircuitschangetheir states and output values at discrete instants of time, which are specified by the rising and falling edge of a free-running **clock signal**. The clock signal is generally some form of square wave as shown in Figure below.



From the diagram you can see that the **clock period** is the time between successive transitionsinthesamedirection,thatis,betweentworisingortwofallingedges.Statetransitions in synchronous sequential circuits are made to take place at times when the clock is making a transitionfrom0to1(risingedge)orfrom 1to0(fallingedge).Betweensuccessiveclockpulses thereisnochangeinthe information storedinmemory.

The reciprocal of the clock period is referred to as the **clock frequency**. The **clock width** isdefinedasthetimeduringwhichthevalueoftheclocksignalisequalto1.Theratiooftheclock width and clock period is referred to as the duty cycle. A clock signal is said to be **active high** if thestatechangesoccurattheclock'srisingedgeorduringtheclockwidth.Otherwise,theclockis said to be **active low**. Synchronous sequential circuits are also known as **clocked sequential circuits**.

Thememoryelementsusedinsynchronoussequentialcircuitsareusuallyflip-flops.These circuits are binary cells capable of storing one bit of information. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the bit stored in it. Binary information can enter a flip-flop in a varietyof ways, a fact which give rise to the different types of flip-flops. For information on the different types of basic flip-flop circuits and their logical properties, see the previous tutorial on flip-flops.

In ***asynchronous*** sequential circuits, the transition from one state to another is initiated by the changeintheprimaryinputs;thereisnoexternalsynchronization.Thememorycommonlyusedin asynchronous sequential circuits are time-delayed devices, usually implemented by feedback among logic gates. Thus, asynchronous sequential circuits may be regarded as combinational circuits with feedback. Because of the feedback among logic gates, asynchronous sequential circuits may, at times, become unstable due to transient conditions. The instability problem imposesmanydifficultiesonthedesigner.Hence, theyarenot ascommonlyused assynchronous systems.

FundamentalModeCircuitsassumesthat:

1. Theinputvariableschangeonlywhenthe circuitisstable
2. Onlyoneinput variablecanchangeatagiventime
3. Inputsarelevelsarenotpulses

Apulsemodecircuitassumesthat:

1. Theinputvariablesarepulsesinsteadoflevels
2. Thewidthofthepulsesislongenoughforthecircuit torespondtotheinput
3. Thepulsewidthmustnotbesolongthatisstillpresentafterthenewstateisreached.

Latchesandflip-flops

Latchesandflip-flopsarethebasicelementsforstoringinformation.Onelatchorflip-flop can store one bit of information. The main difference between latches and flip-flops is that for latches,theiroutputsareconstantlyaffectedbytheirinputsaslongastheenablesignalisasserted. In other words, when they are enabled, their content changes immediately when their inputs change. Flip-flops,on theotherhand,havetheircontentchangeonlyeitheratherisingorfalling edgeoftheenablesignal.Thisenablesignalisusuallythecontrollingclocksignal.Aftertherising or falling edge of the clock, the flip-flop content remains constant even if the input changes.

Therearebasicallyfour main types oflatches and flip-flops: SR,D,JK,and T.Themajor differences in theseflip-flop types arethenumber of inputs theyhave and how theychange state.

For each type, there are also different variations that enhance their operations. In this chapter, we will look at the operations of the various latches and flip-flops.the flip-flops has two outputs, labeled Q and Q‘. the Q output is thenormal output of the flip flop and Q‘ is the inverted output.

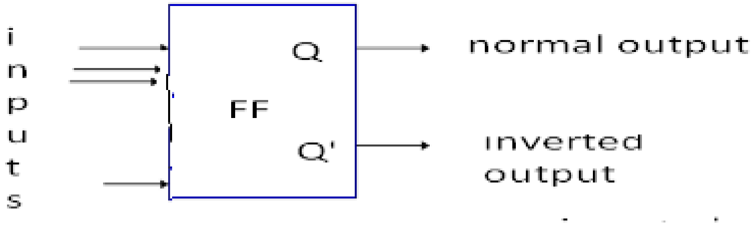


Figure:basicsymbolofflipflop

A latch may be an active-high input latch or an active –LOW input latch.active –HIGH means that the SET and RESET inputs arenormallyrestingin the low state and one of them will be pulsed high whenever we want to change latch outputs.

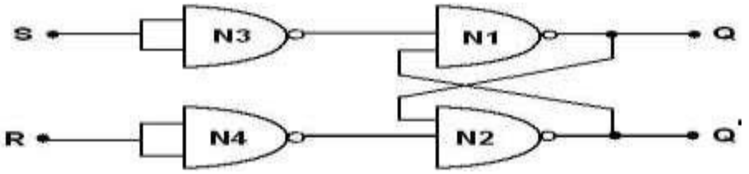
SRlatch:

The latch has two outputs Q and Q‘. When the circuit is switched on the latch may enter into any state. If Q=1, then Q‘=0, which is called SET state. If Q=0, then Q‘=1, which is called RESET state. Whether the latch is in SET state or RESET state, it will continue to remain in the same state, as long as the power is not switched off. But the latch is not an useful circuit, since thereis no wayof enteringthedesired input. It is thefundamental buildingblockin constructing flip-flops, as explained in the following sections

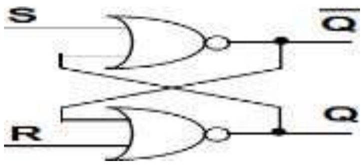
NANDlatch

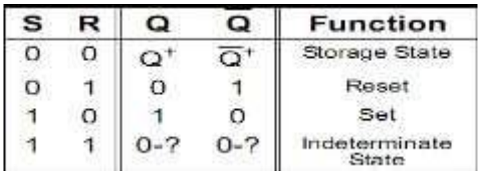
NAND latch is the fundamental building block in constructing a flip-flop. It has the propertyof holding on to any previous output, as long as it is not disturbed.

TheoprationofNANDlatchisthereverseoftheoperationofNORlatch.if0‘sarereplaced by1‘s and 1‘s are replaced by0‘s we get the same truth table as that of the NOR latch shown



NORlatch





S	R	Q	$\overline{Q}$	Function
0	0	Q^+	$\overline{Q}^+$	Storage State
0	1	0	1	Reset
1	0	1	0	Set
1	1	0-?	0-?	Indeterminate State

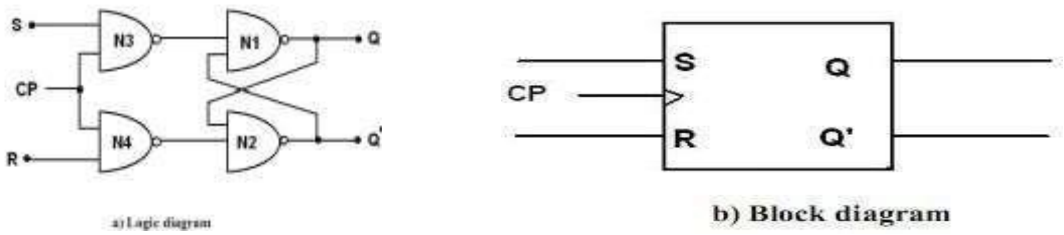
Theanalysisoftheoperationoftheactive-HIGHNORlatchcanbesummarizedasfollows.

1. SET=0, RESET=0: this is normal resting state of the NOR latch and it has no effect on the output state. Q and Q‘ will remain in whatever stste theywere prior to the occurrence of this inputcondition.
2. SET=1,RESET=0:thiswillalwayssetQ=1,whereitwillremainevenafterSETreturnsto0
3. SET=0,RESET=1:thiswillalwaysresetQ=0,whereit willremainevenafterRESET returns to 0
4. SET=1,RESET=1; this condition tries to SET and RESET the latch at the same time, and it producesQ=Q‘=0. Ifthe inputsarereturnedtozerosimultaneously,theresultingoutputstste is erratic and unpredictable. This input condition should not beused.

TheSETandRESETinputsarenormallyinthe LOWstateandoneofthemwillbepulsed HIGH. Whenever we want to change the latch outputs..

RSFlip-flop:

The basic flip-flop is a one bit memory cell that gives the fundamental idea of memory device.ItconstructedusingtwoNANDgates.ThetwoNANDgatesN1andN2areconnectedsuch that, output of N1 is connected to input of N2 and output of N2 to input of N1. These form the feedback path the inputs are S and R, and outputs are Q and Q‘. The logic diagram and the block diagram of R-S flip-flop with clocked input



a) Logic diagram

b) Block diagram

Figure:RSFlip-flop

Theflip-flop can be made to respond onlyduring the occurrence of clock pulse byadding twoNANDgatestotheinputlatch.Sosynchronizationisachieved.i.e.,flip-flopsareallowed to change their states only at particular instant of time. The clock pulses are generated by a clock pulse generator. The flip-flops are affected only with the arrival of clock pulse**.**

Operation:

1. WhenCP=0theoutputofN3andN4are1regardlessofthevalueofSandR.Thisisgiven as input to N1 and N2. Thismakes the previous value of Q andQ‘unchanged.

2. WhenCP=1theinformationatSandRinputsareallowedtoreachthelatchandchangeof state in flip-flop takes place.

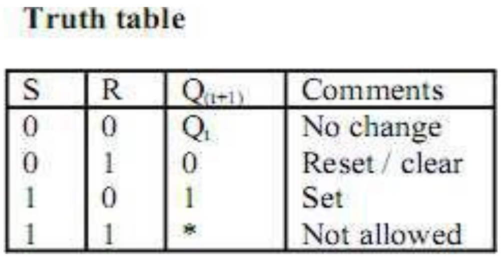
3. CP=1,S=1,R=0givestheSETstatei.e.,Q=1,Q‘=0.

4. CP=1,S=0,R=1givestheRESETstatei.e.,Q=0,Q‘=1.

5. CP=1,S=0,R=0doesnotaffectthestateofflip-flop.

6. CP=1,S=1,R=1isnotallowed,becauseitisnotabletodeterminethenextstate.This condit ion is said to be a ―race condition.

In the logic symbol CP input is marked with a triangle. It indicates the circuit responds to an input change from 0 to 1. The characteristic table gives the operation conditions of flip-flop. Q(t) is the present state maintained in the flip-flop at time ‗t‘. Q(t+1) is the state after the occurrence of clock pulse.

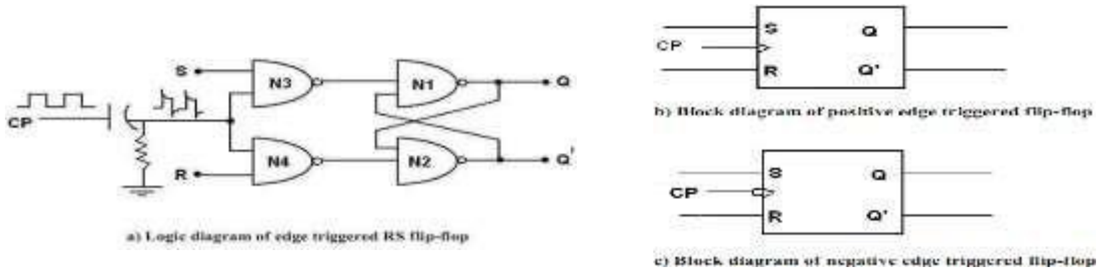


Truth table

S	R	$Q_{(t+1)}$	Comments
0	0	Q_t	No change
0	1	0	Reset / clear
1	0	1	Set
1	1	*	Not allowed

EdgetriggeredRSflip-flop:

Some flip-flops have an RC circuit at the input next to the clock pulse. By the design of the circuit the R-C time constant is much smaller than the width of the clock pulse. So the output changes will occur only at specific level of clock pulse. The capacitor gets fully charged when clock pulse goes from low to high. This change produces a narrow positive spike. Later at the trailing edge it produces narrow negative spike. This operation is called edge triggering, as the flip-flop responds only at the changing state of clock pulse. If output transition occurs at rising edge of clock pulse (0⊓1),it is called positivelyedge triggering. If it occurs at trailing edge (1 ┌ 0)itiscallednegativeedgetriggering.Figureshowsthelogicandblockdiagram.



a) Logic diagram of edge triggered RS flip-flop

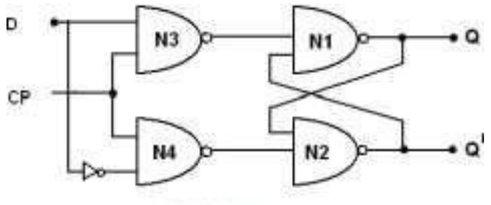
b) Block diagram of positive edge triggered flip-flop

c) Block diagram of negative edge triggered flip-flop

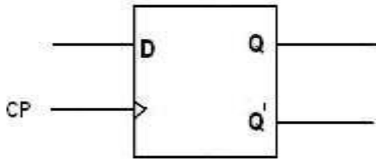
Figure:EdgetriggeredRSflip-flop

Dflip-flop:

The D flip-flop is the modified form of R-S flip-flop. R-S flip-flop is converted to D flip-flop by addinganinverterbetweenSandRandonlyoneinputDistakeninsteadofSandR.Sooneinput is D and complement of D is given as another input. The logic diagram and the block diagram of D flip-flop with clocked input



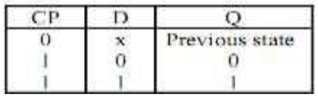
a) Logic diagram



b) Block diagram

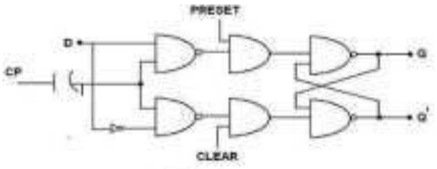
WhentheclockislowboththeNANDgates(N1andN2)aredisabledandQretainsitslast value. When clock is high both the gates are enabled and the input value at D is transferred to its outputQ.Dflip-flopisalsocalled–Dataflip-flop‖.

Truth table

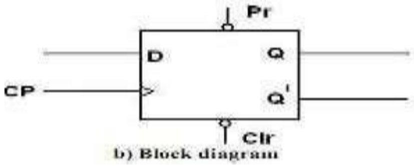


CP	D	Q
0	x	Previous state
1	0	0
1	1	1

EdgeTriggeredDFlip-flop:

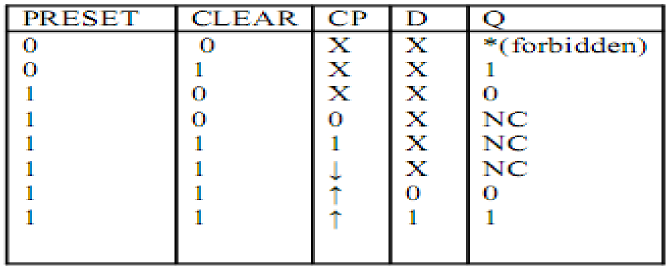


a) Logic diagram



b) Block diagram

Truth table

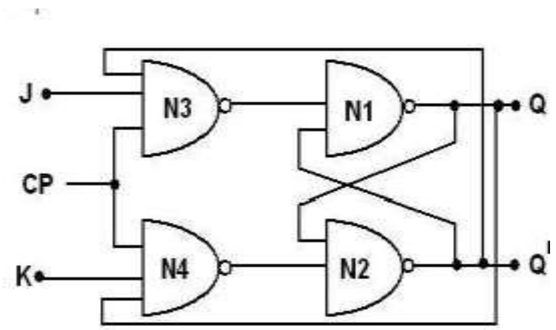


PRESET	CLEAR	CP	D	Q
0	0	X	X	*(forbidden)
0	1	X	X	1
1	0	X	X	0
1	0	0	X	NC
1	1	1	X	NC
1	1	↓	X	NC
1	1	↑	0	0
1	1	↑	1	1

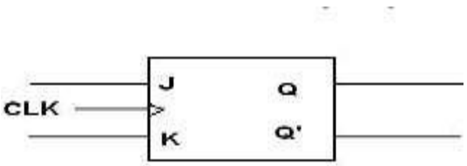
Figure:truthtable,blockdiagram,logicdiagramofedgetriggeredflip-flop JK

flip-flop (edge triggered JK flip-flop)

The race condition in RS flip-flop, when R=S=1 is eliminated in J-K flip-flop. There is a feedback from the output to the inputs. Figure 3.4 represents one wayof building a JK flip-flop**.**

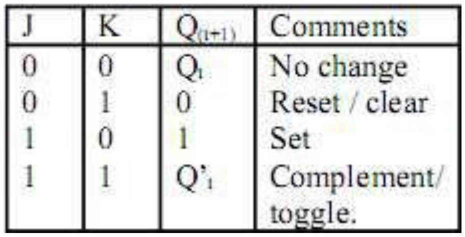


a) Logic diagram



b) Block diagram

Truth table



J	K	$Q_{(t+1)}$	Comments
0	0	Q_t	No change
0	1	0	Reset / clear
1	0	1	Set
1	1	Q'_t	Complement/ toggle.

Figure:JKflip-flop

TheJandKarecalledcontrolinputs,becausetheydeterminewhattheflip-flopdoeswhen a positive clock edge arrives.

Operation:

1. WhenJ=0,K=0thenbothN3andN4willproducehighoutput andthepreviousvalueofQ and Q‘ retained as it is.
2. When J=0, K=1, N3 will get an output as 1 and output of N4 depends on the valueof Q. The final output is Q=0, Q‘=1 i.e., reset state
3. When J=1, K=0the output of N4 is1 and N3 depends on the value of Q‘. Thefinal output is Q=1 and Q‘=0 i.e., setstate
4. When J=1, K=1 it is possible to set (or) reset the flip-flop depending on the current state of output. If Q=1, Q‘=0 then N4 passes ‘0‘to N2 which produces Q‘=1, Q=0 whichis reset state.WhenJ=1,K=1,Qchangestothecomplementofthelaststate.Theflip-flopissaidtobe in the toggle state.

ThecharacteristicequationoftheJKflip-flopis:

$$Q_{next} = J\overline{Q} + \overline{K}Q$$

JKflip-flopoperation								
Characteristictable				**Excitationtable**				
J	**K**	**Q_{next}**	**Comment**	**Q**	**Q_{next}**	**J**	**K**	**Comment**
0	0	Q	hold state	0	0	0	X	Nochange
0	1	0	reset	0	1	1	X	Set
1	0	1	Set	1	0	X	1	Reset
1	1	Q	toggle	1	1	X	0	Nochange

Tflip-flop:

IftheTinputishigh,theTflip-flopchangesstate("toggles")whenevertheclockinputisstrobed. If the T input is low, the flip-flop holds the previous value. This behavior is described by the characteristic equation

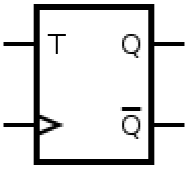


Figure:symbolforTflip flop

$Q_{next} = T \oplus Q = T\overline{Q} + \overline{T}Q$(expandingthe XORoperator)

When T is held high, the toggle flip-flop divides the clock frequency by two; that is, if clock frequency is 4 MHz, the output frequency obtained from the flip-flop will be 2 MHz This "divide by" feature has application in various types of digital counters. A T flip-flop can also be built using a JK flip-flop (J & K pins are connected together and act as T) or D flip-flop (T input and $P_{revious}$ is connected to the D input through an XOR gate).

Tflip-flopoperation[28]							
Characteristictable				**Excitationtable**			
T	Q	Q_{next}	**Comment**	Q	Q_{next}	T	**Comment**
0	0	0	holdstate(no clk)	0	0	0	Nochange
0	1	1	holdstate(no clk)	1	1	0	Nochange
1	0	1	Toggle	0	1	1	Complement
1	1	0	Toggle	1	0	1	Complement

Flipflopoperatingcharacteristics:

Theoperationcharacteristicsspecifytheperformance,operatingrequirements,andoperating limitations of the circuits. The operation characteristics mentions here apply to all flip- flops regardless of the particular form of the circuit.

Propagation Delay Time: is the interval of time required after an input signal has been applied for the resulting output change to occur.

Set-up Time:istheminimumintervalrequiredforthelogiclevelstobemaintainedconstantlyon theinputs(JandK,orSandR,orD)priortothetriggeringedgeoftheclockpulseinorderforthe levels to be reliably clocked into the flip-flop.

Hold Time: istheminimumintervalrequiredforthelogiclevelstoremainontheinputsafterthe triggering edge of the clock pulse inorder for the levels to bereliablyclocked into the flip- flop.

MaximumClockFrequency:isthehighestratethataflip-flopcanbereliablytriggered.**Power**

Dissipation:isthetotalpowerconsumptionofthedevice. Itisequaltoproductofsupply voltage (Vcc) and the current (Icc).

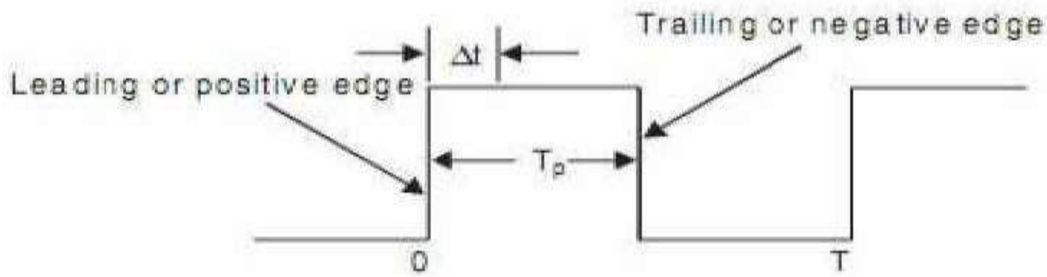
$$P=V_{cc}.I_{cc}$$

Thepower dissipationofaflipflop isusuallyin mW.

PulseWidths:aretheminimumpulsewidthsspecifiedbythemanufacturerfortheClock,SET andCLEARinputs.

Clocktransitiontimes:forreliabletriggering,theclockwaveformtransitiontimesshouldbekept very short. If the clock signal takes too long to make the transitions from one level to other, the flip flop may either triggering erratically or not trigger at all. Race around Condition

The inherent difficulty of an S-R flip-flop (i.e., S = R = 1) is eliminated by using the feedbackconnectionsfromtheoutputstotheinputsofgate1andgate2asshowninFigure.Truth tables in figure were formed with the assumption that the inputs do not change during the clock pulse (CLK = 1). But the consideration is not true because of the feedback connections.



- Consider, for example, that the inputs are J = K = 1 and Q = 1, and a pulse as shown in Figure is applied at the clock input.
- After a time interval t equal to the propagation delay through two NAND gates in series, the outputs will change to Q = 0.So now we haveJ = K = 1 and Q = 0.
- After another time interval of t the output will change back to Q = 1. Hence, we conclude that for the time duration oftP of the clock pulse, the output will oscillate between 0 and 1.Hence,attheendoftheclockpulse,thevalueoftheoutputisnotcertain.Thissituation is referred to as a race-around condition.
- Generally,thepropagationdelayofTTLgatesisoftheorderofnanoseconds.Soif the clock pulse is of the order of microseconds, then the output will change thousands of times within the clockpulse.
- Thisrace-aroundconditioncanbeavoidediftp<t<T.Duetothesmallpropagationdelay of the ICs itmaybe difficult to satisfythe abovecondition.
- A more practical wayto avoid the problem is to use the master-slave (M-S) configuration as discussed below.

Applicationsofflip-flops:

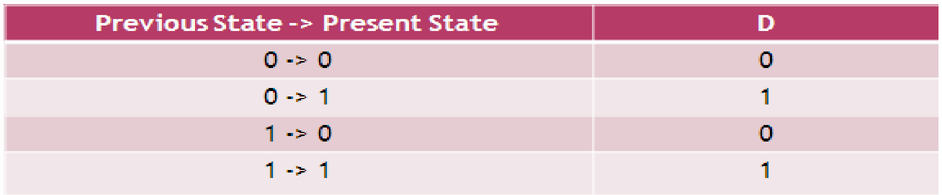
Frequency Division: When a pulse waveform is applied to the clock input of a J-K flip-flopthatisconnectedtotoggle,theQoutputisasquarewavewithhalfthefrequencyof theclock input. If more flip-flops are connected together as shown in the figure below, further division of the clock frequency can be achieved

. **Parallel data storage:** a group of flip-flops is called register. To store data of N bits, N flip-flops are required. Since the data is available in parallel form. When a clock pulse is applied to all flip-flops simultaneously, these bits will transfer will be transferred to the Q outputs of the flipflops.

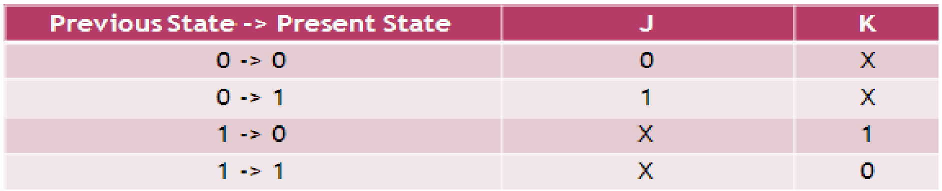
Serial data storage: to store data of N bits available in serial form, N number of D-flip-flopsisconnectedin cascade.Theclocksignalisconnectedto alltheflip-flops.Theserialdatais applied to the D input terminal of the first flip-flop.

Transfer of data: datastoredinflip-flopsmaybetransferredoutinaserialfashion,i.e., bit-by-bit from the output of one flip-flops or may be transferred out in parallel form.

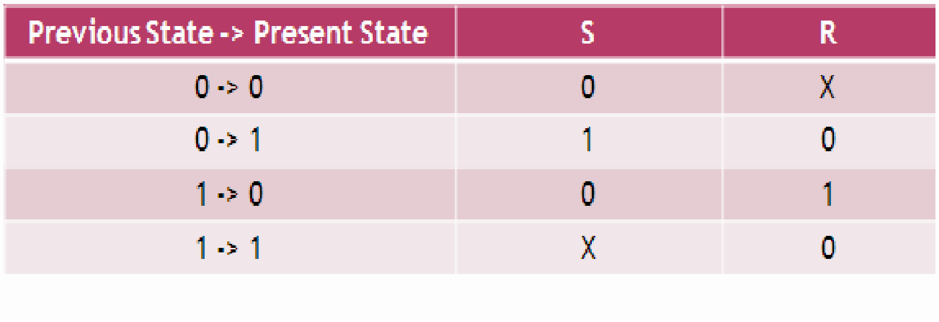
ExcitationTables:



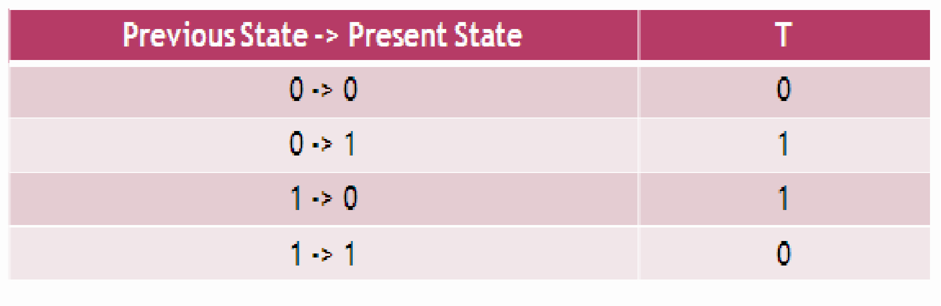
Previous State -> Present State	D
0 -> 0	0
0 -> 1	1
1 -> 0	0
1 -> 1	1



Previous State -> Present State	J	K
0 -> 0	0	X
0 -> 1	1	X
1 -> 0	X	1
1 -> 1	X	0

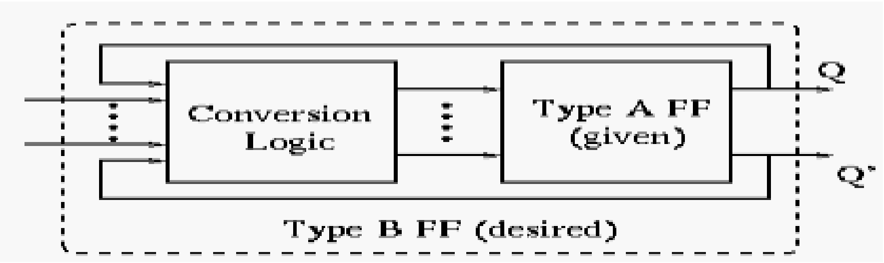


Previous State -> Present State	S	R
0 -> 0	0	X
0 -> 1	1	0
1 -> 0	0	1
1 -> 1	X	0

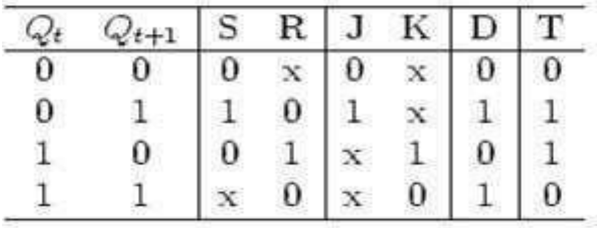


Previous State -> Present State	T
0 -> 0	0
0 -> 1	1
1 -> 0	1
1 -> 1	0

Conversionsofflip-flops:

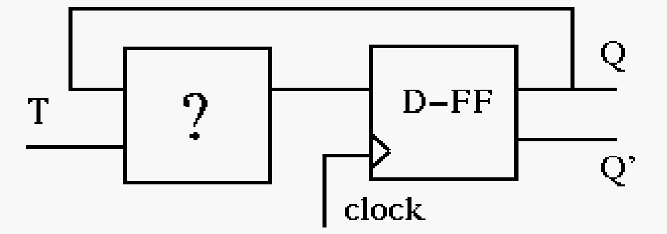


Thekeyhereistousetheexcitationtable,whichshowsthenecessarytriggeringsignal (S,R,J,K, D and T) for a desired flip-flop state transition :



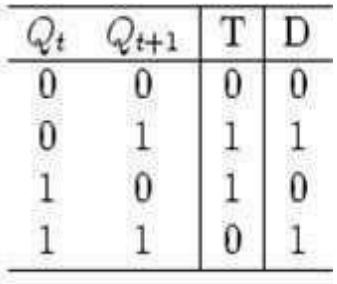
Q_t	Q_{t+1}	S	R	J	K	D	T
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	0	1
1	1	x	0	x	0	1	0

ConvertaD-FFtoaT-FF:



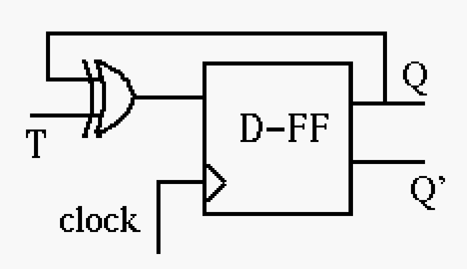
Weneedtodesignthecircuitto generatethetriggeringsignalD asafunctionofTandQ: .Considertheexcitationtable:

$$D = f(T, Q).$$



Q_t	Q_{t+1}	T	D
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

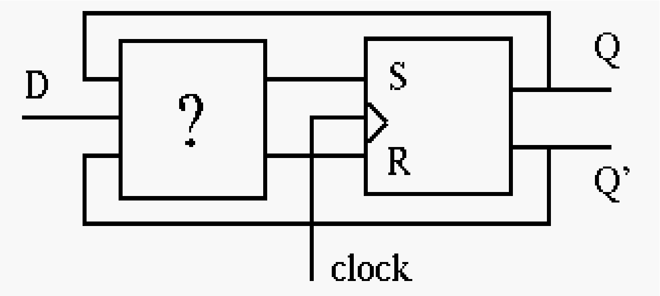
TreatingasafunctionofandcurrentFFstate,wehave

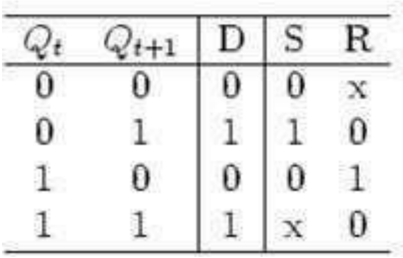


$$D = T'Q + TQ' = T \oplus Q$$

ConvertaRS-FFtoaD-FF:

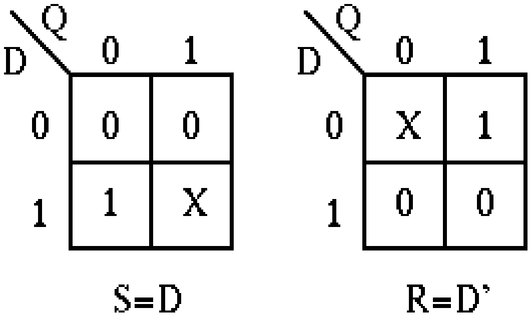
Weneedtodesignthecircuittogeneratethetriggeringsignals SandRasfunctionsof and consider the excitation table:





Q_t	Q_{t+1}	D	S	R
0	0	0	0	x
0	1	1	1	0
1	0	0	0	1
1	1	1	x	0

Thedesiredsignalandcanbeobtainedasfunctionsofandcurrent FFstatefromthe Karnaugh maps:



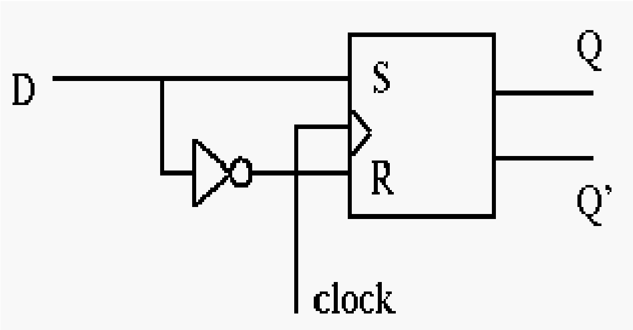
D \ Q	0	1
0	0	0
1	1	X

S=D

D \ Q	0	1
0	X	1
1	0	0

R=D'

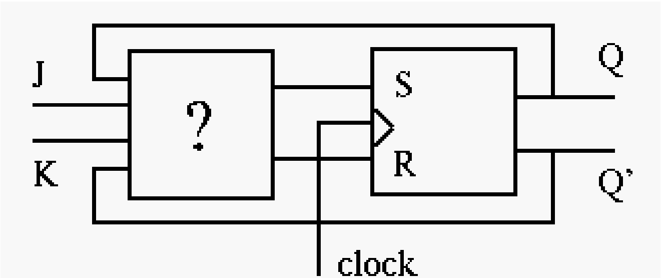
$$S = D, \quad R = D'$$

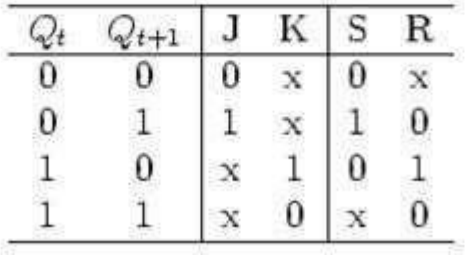


ConvertaRS-FFtoaJK-FF:

Weneedtodesignthecircuittogeneratethetriggeringsignals SandRasfunctionsof,J, K.

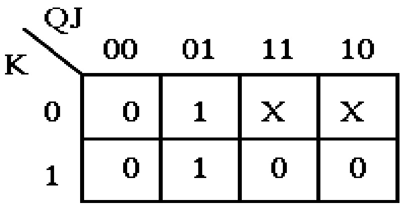
Consider the excitation table**:** Thedesired signal and as functions of, andcurrentFFstate can be obtained from the Karnaugh maps:





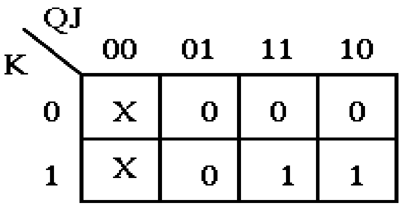
Q_t	Q_{t+1}	J	K	S	R
0	0	0	x	0	x
0	1	1	x	1	0
1	0	x	1	0	1
1	1	x	0	x	0

K-maps:



K \ QJ	00	01	11	10
0	0	1	X	X
1	0	1	0	0

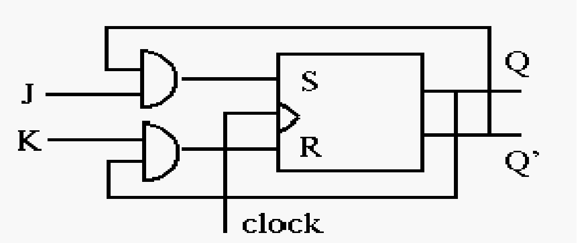
S=Q'J



K \ QJ	00	01	11	10
0	X	0	0	0
1	X	0	1	1

R=QK

$$S = Q'J, \quad R = QK$$



TheMaster-SlaveJKFlip-flop:

The Master-Slave Flip-Flop is basically two gated SR flip-flops connected together in a seriesconfigurationwiththeslavehavinganinvertedclockpulse.TheoutputsfromQand Q from the "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip-flop being connected to the two inputs of the "Slave" flip-flop. This feedback configurationfromtheslave'soutputtothemaster'sinputgivesthecharacteristictoggleof theJK flip-flop as shown below.

The input signals J and K are connected to the gated "master" SR flip-flop which "locks" theinput condition whiletheclock (Clk)input is "HIGH"at logiclevel "1". Astheclock input of the "slave" flip-flop is the inverse (complement) of the "master" clock input, the "slave" SR flip- flop does not toggle. The outputs from the "master" flip-flop are only "seen" bythe gated "slave" flip-flop when the clock input goes "LOW" to logic level "0". When the clock is "LOW", the outputsfromthe"master"flip-floparelatchedandanyadditionalchangestoitsinputsareignored. The gated "slave" flip-flop now responds to the state of its inputs passed over by the "master" section. Then on the "Low-to-High" transition of the clock pulse the inputs of the "master" flip-floparefedthroughtothegatedinputsofthe"slave"flip-flopandonthe"High-to-Low"transition the same inputs are reflected on the output of the "slave" making this type of flip-flop edge or pulse-triggered. Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal. In other words, the Master-Slave JK Flip-flop is a "Synchronous" device as it onlypasses data with the timing of the clock signal.

SequentialCircuitDesign

- Stepsinthedesignprocessforsequentialcircuits
- StateDiagramsandStateTables
- Examples

- StepsinDesignofaSequentialCircuit

1. Specification–Adescriptionofthesequentialcircuit.Shouldincludeadetailingoftheinputs, the outputs, and the operation. Possibly assumes that you have knowledge of digital system basics.
2. Formulation:Generateastatediagramand/orastatetablefromthestatementoftheproblem.
3. StateAssignment: Fromastatetableassignbinarycodestothestates.
4. Flip-flop Input Equation Generation: Select the type of flip-flop for the circuit and generate the needed input for the required statetransitions
5. OutputEquationGeneration:Deriveoutputlogicequationsforgenerationoftheoutputfrom the inputs and current state.
6. Optimization: Optimize the input and output equations. Today, CAD systems are typically used for this in real systems.
7. TechnologyMapping: Generate a logic diagram of the circuit using ANDs, ORs, Inverters, and F/Fs.
8. Verification:UseaHDLtoverifythedesign.

Shiftregisters:

Indigitalcircuits,a**shift register** isacascadeofflip-flopssharingthesameclock,inwhich the output of each flip-flop is connected to the "data" input of the next flip-flop in the chain, resultinginacircuitthatshiftsbyonepositionthe"bitarray"storedinit,*shiftingin*thedatapresent at its input and *shifting out* the last bit in the array, at each transition of the clock input. More generally, a **shift register** may be multidimensional, such that its "data in" and stage outputs are themselvesbitarrays:thisisimplementedsimplybyrunningseveralshiftregistersofthesamebit- length in parallel.

Shift registers can have both parallel and serial inputs and outputs. These are often configured as **serial-in, parallel-out** (SIPO)oras**parallel-in, serial-out** (PISO).Therearealsotypesthathave bothserialandparallelinputandtypeswithserialandparalleloutput.Therearealso**bi-directional** shift registers which allow shifting in both directions: L→R or R→L. The serial input and last output of a shift register can also be connected to create a **circular shift register**

Shiftregisters areatype oflogiccircuitscloselyrelatedtocounters.Theyarebasicallyforthe storage and transfer of digital data.

Bufferregister:

The buffer register is the simple set of registers. It is simply stores the binary word. The buffer may be controlled buffer. Most of the buffer registers used D Flip-flops.

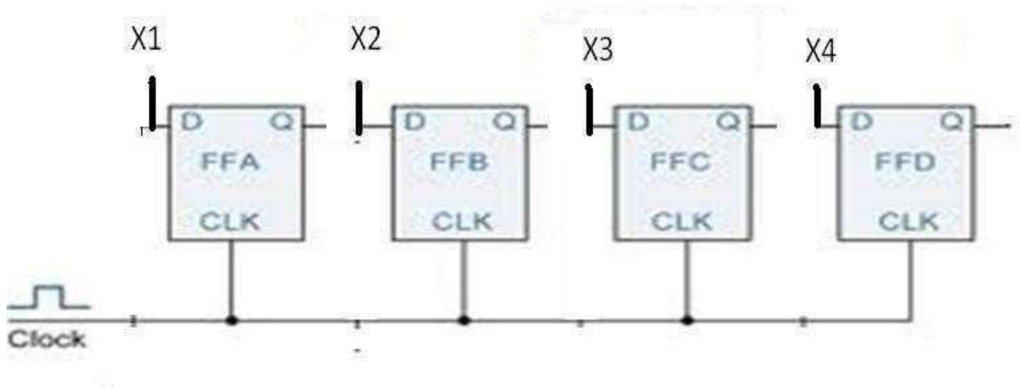


Figure:logicdiagramof4-bitbufferregister

The figure shows a 4-bit buffer register. The binary word to be stored is applied to the data terminals.Ontheapplicationofclockpulse,theoutputwordbecomesthesameasthewordapplied at the terminals. i.e., the input word is loaded into the register bythe application of clock pulse.
Whenthepositiveclockedgearrives,thestoredword becomes:

$$Q_4Q_3Q_2Q_1=X_4X_3X_2X_1$$

$$Q=X$$

Controlledbufferregister:

IfCLRgoesLOW,alltheFFsareRESETandtheoutputbecomes,Q=0000.

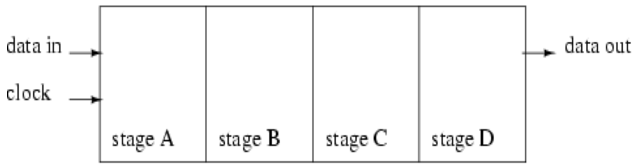
WhenCLR isHIGH,the registeris readyforaction. LOADis thecontrolinput.When LOADis HIGH, the data bits X can reach the D inputs of FF's.

$$Q4Q3Q2Q1=X4X3X2X1$$

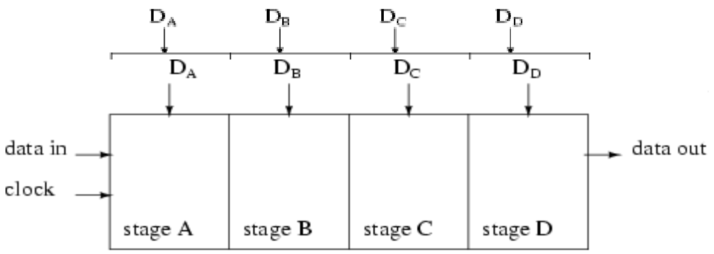
$$Q=X$$

Whenloadislow,theXbitscannotreachthe FF's.

Datatransmissioninshiftregisters:



Serial-in, serial-out shift register with 4-stages



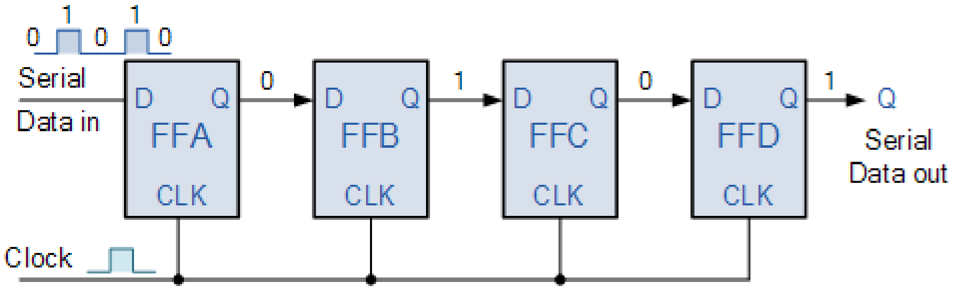
Parallel-in, serial-out shift register with 4-stages

A numberof ff‟s connected together such that data maybe shifted into and shifted out of them is calledshiftregister.datamaybeshiftedintooroutoftheregisterinserialformorinparallelform. There are four basic types of shift registers.

1. Serialin,serialout,shiftright,shiftregisters
2. Serialin,serial out,shiftleft,shiftregisters
3. Parallelin,serialoutshiftregisters
4. Parallelin,paralleloutshiftregisters

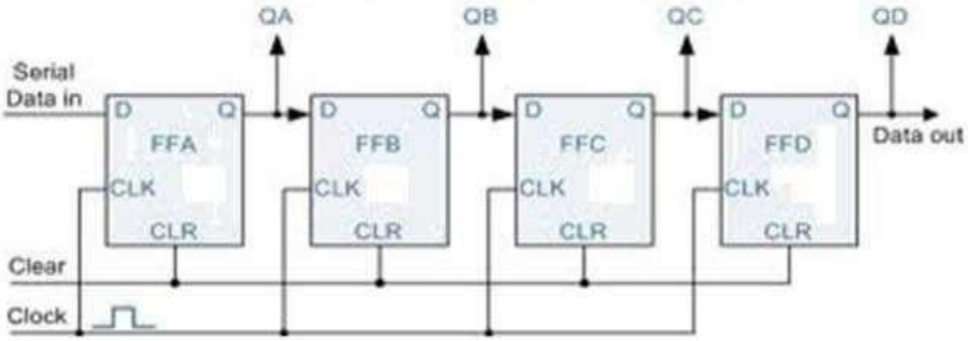
SerialIN,serialOUT,shiftright,shiftleftregister:

Thelogicdiagramof4-bitserialinserialout,rightshiftregisterwithfourstages.Theregistercan store fourbits of data. Serial data is applied at the input D of the first FF. the Q output of the first FF is connected to the D input of another FF. the data is outputted from the Q terminal of the last FF.



When serial data is transferred into a register, each new bit is clocked into the first FF at the positive going edge of each clock pulse. The bit that was previously stored by the first FF is transferredtothesecondFF.thebitthatwasstoredbytheSecondFFistransferredtothethirdFF.

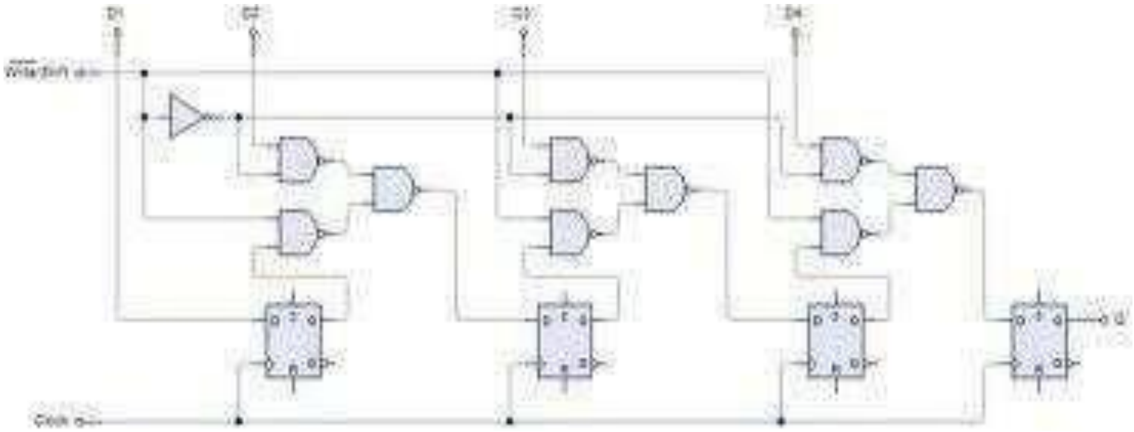
Serial-in,parallel-out,shiftregister:



Inthistypeofregister,thedatabitsareenteredintotheregisterserially,butthedatastoredinthe registerisshiftedoutinparallelform.

Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously, rather than on a bit-by-bit basis with the serial output. The serial-in, parallel out, shift register can be used as serial-in, serial out, shift register if the output is taken from the Q terminal of the last FF.

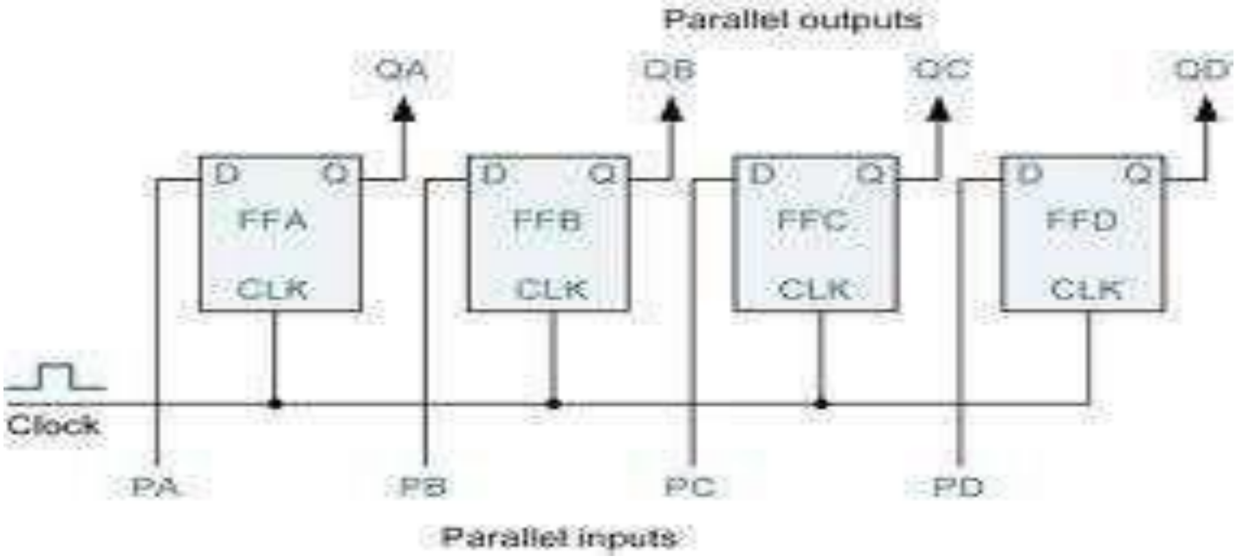
Parallel-in,serial-out,shiftregister:



For a parallel-in, serial out, shift register, the data bits are entered simultaneously into their respective stages on parallel lines, rather than on a bit-by-bit basis on one line as with serial data bits are transferred out of the register serially. On a bit-by-bit basis over a single line.

ThereareFourdatalinesA,B,C,Dthroughwhichthedataisenteredintotheregisterinparallel form. Thesignal shift/ load allows the data to be entered in parallel form into the register and the data is shifted out serially from terminalQ4

Parallel-in,parallel-out,shiftregister



In a parallel-in, parallel-out shift register, the data is entered into the register in parallel form, andalsothedataistakenoutoftheregisterinparallelform.DataisappliedtotheDinputterminals oftheFF‘s.Whenaclockpulseisapplied,atthepositivegoingedgeofthepulse,theDinputsare shiftedintotheQoutputsoftheFFs.Theregisternowstoresthedata.Thestoreddataisavailable instantaneously for shifting out in parallel form.

Bidirectionalshiftregister:

Abidirectionalshiftregisterisonewhichthedatabitscanbeshifted fromlefttorightor from right to left. A fig shows the logic diagram of a 4-bit serial-in, serial out, bidirectional shift register. Right/left is the mode signal, when right /left is a 1, the logic circuit works as a shift-register.thebidirectionaloperationisachievedbyusingthemodesignalandtwoNANDgatesand one OR gate for each stage.

A HIGH on the right/left control input enables the AND gates G1, G2, G3 and G4 and disablestheANDgatesG5,G6,G7andG8,andthestateofQoutputofeachFFispassedthrough the gate to the D input of the following FF. when a clock pulse occurs, the data bits are then effectivelyshiftedoneplacetotheright.ALOWontheright/leftcontrolinputsenablestheAND gatesG5,G6,G7andG8anddisablestheAndgatesG1,G2,G3andG4andtheQoutputofeach FF is passed to the D input of the preceding FF. when a clock pulse occurs, the data bits are then effectivelyshifted one place to the left. Hence, the circuitworks as a bidirectional shift register

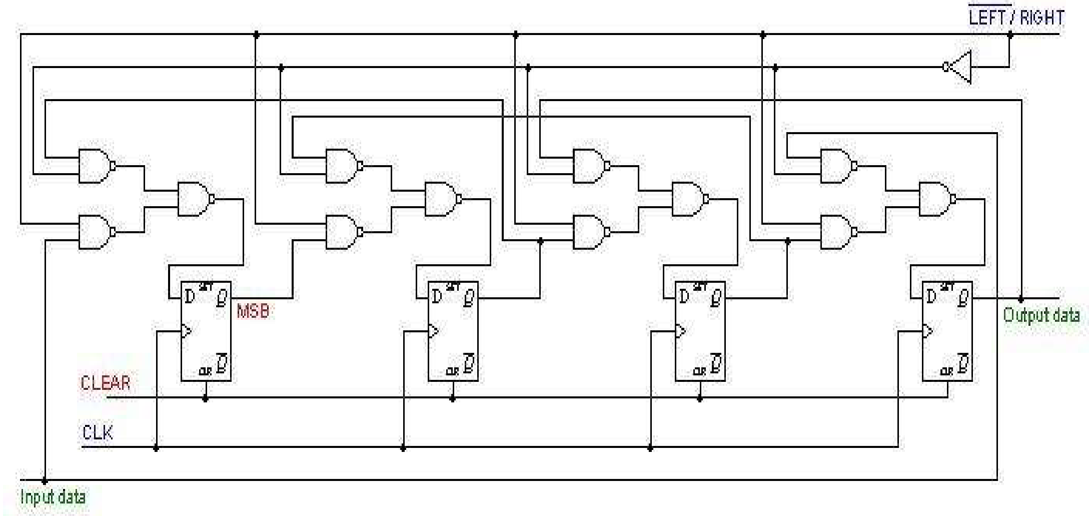


Figure:logicdiagramofa4-bitbidirectionalshiftregister

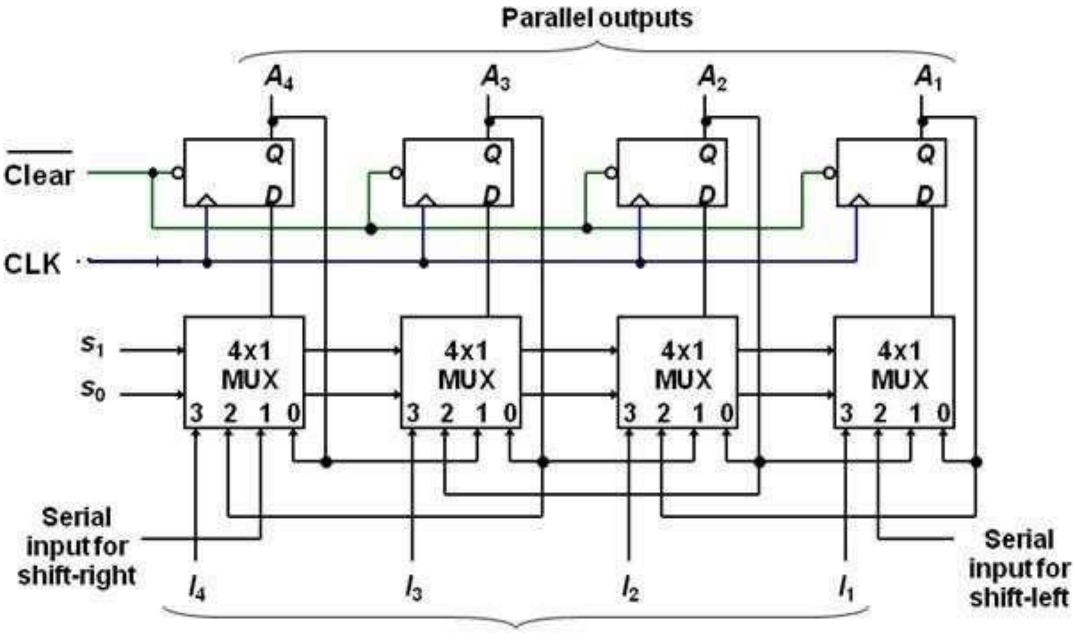
Universal shift register:

Aregisteriscapableofshiftinginonedirectiononlyisaunidirectionalshiftregister.Onethatcan shiftbothdirectionsisabidirectionalshiftregister.Iftheregisterhasboth shiftsandparallelload capabilities,it is referred to as auniversal shift registers.Universal shift registeris abidirectional register,whoseinput can beeitherin serialformorinparallelformandwhoseoutput also can be in serial form or I parallel form.

Themostgeneralshiftregisterhasthefollowingcapabilities.

1. Aclearcontroltocleartheregisterto0
2. Aclockinputtosynchronizetheoperations
3. Ashift-rightcontroltoenabletheshift-rightoperationandserialinputandoutput lines associated with the shift-right. A shift-left control to enable the shift-left operation and serial input and outputlines associated with theshift-left
4. Aparallelloadscontroltoenableaparalleltransferandtheninput linesassociatedwith the paralleltransfer
5. Nparalleloutputlines
6. Acontrolstatethatleavestheinformationintheregisterunchangedinthepresenceof theclock.

A universal shift register can be realized using multiplexers. The below fig shows the logic diagram of a 4-bit universal shift register that has all capabilities. It consists of 4 D flip-flops and four multiplexers. The four multiplexers have two common selection inputs s1 and s0. Input 0 in each multiplexer is selected when S1S0=00, input 1 is selected when S1S0=01 and input 2 is selected when S1S0=10 and input 4 is selected when S1S0=11. The selection inputs control the mode of operation of the register according to the functions entries. When S1S0=0, the present value of the register is applied to the D inputs of flip-flops. The condition forms a path from the outputofeachflip-flopintotheinputofthesameflip-flop.Thenextclockedgetransfersintoeach flip-flop the binary value it held previously, and no change of state occurs. When S1S0=01, terminal1ofthemultiplexerinputshaveapathtotheDinputsoftheflip-flop.Thiscausesashift-rightoperation,withserialinputtransferredintoflip-flopA4.WhenS1S0=10,ashiftleftoperation results with the other serial input going into flip-flop A1. Finally when S1S0=11, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock cycle



4-bit universal shift register

Figure:logicdiagram4-bituniversalshiftregister

Functiontablefortheregister

modecontrol		
S0	**S1**	**registeroperation**
0	0	Nochange
0	1	ShiftRight
1	0	Shiftleft
1	1	Parallelload

Counters:

Counterisadevicewhichstores(andsometimesdisplays)thenumber oftimesparticular event or process has occurred, often in relationship to a clock signal. A Digital counter is a set of flip flops whose state change in response to pulses applied at the input to the counter. Counters may be asynchronous counters or synchronous counters. Asynchronous counters are also called ripple counters

Inelectronicscounterscanbeimplementedquiteeasilyusingregister-typecircuitssuchas the flip-flops and a wide variety of classifications exist:

- Asynchronous(ripple)counter–changingstatebitsareusedasclockstosubsequentstate flip-flops
- Synchronouscounter–allstatebitschangeundercontrolofasingleclock
- Decadecounter–countsthrough tenstatesperstage
- Up/downcounter–countsbothupanddown,undercommandofacontrolinput
- Ringcounter–formedbyashiftregisterwithfeedbackconnectioninaring
- Johnsoncounter–a*twisted*ringcounter
- Cascaded counter
- Moduluscounter.

Each is useful for different applications. Usually, counter circuits are digital in nature, and count in natural binary Many types of counter circuits are available as digital building blocks, for example a number of chips in the 4000 series implement different counters.

Occasionally there are advantages to using a counting sequence other than the natural binary sequence such as the binary coded decimal counter, a linear feed-back shift register counter, or a gray-codecounter.

Countersareusefulfordigitalclocksandtimers,andinoventimers,VCRclocks, etc.

Asynchronouscounters:

An asynchronous (ripple) counter is a single JK-type flip-flop, with its J (data) input fed from its owninvertedoutput. This circuit can store onebit, and hence can count from zeroto one before it overflows (starts over from 0). This counter will increment once for every clock cycle andtakestwoclockcyclestooverflow,soeverycycleitwillalternatebetweenatransitionfrom0 to 1 and a transition from 1 to 0. Notice that this creates a new clock with a 50% duty cycleat exactly half the frequency of the input clock. If this output is then used as the clock signal for a similarlyarrangedDflip-flop(rememberingtoinverttheoutputtotheinput),onewillgetanother 1 bitcounter that counts half as fast. Putting them together yields a two-bit counter:

Two-bitrippleup-counterusingnegativeedgetriggeredflipflop:

Twobitripplecounterusedtwoflip-flops.Therearefourpossiblestatesfrom2–bitup-counting I.e.00,01,10and11.

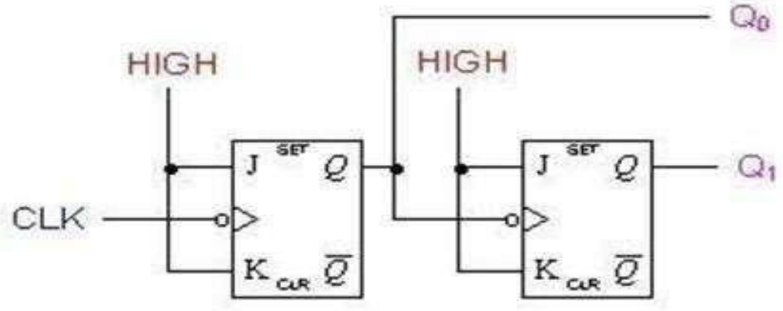
· Thecounterisinitiallyassumedtobeatastate00wheretheoutputsofthetowflip-flopsare noted as Q_1Q_0. WhereQ_1 forms the MSB and Q_0 forms theLSB.

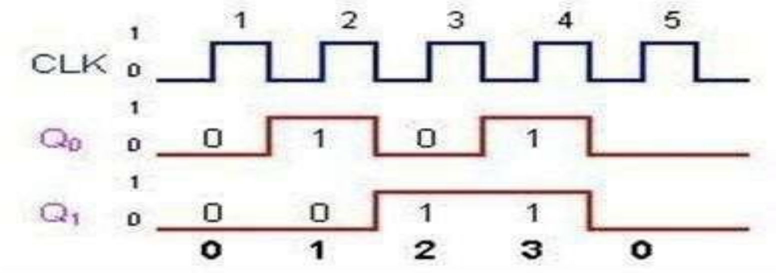
· Forthenegativeedgeofthefirstclockpulse,outputofthefirstflip-flopFF$_1$togglesitsstate. Thus Q_1 remains at 0 and Q_0 toggles to 1 and the counter state are now read as01.

· DuringthenextnegativeedgeoftheinputclockpulseFF$_1$togglesandQ_0=0.TheoutputQ0 being a clock signal for the second flip-flop FF$_2$ and the present transition acts as a negative edge for FF$_2$thus toggles its state $Q_1 = 1$. The counter state isnow read as10.

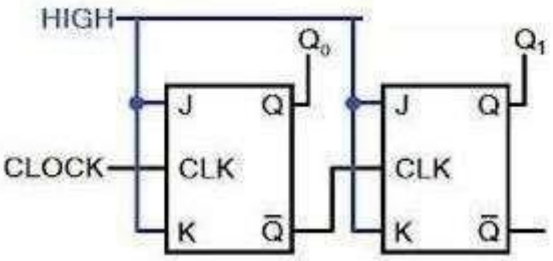
· ForthenextnegativeedgeoftheinputclocktoFF$_1$outputQ0togglesto1.Butthistransition from 0 to 1 being a positive edge for FF$_2$ output Q_1 remains at 1. The counter state is now readas 11.

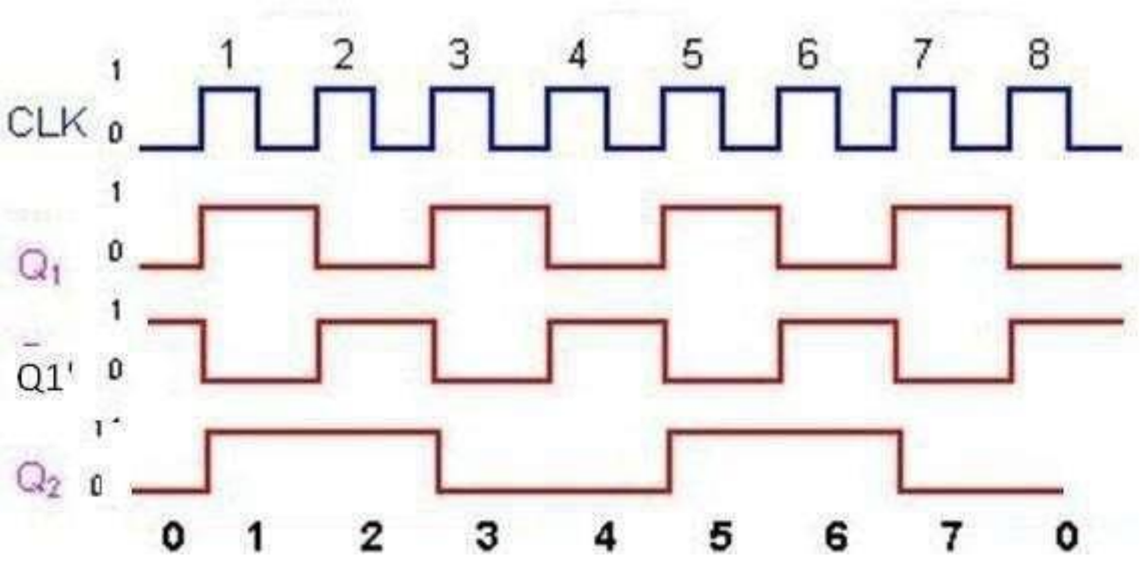
· Forthenextnegativeedgeof theinputclock,Q_0togglesto0.Thistransitionfrom1to0acts asanegativeedgeclockforFF$_2$anditsoutputQ_1togglesto0.Thusthestartingstate00isattained. Figure shownbelow





Two-bitrippledown-counterusingnegativeedgetriggeredflipflop:





A 2-bit down-counter counts in the order 0,3,2,1,0,1…….,i.e, 00,11,10,01,00,11 …..,etc. the above fig. shows ripple down counter, using negative edge triggered J-K FFs and its timing diagram.

- Fordowncounting,Q1‘ofFF1isconnectedtotheclockofFf2. LetinitiallyalltheFF1 toggles, so, Q1 goes from a 0 to a 1 and Q1‘ goes from a 1 to a0.

- The negative-going signal at Q1‘ is applied to the clock input of FF2, toggles Ff2 and, therefore, Q2 goes from a0 to a1.so, after one clock pulseQ2=1 and Q1=1, I.e., thestate of the counter is 11.
- Atthenegative-goingedgeofthesecondclockpulse,Q1changesfroma1toa0andQ1‘ from a 0 to a 1.
- This positive-going signal at Q1‘ does not affect FF2 and, therefore, Q2 remains at a 1. Hence , the state of the counter after second clock pulse is10
- Atthenegativegoingedgeofthethirdclockpulse,FF1toggles.SoQ1, goesfroma0toa 1 and Q1‘from 1 to 0. This negative goingsignal at Q1‘ toggles FF2 and, so, Q2changes from 1 to 0, hence, the state of the counter after the third clock pulse is01.
- Atthenegativegoingedgeofthefourthclockpulse,FF1toggles.SoQ1, goesfroma1to a0andQ1‘from0to1..ThispositivegoingsignalatQ1‘doesnotaffectFF2and,so,Q2 remains at 0, hence,the state of the counter after the fourth clock pulse is 00.

Two-bitrippleup-downcounterusingnegativeedgetriggeredflipflop:

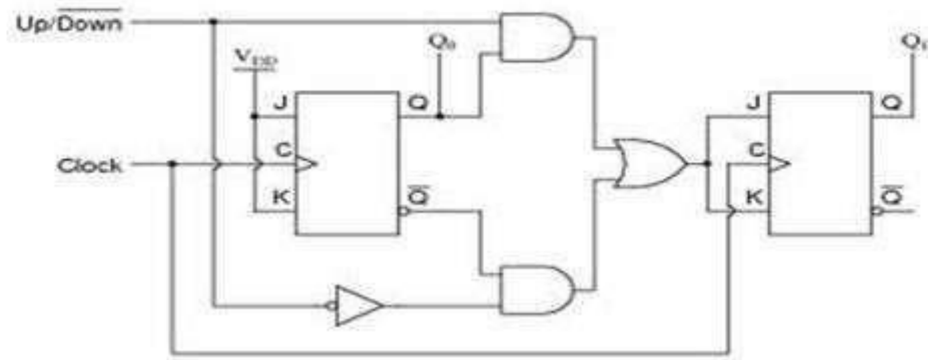


Figure:asynchronous2-bit rippleup-downcounterusingnegative edgetriggered flipflop

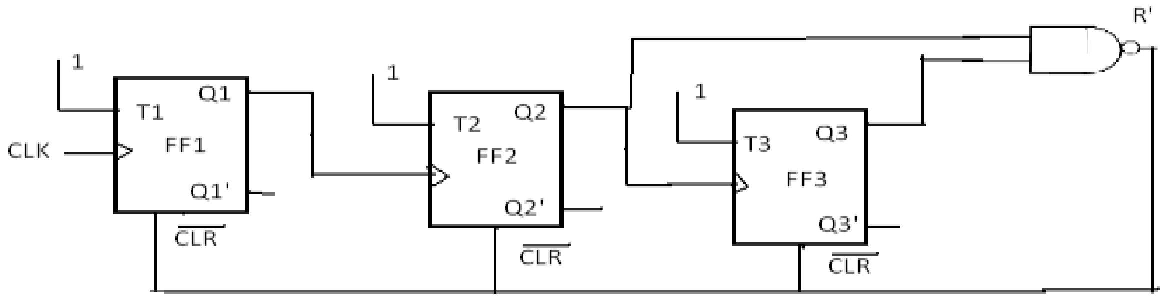
- Asthenameindicatesanup-downcounterisacounterwhichcancountbothinupwardand downward directions. An up-down counter is also called a forward/backward counter or a bidirectional counter. So, a control signal or a mode signal M is required to choose the directionofcount.WhenM=1forupcounting,Q1istransmittedtoclockofFF2andwhen M=0fordowncounting, Q1‘istransmittedtoclockofFF2.Thisisachievedbyusingtwo AND gates and one OR gates. The external clock signal isapplied toFF1.
- Clocksignal toFF2= (Q1.Up)+(Q1‘.Down)=Q1m+Q1‘M‘

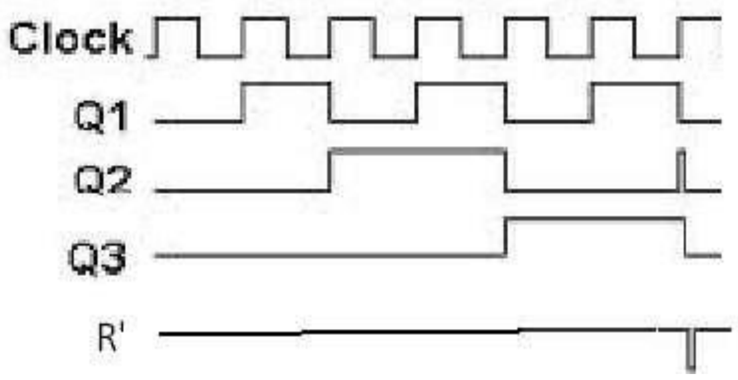
DesignofAsynchronouscounters:

To design a asynchronous counter, first we write the sequence , then tabulate the values of reset signal R for various states of the counter and obtain the minimal expression for R and R‘ usingK-Map oranyothermethod. Provideafeedback such that R andR‘resets all theFF‘safter the desired count

DesignofaMod-6asynchronouscounterusingTFFs:

Amod-6counterhassixstablestates000,001,010,011,100,and101.Whenthesixthclock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of thefeedbackprovided.itis–divideby-6-counter‖,in thesensethatitdividestheinput clock frequency by 6.it requires three FFs, because the smallest value of n satisfying the condition$N \leq 2^n$ isn=3; three FFs can have 8 possible states, out of which onlysix are utilized and the remaining two states 110and 111, are invalid. If initiallythe counter is in 000 state, then after the sixth clock pulse, itgoes to 001, after the second clock pulse, itgoes to 010, and so on.





After sixth clock pulse it goes to 000. For the design, write the truth table with present state outputs Q3, Q2 and Q1 as the variables, and reset R as the output and obtain an expression for R in terms of Q3, Q2, and Q1that decides the feedback into be provided. From the truth table, R=Q3Q2. For active-low Reset, R‘ is used. The reset pulse is of very short duration, of the order of nanoseconds and it is equal to the propagation delay time of the NAND gate used. The expression for R can also be determined as follows.

R=0for000to101,R=1for110,andR=X=for111 R=Q3Q2Q1‘+Q3Q2Q1=Q3Q2

Therefore,

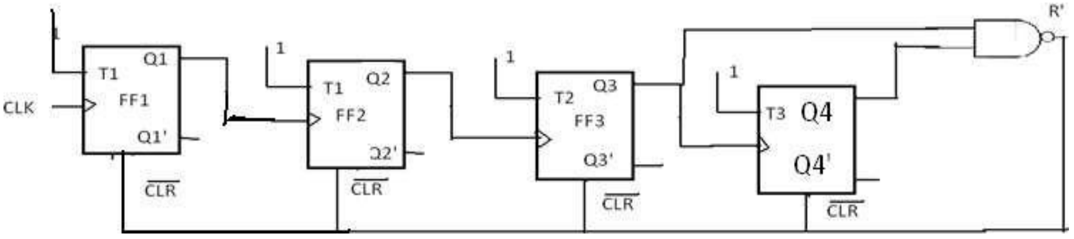
ThelogicdiagramandtimingdiagramofMod-6counterisshownintheabovefig. The

truth table is as shown in below.

After pulses	States			
	Q3	Q2	Q1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓ 0	↓ 0	↓ 0	0
7	0	0	0	0

Designofamod-10asynchronouscounterusingT-flip-flops:

A mod-10 counter is a decade counter. It also called a BCD counter or a divide-by-10 counter.Itrequiresfourflip-flops(condition10≤2^n isn=4).So,thereare16possiblestates,out of which ten are valid and remaining six are invalid. The counter has ten stable state, 0000 through 1001, i.e., it counts from 0 to 9. The initial state is 0000 and after nine clock pulses it goes to 1001. When the tenth clock pulse is applied, the counter goes to state 1010 temporarily, but because of the feedback provided, it resets to initial state 0000. So, there will be a glitch in thewaveformofQ2.Thestate1010isatemporarystateforwhichtheresetsignalR=1,R=0for 0000 to 1001, and R=C for 1011 to 1111.



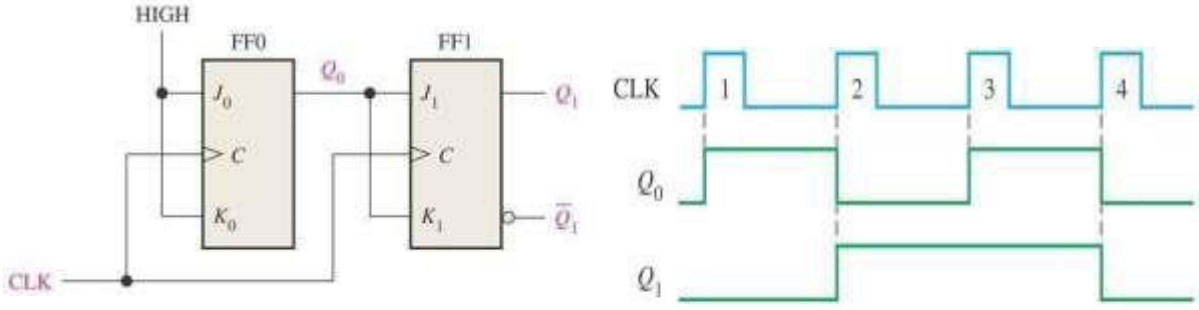
The count table and the K-Map for reset are shown in fig. from the K-Map R=Q4Q2. So, feedbackisprovidedfromsecondandfourthFFs.Foractive–HIGHreset,Q4Q2isappliedtothe clear terminal. For active-LOW reset $Q4Q2$ isconnected CLR isof all Flip=flops.

Q4Q3 \ Q2Q1	00	01	11	10
00				
01				
11	X	X	X	X
10		X	X	1

After pulses	Count Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	0	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	0	1	0	1
10	0	0	0	0

Synchronouscounters:

Asynchronous counters are serial counters. They are slow because each FF can change state only if all the preceding FFs have changed their state. if the clock frequency is very high, the asynchronous counter may skip some of the states. This problem is overcome in synchronous counters or parallel counters. Synchronous counters are counters in which all the flip flops are triggered simultaneously by the clock pulses Synchronous counters have a common clock pulse applied simultaneously to all flip-flops.A 2-Bit Synchronous Binary Counter



Designofsynchronouscounters:

Forasystematicdesignofsynchronouscounters.Thefollowingprocedureisused.

Step 1: StateDiagram:drawthestatediagramshowingallthepossiblestatesstatediagramwhich also be called nth transition diagrams, is a graphical means of depicting the sequence of states through which the counter progresses.

Step2: number of flip-flops: based on the description of the problem, determine the required numbernoftheflip-flops-thesmallestvalueofnissuchthatthenumberofstates$N\le2^n$---andthe desired counting sequence.

Step3: choice of flip-flops excitation table: select the type of flip-flop to be used and write the excitationtable.Anexcitationtableisatablethatliststhepresentstate(ps),thenextstate(ns)and required excitations.

Step4: minimal expressions for excitations: obtain the minimal expressions for the excitations of the FF using K-maps drawn for the excitation of the flip-flops in terms of the present states and inputs.

Step5:logicdiagram:drawalogicdiagrambased ontheminimalexpressions

Designofasynchronous3-bitup-downcounterusingJKflip-flops:

Step1: determine the number of flip-flops required. A 3-bit counter requires three FFs. It has 8 states (000,001,010,011,101,110,111) and all the states are valid. Hence no don‘t cares. For selectingupanddownmodes,acontrolormodesignalMisrequired.WhenthemodesignalM=1 and counts down when M=0. The clock signal is applied to all the FFs simultaneously.

Step2:drawthestatediagrams:thestatediagramofthe3-bitup-downcounterisdrawnas

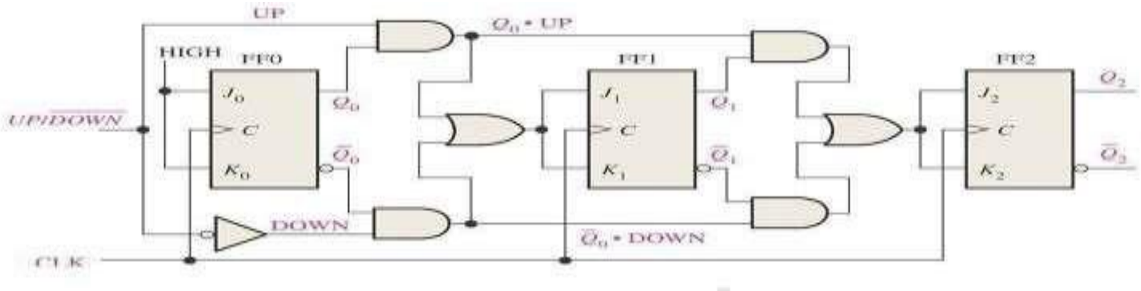
Step3: select thetypeof flip flopand drawtheexcitation table: JKflip-flops areselected and the excitation table of a 3-bit up-down counter using JK flip-flops is drawn as shown in fig.

PS			mode	NS			requiredexcitations					
Q3	Q2	Q1	M	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1
0	0	0	0	1	1	1	1	x	1	x	1	x
0	0	0	1	0	0	1	0	x	0	x	1	x
0	0	1	0	0	0	0	0	x	0	x	x	1
0	0	1	1	0	1	0	0	x	1	x	x	1
0	1	0	0	0	0	1	0	x	x	1	1	x
0	1	0	1	0	1	1	0	x	x	0	1	x
0	1	1	0	0	1	0	0	x	x	0	x	1
0	1	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	1	1	x	1	1	x	1	x
1	0	0	1	1	0	1	x	0	0	x	1	x
1	0	1	0	1	0	0	x	0	0	x	x	1
1	0	1	1	1	1	0	x	0	1	x	x	1
1	1	0	0	1	0	1	x	0	x	1	1	x
1	1	0	1	1	1	1	x	0	x	0	1	x
1	1	1	0	1	1	0	x	0	x	0	x	1
1	1	1	1	0	0	0	x	1	x	1	x	1

Step4: obtain the minimal expressions: From the excitation table we can conclude that J1=1 and K1=1, because all the entries for J1and K1 are either X or 1. The K-maps for J3, K3,J2 and K2 based on the excitation table and the minimal expression obtained from them are shown in fig.

Q3Q2Q1M	00	01	11	10
	1			
			1	
	X	X	X	X
	X	X	X	X

Step5:drawthelogicdiagram:alogicdiagramusingthoseminimalexpressionscanbedrawnas shown in fig.



UNIT IV

MEMORY DEVICES

ROM

The **ROM** (**Read-Only Memory**) is a type of non-volatile memory that aids in storing and retrieving information on a computer. As the name goes, this type of memory allows memory to be read-only. Since it is non-volatile, it does not require a constant power supply to retain the data stored in it. ROM is read-only and cannot have the data altered easily; thus, it is mainly used for firmware purposes.

In the earlier days, the ROM had to be removed and changed physically to change the contents of the memory. But now, there are new types of ROM which allow *limited* rewriting of instructions. We shall discuss these special types of ROM

What are the different types of ROM?

- PROM
- EPROM
- EEPROM.

PROM

The Programmable Read-Only Memory (or the PROM) is a type of ROM that can be programmed only once after its manufacturing.

After the initial programming, no other information can be altered, and the information written on the PROM is permanent. This memory is also described as FPROM (Field Programmable read-only memory) or an OTP (One-Time Programmable) memory.

This memory is better suited to prototyping and low-volume applications. The process of programming the PROM memory is known as burning. To burn information into a PROM, we need to provide a file containing the required contents to be entered into the PROM. A Gang Programmer/Gang Burner then configures each connection as presented in the file.

When a PROM is initially made, all the bits on the memory read as bit 1. We can also consider this as

fused-connections.

If any bit needs to be changed to a 0, it is either etched or burned into the IC, i.e. a large number of current needs to be passed where we do not require a connection and thus blow a fuse. While doing so, if any error occurs in the programming or if there needs to be an update in the information, nothing can be done about it. The entire chip is discarded, and instead a new PROM chip is fabricated, thus having the same data entered into it.

EPROM

Another type of ROM that has often replaced the PROM is the EPROM or the **Erasable Programmable Read-Only Memory.** This system uses a MOSFET as its main programmable component in the circuitry. This MOSFET has a ‘floating gate’, which means that the gate has not been connected yet.

When the chip needs to be programmed, electrons are injected into this floating gate using high voltage, causing the flowing electrons to ‘tunnel’ into the gate. Once the application of high voltage has been removed, these electrons can no longer escape, thus charging the gate and hence, programming is complete.

When we say ‘erasable’, we mean that the information can be rewritten (to some extent). To erase the data written into the EPROM, the electrons which have been trapped need to be excited to escape from the MOSFET’s gate. For this role, ultraviolet light is used for reprogramming.

The EPROM chip is placed under UV light for some time, ranging between 5 to 30 minutes, after which this memory can be rewritten. An EPROM comes with a small quartz circular window which allows the UV rays to reach the chip. For this reason, the EPROM is also called the ‘windowed ROM’.

A programmed EPROM can hold the data programmed in it for a minimum of 10 – 20 years. The

EPROM can be reprogrammed only for a limited number of erasures, as excessive erasing damages the SiO2 layer and makes the EPROM unreliable.

EEPROM

The **Electrically Erasable Programmable Read-Only Memory**, also recognized as the EEPROM or the E^{2}PROM, is another type of ROM which is extensively used in computing systems. This type of ROM is not only programmed electronically, but also erasures to the information in the memory are done electronically.

Unlike the PROM and the EPROM chips, the EEPROM memory chips need not be removed for programming, thus eliminating any possible delays of correcting or updating the data contained in the memory.

The only problem which arises is that the entire contents of the memory need to be rewritten, and not selective erasure. Like the EPROM, this memory can also be programmed only a limited number of times, about a few hundred times. Modern EEPROMs can be programmed around a few million times.

The EEPROMs are calibrated as arrays of floating-gate transistors. One type of EEPROM is the Flash Memory, where selective updating can be done for the information.

EEPROM is mainly used in digital potentiometers, digital temperature sensors, and real-time clocks.

EAPROM stands for **Electronically Alterable Programmable Read-Only Memory**. It is a type of PROM whose contents can be changed. It acts as a non-volatile storage device, and its individual bits can be re-programmed during the course of system operation.

Random Access Memory (RAM)

RAM is called "Random Access Memory" because through it any storage location can be accessed directly. RAM belongs to the class of **volatile memory** which means if the power supplied to the system goes OFF, then data stored inside the RAM will get lost, that is why RAM is generally used to store only temporary data. Hence, RAM is also known as **data memory.**

- **Memory Write** operation can be defined as the process of storing new information into memory.
- **Memory Read** operation can be defined as the process of transferring the stored information out of memory.
- A RAM is capable of performing both Read and Write operations that is why it is also called **Read/Write**

The block diagram of a RAM can be drawn as:

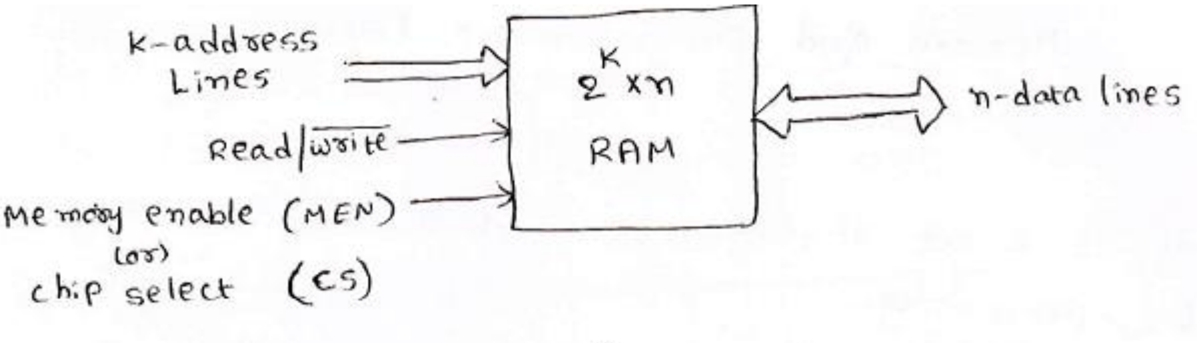


fig: B·D of $2^K \times n$ RAM

In the above figure, the size of RAM is **2^K * N**, it means RAM consists of 2K memory locations and each memory location has a size of n-bits.
The communication between memory and other devices can be achieved through data lines. Each data line carries one bit of binary information. Data lines are bidirectional in nature, but at any time they act as either input or output lines. During memory write operation data lines act as input lines. An address line carries the desired memory location address for memory read or writes operation.

Read / Write is a control signal. It is used to select either Read or Write operation. If **Read / Write =0**, then RAM performs write operation. RAM enables or performs either read or write operation only when its **Chip Select (CS) or Memory Enable (MEN)** input is high. If CS=0, then RAM is disabled.

- **Static Random-Access Memory (SRAM):** SRAM consists of flip-flops to store binary information.
- **Dynamic Random-Access Memory (DRAM):** DRAM consists of CMOS transistors and capacitors. It stored the binary information in the form of electric charges on capacitors.

RAM (Random Access Memory) is the internal memory of the CPU for storing data, program, and program result. It is a read/write memory which stores data until the machine is working. As soon as the machine is switched off, data is erased.

Access time in RAM is independent of the address, that is, each storage location inside the memory is as easy to reach as other locations and takes the same amount of time. Data in the RAM can be accessed randomly but it is very expensive.

RAM is volatile, i.e. data stored in it is lost when we switch off the computer or if there is a power failure. Hence, a backup Uninterruptible Power System (UPS) is often used with computers. RAM is small, both in terms of its physical size and in the amount of data it can hold.

RAM is of two types –

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

Static RAM (SRAM)

The word **static** indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not be refreshed on a regular basis.

There is extra space in the matrix, hence SRAM uses more chips than DRAM for the same amount of storage space, making the manufacturing costs higher. SRAM is thus used as cache memory and has

very fast access.

Characteristic of Static RAM

- Long life
- No need to refresh
- Faster
- Used as cache memory
- Large size
- Expensive
- High power consumption

Dynamic RAM (DRAM)

DRAM, unlike SRAM, must be continually **refreshed** in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory as it is cheap and small. All DRAMs are made up of memory cells, which are composed of one capacitor and one transistor.

Characteristics of Dynamic RAM

- Short data lifetime
- Needs to be refreshed continuously
- Slower as compared to SRAM
- Used as RAM
- Smaller in size
- Less expensive
- Less power consumption

What is ROM?

ROM stands for **Read-only Memory**. It is a type of memory that does not lose its contents when the power is turned off. For this reason, ROM is also called **non-volatile memory**.

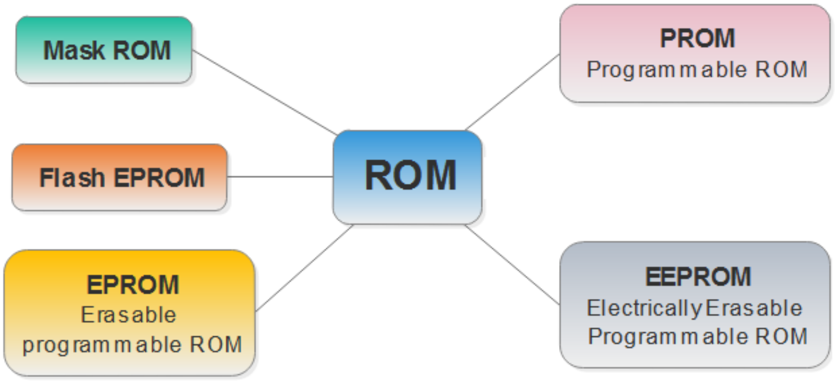
Because ROMs are deployed in such a wide variety of applications, there are different types of ROMs suited to different applications across the industry.

Different Types of ROM

Although all ROM basically serves the same purpose, there are a few different types commonly in use today.

The different types of ROM used in the industry are:

1. **PROM (Programmable ROM)**
2. **EPROM (Erasable Programmable ROM)**
3. **EEPROM (electrically erasable programmable ROM)**
4. **Flash EPROM**
5. **Mask ROM**



Programmable Logic Devices PLDsPLDs are the integrated circuits. They contain an array of AND gates & another array of OR gates. There are three kinds of PLDs based on the type of arrayss, which has programmable feature.

- Programmable Read Only Memory

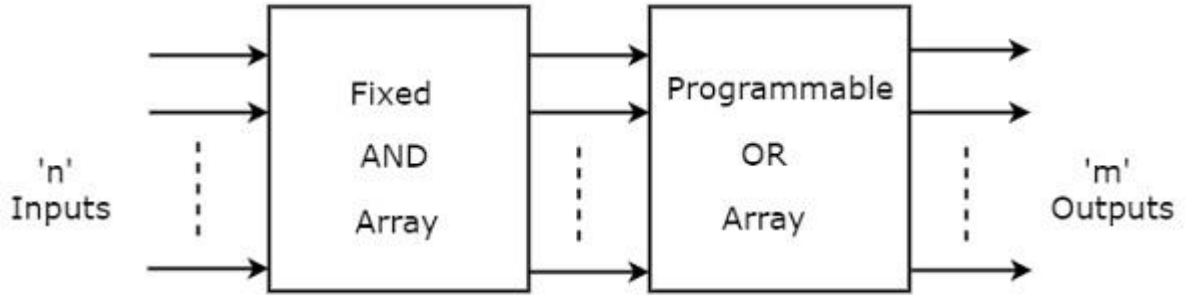
- Programmable Array Logic
- Programmable Logic Array

The process of entering the information into these devices is known as **programming**. Basically, users can program these devices or ICs electrically in order to implement the Boolean functions based on the requirement. Here, the term programming refers to hardware programming but not software programming.

Programmable Read Only Memory PROM

Read Only Memory ROMROM is a memory device, which stores the binary information permanently. That means, we can't change that stored information by any means later. If the ROM has programmable feature, then it is called as **Programmable ROM** PROMPROM. The user has the flexibility to program the binary information electrically once by using PROM programmer.

PROM is a programmable logic device that has fixed AND array & Programmable OR array. The **block diagram** of PROM is shown in the following figure.



Here, the inputs of AND gates are not of programmable type. So, we have to generate 2^n product terms by using 2^n AND gates having n inputs each. We can implement these product terms by using nx2^n decoder. So, this decoder generates 'n' **min terms**.

Here, the inputs of OR gates are programmable. That means, we can program any number of required product terms, since all the outputs of AND gates are applied as inputs to each OR gate. Therefore, the outputs of PROM will be in the form of **sum of min terms**.

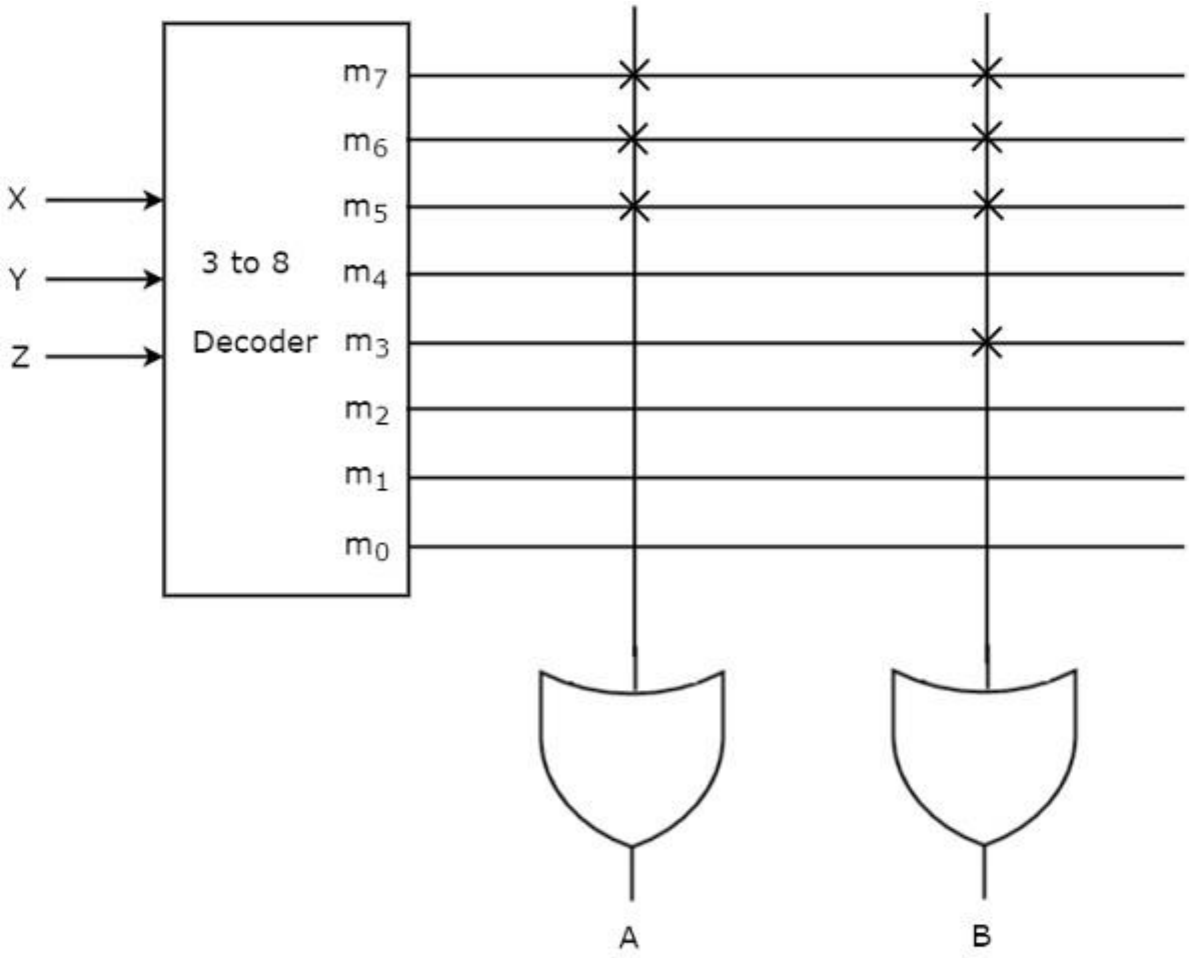
Example

Let us implement the following **Boolean functions** using PROM.

$A(X,Y,Z)=\sum m(5,6,7)$A(X,Y,Z)=∑m(5,6,7)

$B(X,Y,Z)=\sum m(3,5,6,7)$B(X,Y,Z)=∑m(3,5,6,7)

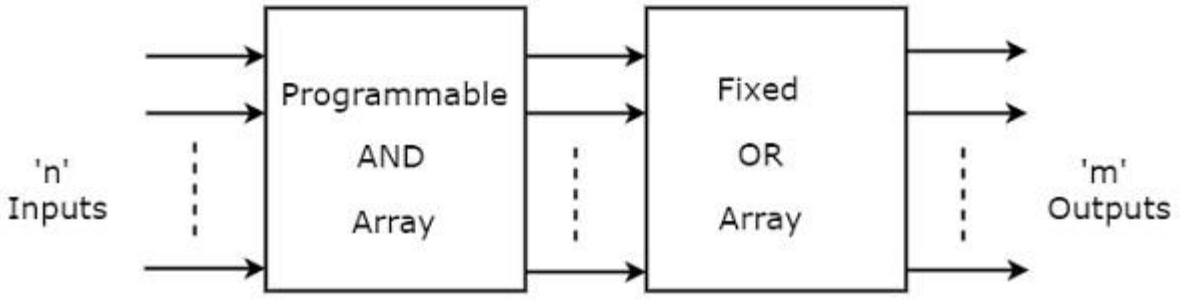
The given two functions are in sum of min terms form and each function is having three variables X, Y & Z. So, we require a 3 to 8 decoder and two programmable OR gates for producing these two functions. The corresponding **PROM** is shown in the following figure.



Here, 3 to 8 decoder generates eight min terms. The two programmable OR gates have the access of all these min terms. But, only the required min terms are programmed in order to produce the respective Boolean functions by each OR gate. The symbol ‘X’ is used for programmable connections.

Programmable Array Logic PAL

PAL is a programmable logic device that has Programmable AND array & fixed OR array. The advantage of PAL is that we can generate only the required product terms of Boolean function instead of generating all the min terms by using programmable AND gates. The **block diagram** of PAL is shown in the following figure.



Here, the inputs of AND gates are programmable. That means each AND gate has both normal and complemented inputs of variables. So, based on the requirement, we can program any of those inputs. So, we can generate only the required **product terms** by using these AND gates.

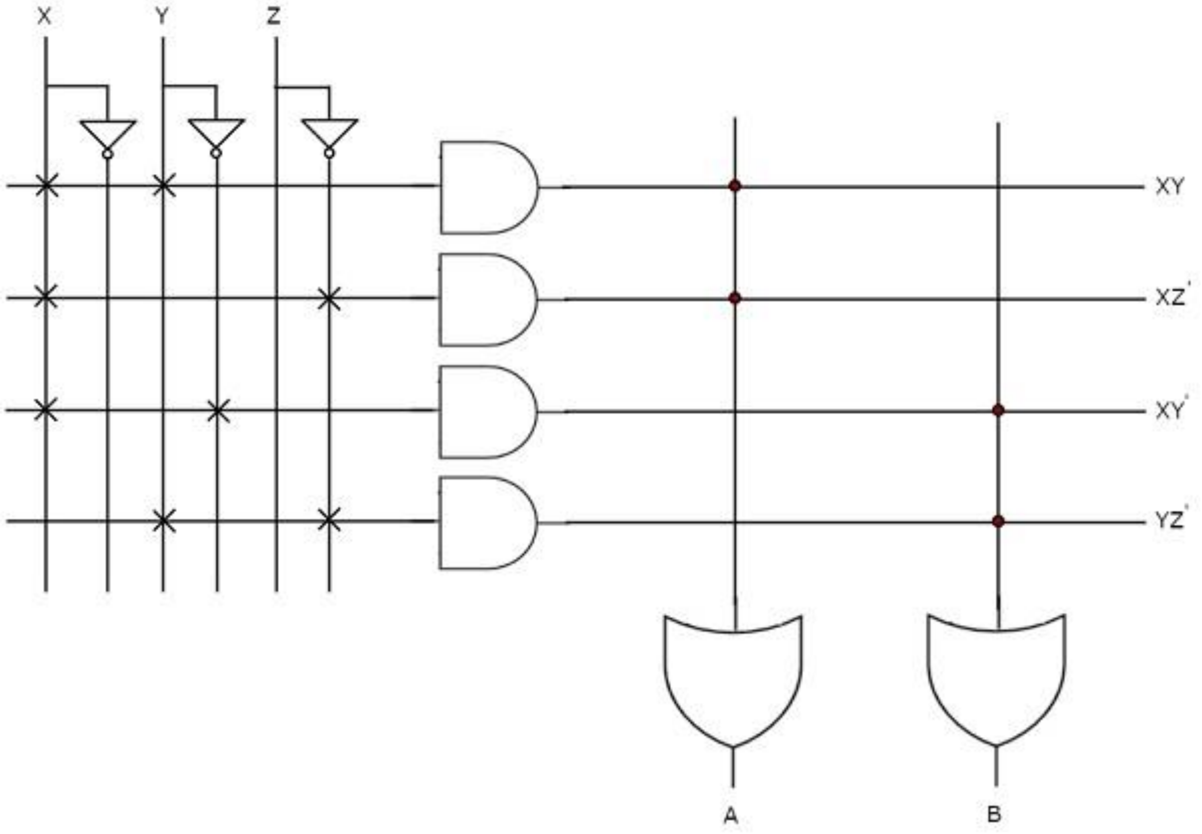
Here, the inputs of OR gates are not of programmable type. So, the number of inputs to each OR gate will be of fixed type. Hence, apply those required product terms to each OR gate as inputs. Therefore, the outputs of PAL will be in the form of **sum of products form**.

Example

Let us implement the following **Boolean functions** using PAL.

$$A=XY+XZ' A=XY+XZ'$$
$$A=XY'+YZ' A=XY'+YZ'$$

The given two functions are in sum of products form. There are two product terms present in each Boolean function. So, we require four programmable AND gates & two fixed OR gates for producing those two functions. The corresponding **PAL** is shown in the following figure.

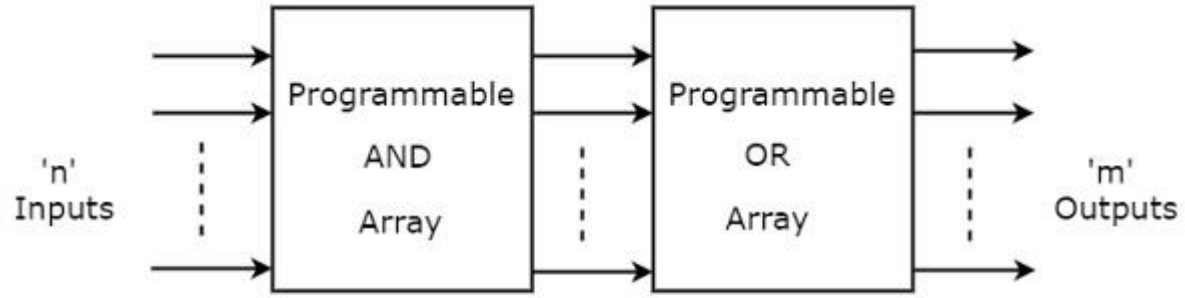


The **programmable AND gates** have the access of both normal and complemented inputs of variables. In the above figure, the inputs X, X'X′, Y, Y'Y′, Z & Z'Z′, are available at the inputs of each AND gate. So, program only the required literals in order to generate one product term by each AND gate. The symbol ‘X’ is used for programmable connections.

Here, the inputs of OR gates are of fixed type. So, the necessary product terms are connected to inputs of each **OR gate**. So that the OR gates produce the respective Boolean functions. The symbol '.' is used for fixed connections.

Programmable Logic Array PLA

PLA is a programmable logic device that has both Programmable AND array & Programmable OR array. Hence, it is the most flexible PLD. The **block diagram** of PLA is shown in the following figure.



Here, the inputs of AND gates are programmable. That means each AND gate has both normal and complemented inputs of variables. So, based on the requirement, we can program any of those inputs. So, we can generate only the required **product terms** by using these AND gates.

Here, the inputs of OR gates are also programmable. So, we can program any number of required product terms, since all the outputs of AND gates are applied as inputs to each OR gate. Therefore, the outputs of PAL will be in the form of **sum of products form**.

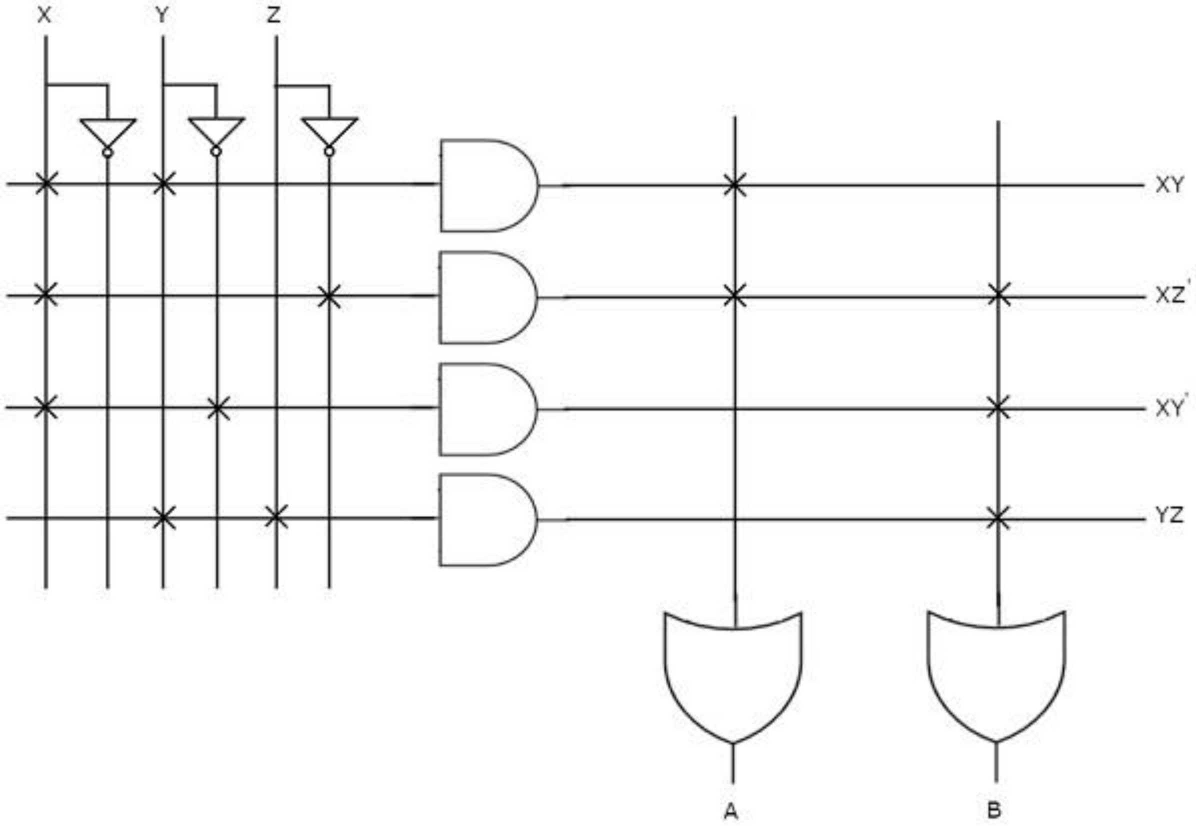
Example

Let us implement the following **Boolean functions** using PLA.

$$A=XY+XZ' \quad A=XY+XZ'$$
$$B=XY'+YZ+XZ' \quad B=XY'+YZ+XZ'$$

The given two functions are in sum of products form. The number of product terms present in the given Boolean functions A & B are two and three respectively. One product term, $Z'X$Z′X is common in each function.

So, we require four programmable AND gates & two programmable OR gates for producing those two functions. The corresponding **PLA** is shown in the following figure.



The **programmable AND gates** have the access of both normal and complemented inputs of variables. In the above figure, the inputs X, X′X′, Y, Y′Y′, Z & Z′Z′, are available at the inputs of each AND gate. So, program only the required literals in order to generate one product term by each AND gate.

All these product terms are available at the inputs of each **programmable OR gate**. But, only program the required product terms in order to produce the respective Boolean functions by each OR gate. The symbol ‘X’ is used for programmable connections.

WEB RESOURCES UNIT IV PLD’S
https://www.tutorialspoint.com/digital_circuits/digital_circuits_programmable_logic_devices.htm#

unit 1,2,3

file:///C:/Users/ksaji/Downloads/Digital-Logic-Design-3%20(2).pdf