LOGIC GATES & COMBINATIONAL CIRCUITS

Compiled By: Afaq Alam Khan

Index

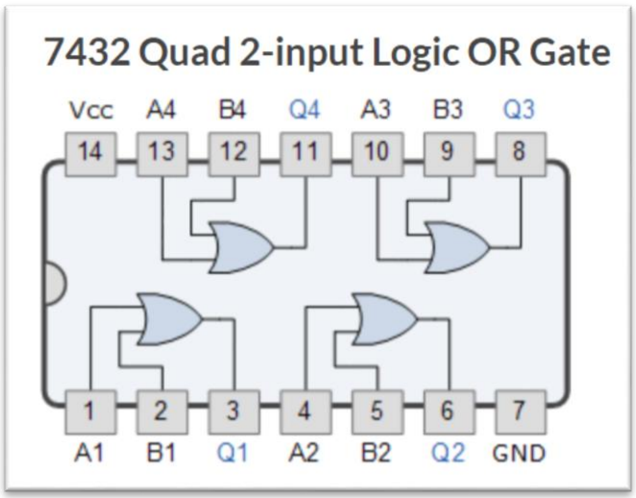
- Logic Gates
- Universal Gates
- Combinational Circuits

Digital Logic Gates

- A Digital Logic Gate is an electronic circuit which makes logical decisions based on the combination of digital signals present on its inputs
- Digital logic gates can have more than one inputs, but generally only have one digital output.
- Individual logic gates can be connected or cascaded together to form a logic gate function with any desired number of inputs, or to form **combinational** and **sequential** type circuits, or to produce different logic gate functions from standard gates.

Digital Logic Gates

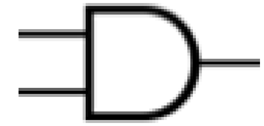
- Commercially available digital logic gates are available in two basic families or forms, **TTL** which stands for *Transistor-Transistor Logic* such as the 7400 series, and **CMOS** which stands for *Complementary Metal-Oxide-Silicon* which is the 4000 series of chips.
- This notation of TTL or CMOS refers to the logic technology used to manufacture the integrated circuit, (IC) or a "chip" as it is more commonly called.
- Generally speaking, **TTL** logic IC's use NPN and PNP type Bipolar Junction Transistors while **CMOS** logic IC's use complementary MOSFET or JFET type Field Effect Transistors for both their input and output circuitry.



Types of Logical Gates

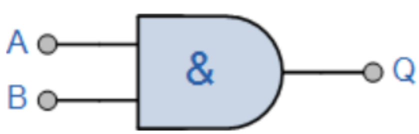
- AND Gate
- OR Gate
- NOT Gate
- NANAD Gate
- NOR Gate
- XOR Gate
- XNOR Gate

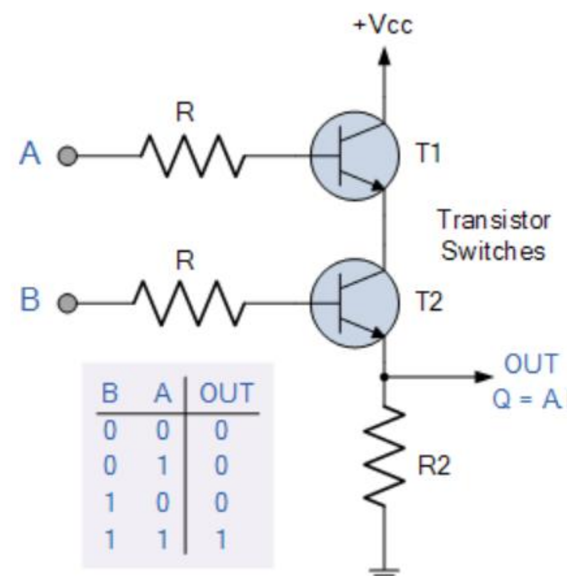
AND Gate



- The Logic AND Gate is a type of digital logic circuit whose output goes HIGH to a logic level 1 only when all of its inputs are HIGH

The 2-input Logic AND Gate

Symbol	Truth Table		
 2-input AND Gate	B	A	Q
	0	0	0
	0	1	0
	1	0	0
	1	1	1
Boolean Expression **Q = A.B**	Read as A **AND** B gives Q		



"If both A and B are true, then Q is true"

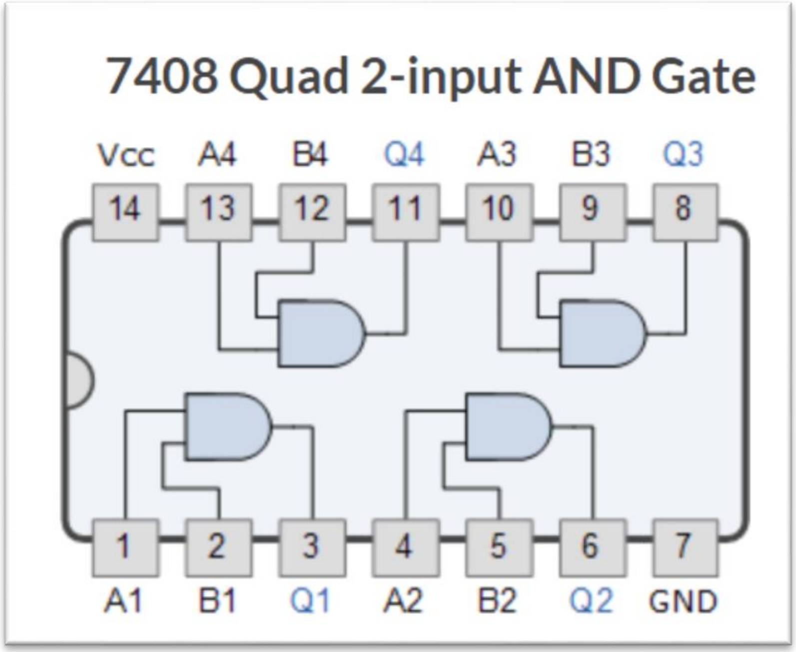
AND Gate

- Commonly available digital logic AND gate IC's include:
- TTL Logic AND Gate

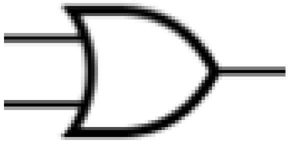
74LS08 Quad 2-input
74LS11 Triple 3-input
74LS21 Dual 4-input

- CMOS Logic AND Gate

CD4081 Quad 2-input
CD4073 Triple 3-input
CD4082 Dual 4-input

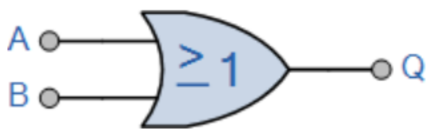


OR Gate



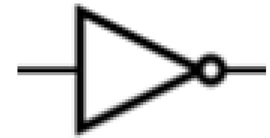
- The Logic OR Gate is a type of digital logic circuit whose output goes HIGH to a logic level 1 only when one or more of its inputs are HIGH

The 2-input Logic OR Gate

Symbol	Truth Table		
	B	A	Q
 2-input OR Gate	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression **Q = A+B**	Read as A **OR** B gives Q		

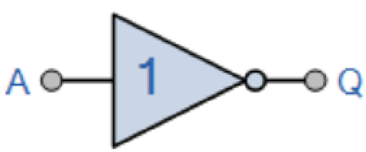
"If either A or B is true, then Q is true"

NOT Gate



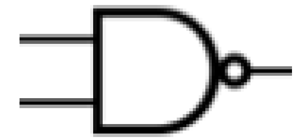
- The Logic NOT Gate is the most basic of all the logical gates and is often referred to as an Inverting Buffer or simply an Inverter.
- Inverting NOT gates are single input devicse which have an output level that is normally at logic level “1” and goes “LOW” to a logic level “0” when its single input is at logic level “1”, in other words it “inverts” (complements) its input signal.

The Logic NOT Gate Truth Table

Symbol	Truth Table	
 Inverter or NOT Gate	A	Q
	0	1
	1	0
Boolean Expression **Q = not A or $\overline{A}$**	Read as inverse of **A** gives **Q**	

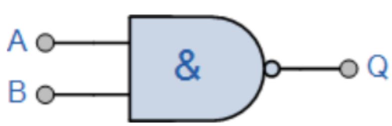
“If A is NOT true, then Q is true”

NAND Gate



- The Logic NAND Gate is a combination of a digital logic AND gate and a NOT gate connected together in series
- The NAND (Not – AND) gate has an output that is normally at logic level "1" and only goes "LOW" to logic level "0" when **ALL** of its inputs are at logic level "1". The **Logic NAND Gate** is the reverse or "*Complementary*" form of the AND gate we have seen previously.

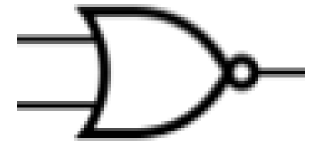
2-input Logic NAND Gate

Symbol	Truth Table		
 2-input NAND Gate	B	A	Q
	0	0	1
	0	1	1
	1	0	1
	1	1	0
Boolean Expression **Q** = $\overline{A.B}$	Read as A **AND** B gives **NOT** Q		

Universal Gate

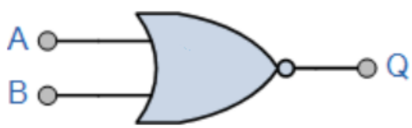
"If both A and B are true, then Q is NOT true"

NOR Gate



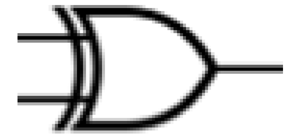
- The Logic NOR Gate gate is a combination of the digital logic OR gate and an inverter or NOT gate connected together in series
- Universal Gate

2-input NOR Gate

Symbol	Truth Table		
 2-input NOR Gate	B	A	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	0
Boolean Expression **Q** = $\overline{A+B}$	Read as A **OR** B gives **NOT** Q		

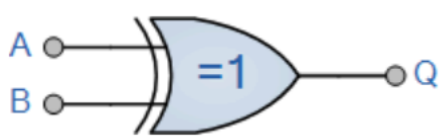
"If both A and B are NOT true, then Q is true"

XOR



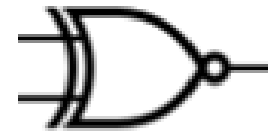
- Exclusive-OR gate ONLY goes “HIGH” when both of its two input terminals are at “DIFFERENT” logic levels with respect to each other. If these two inputs, A and B are both at logic level “1” or both at logic level “0” the output is a “0”

2-input Ex-OR Gate

Symbol	Truth Table		
	B	A	Q
 2-input Ex-OR Gate	0	0	0
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = A \oplus B$	A **OR** B but NOT **BOTH** gives Q		

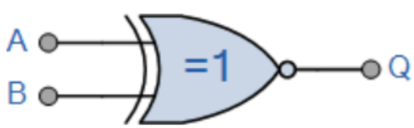
$$Q = (A \oplus B) = A.B' + A'.B$$

XNOR Gate



- The Exclusive-NOR Gate function is a digital logic gate that is the reverse or complementary form of the Exclusive-OR function

2-input Ex-NOR Gate

<table>
<tr><th>Symbol</th><th colspan="3">Truth Table</th></tr>
<tr><td rowspan="5">

2-input Ex-NOR Gate</td><td>B</td><td>A</td><td>Q</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Boolean Expression $Q = \overline{A \oplus B}$</td><td colspan="3">Read if A AND B the SAME gives Q</td></tr>
</table>

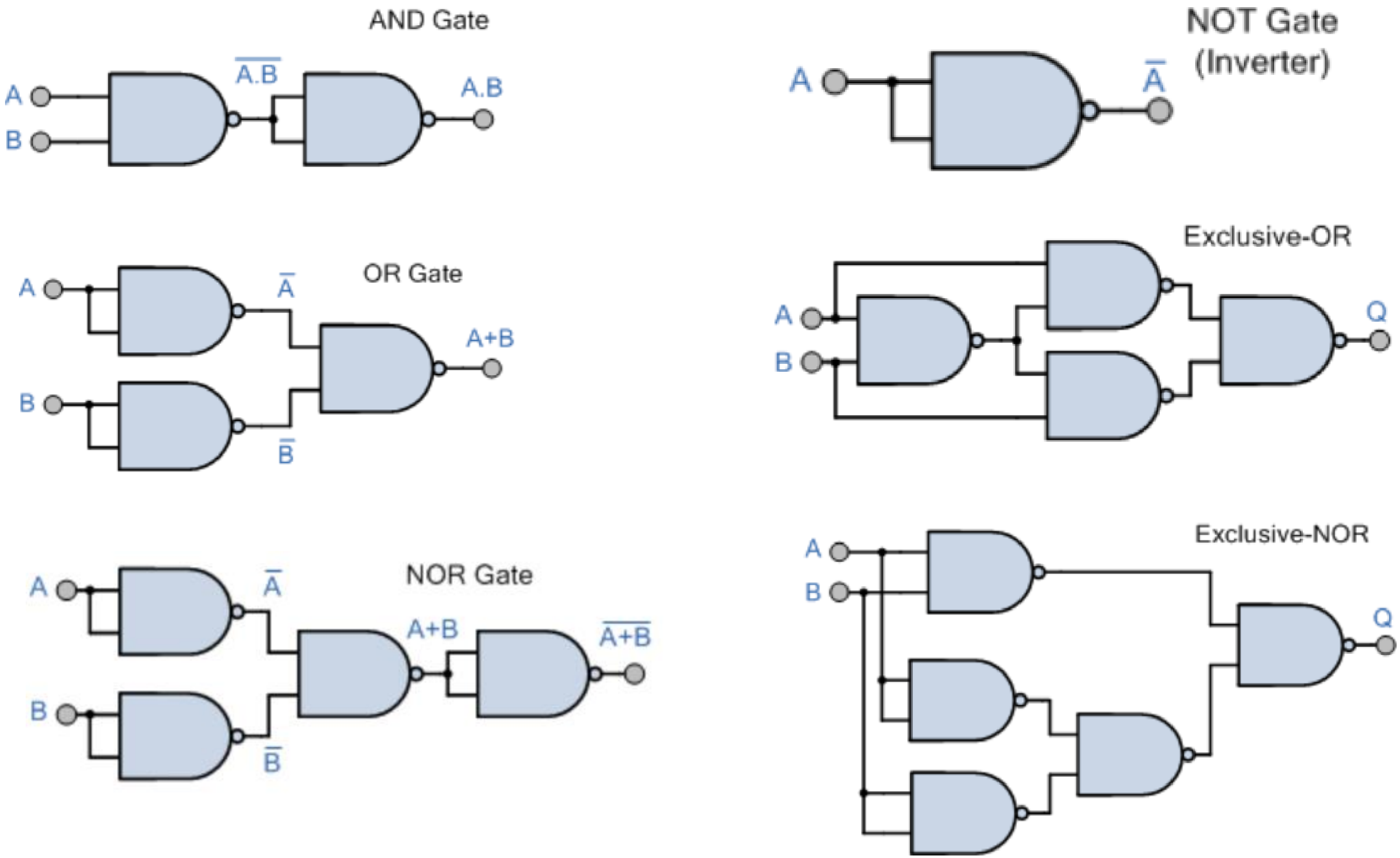
$Q = A'B' + AB$

Universal Logic Gates

- Universal Logic gates can be used to produce any other logic or Boolean function with the NAND and NOR gates being minimal

AND, OR and NOT (a Full Set)
AND and NOT (a Complete Set)
OR and NOT (a Complete Set)
NAND (a Minimal Set)
NOR (a Minimal Set)

Implementation of Gates Using NANAD Gate



Exercise

- Construct All Gates with the help of NOR Gate

List

TTL Logic OR Gates

74LS32 Quad 2-input

CMOS Logic OR Gates

CD4071 Quad 2-input

CD4075 Triple 3-input

CD4072 Dual 4-input

TTL Logic NOT Gates

74LS04 Hex Inverting NOT Gate

74LS14 Hex Schmitt Inverting NOT Gate

74LS1004 Hex Inverting Drivers

CMOS Logic NOT Gates

CD4009 Hex Inverting NOT Gate

CD4069 Hex Inverting NOT Gate

TTL Logic NAND Gates

74LS00 Quad 2-input

74LS10 Triple 3-input

74LS20 Dual 4-input

74LS30 Single 8-input

CMOS Logic NAND Gates

CD4011 Quad 2-input

CD4023 Triple 3-input

CD4012 Dual 4-input

TTL Logic NOR Gates

74LS02 Quad 2-input

74LS27 Triple 3-input

74LS260 Dual 4-input

CMOS Logic NOR Gates

CD4001 Quad 2-input

CD4025 Triple 3-input

CD4002 Dual 4-input

TTL Logic Ex-OR Gates

74LS86 Quad 2-input

CMOS Logic Ex-OR Gates

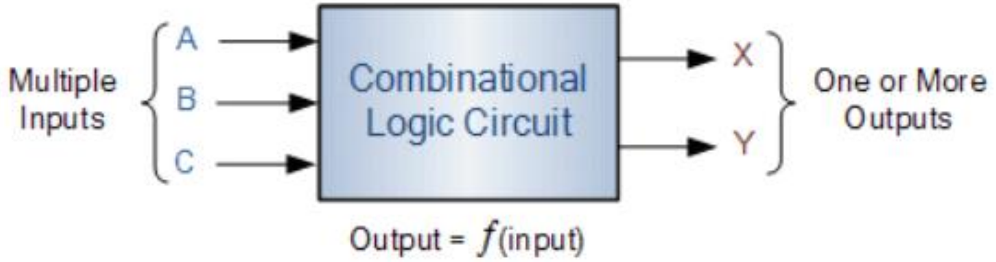
CD4030 Quad 2-input

TTL Logic Ex-NOR Gates

74LS266 Quad 2-input

COMBINATIONAL CIRCUITS

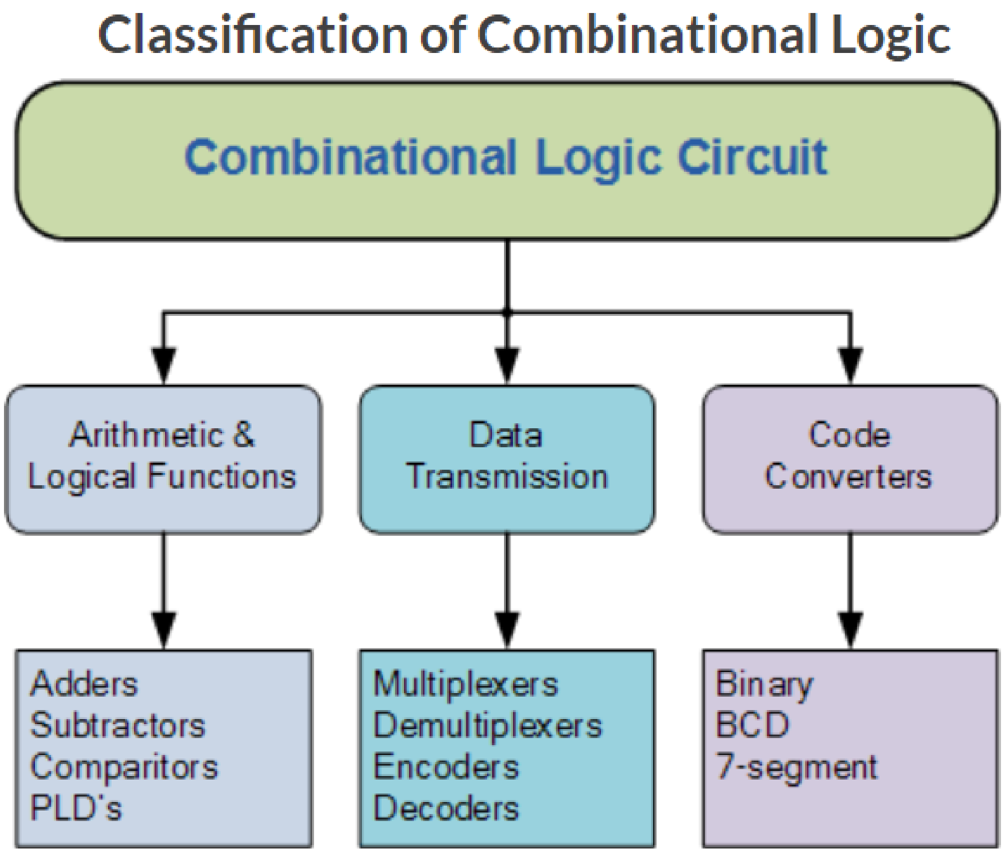
- Combinational Logic Circuits are memory less digital logic circuits whose output at any instant in time depends only on the combination of its inputs



- Combinational logic circuits have no feedback, and any changes to the signals being applied to their inputs will immediately have an effect at the output.
- In other words, in a **Combinational Logic Circuit**, the output is dependant at all times on the combination of its inputs. Thus a combinational circuit is *memoryless.*

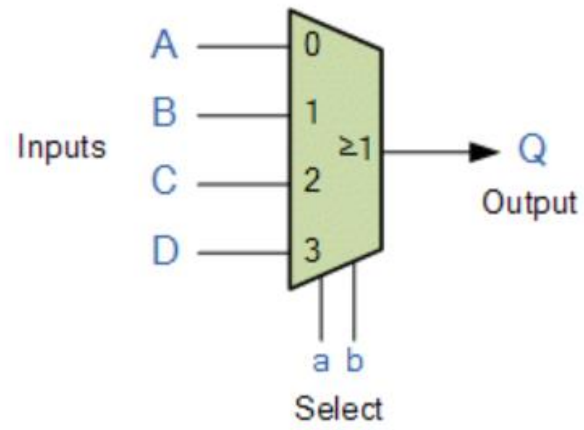
- **Combinational Logic Circuits** are made up from logic gates that are "combined" or connected together to produce more complicated switching circuits. These logic gates are the building blocks of combinational logic circuits.
- The three main ways of specifying the function of a combinational logic circuit are Boolean Algebra, Truth Table, Logic Diagram.
- Common combinational circuits made up from individual logic gates that carry out a desired application include *Multiplexers*, *Demultiplexers*, *Encoders*, *Decoders*, *Full* and *Half Adders* etc.

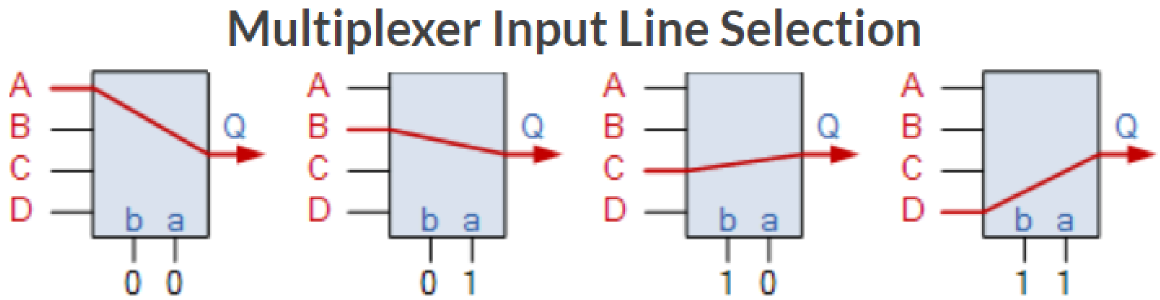
Classification



Multiplexer

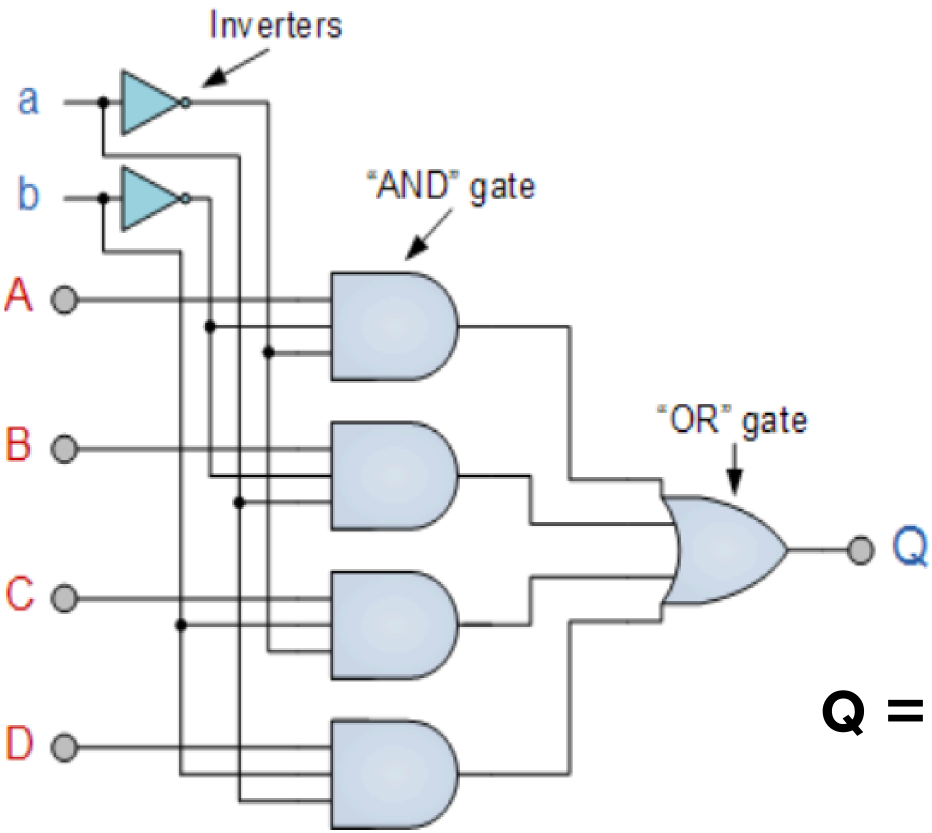
- The multiplexer is a combinational logic circuit designed to switch one of several input lines to a single common output line
- The *multiplexer*, shortened to “MUX” or “MPX”, is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal





4x1 Multiplexer

4 Channel Multiplexer using Logic Gates

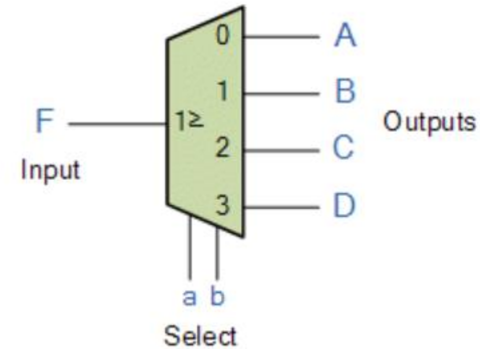


b	a	Q
0	0	A
0	1	B
1	0	C
1	1	D

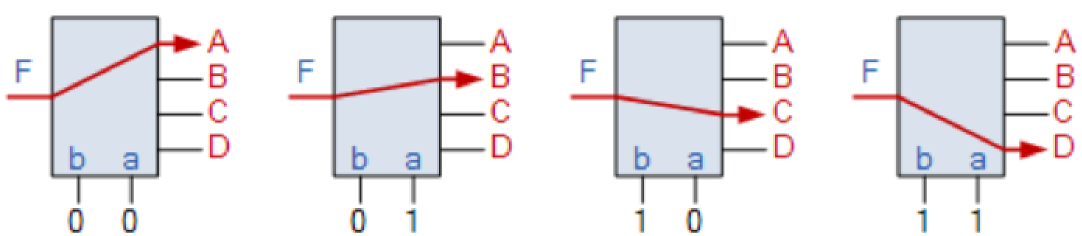
Q = a'b'A + ab'B + a'bC + abD

DeMutipexer

- The data distributor, known more commonly as a **Demultiplexer** or "Demux" for short, is the exact opposite of the Multiplexer
- The *demultiplexer* takes one single input data line and then switches it to any one of a number of individual output lines one at a time.

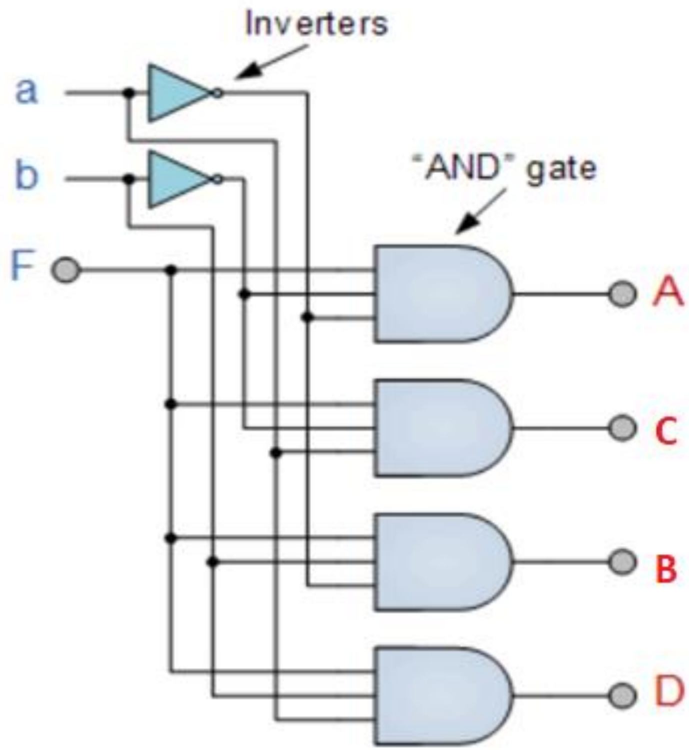


Demultiplexer Output Line Selection



1 to 4 DeMux

4 Channel Demultiplexer using Logic Gates

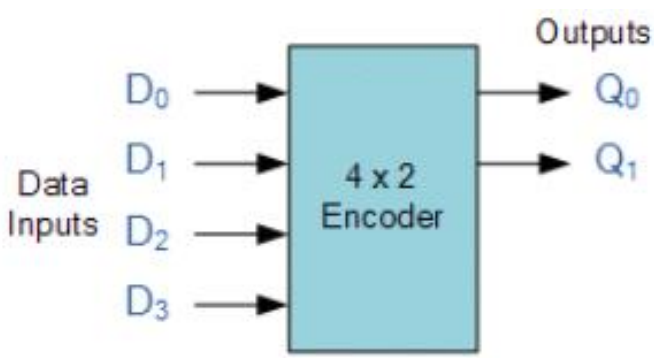


Output Select		Data Output Selected
a	b	
0	0	A
0	1	B
1	0	C
1	1	D

Encoder

- **Digital Encoder** more commonly called a **Binary Encoder** takes its inputs one at a time and then converts them into encoded output.

What if more than one inputs are high??



Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

The output lines of a digital encoder generate the binary equivalent of the input line whose value is equal to “1”

Priority Encoder

The *priority encoders* output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.

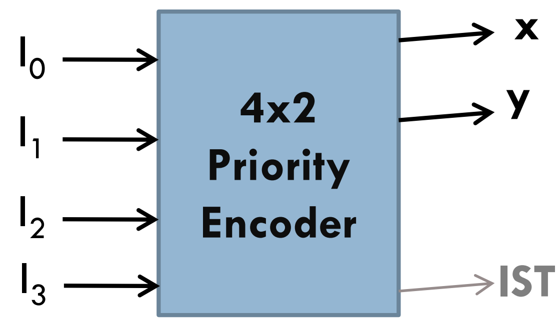
$$X = I_0' I_1'$$

$$Y = I_0' I_1 + I_0' I_2'$$

$$IST = I_0 + I_1 + I_2 + I_3$$

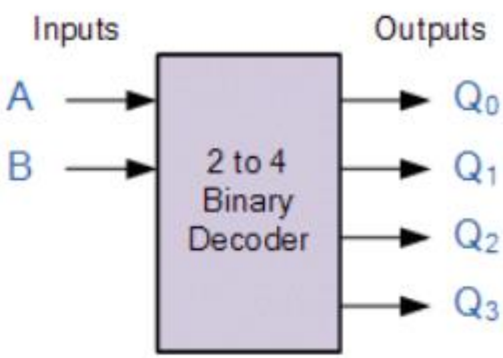
Inputs				Outputs		
I_0	I_1	I_2	I_3	X	Y	IST
1	×	×	×	0	0	1
0	1	×	×	0	1	1
0	0	1	×	1	0	1
0	0	0	1	1	1	1
0	0	0	0	×	×	0

Input with low index is having higher priority



Decoder

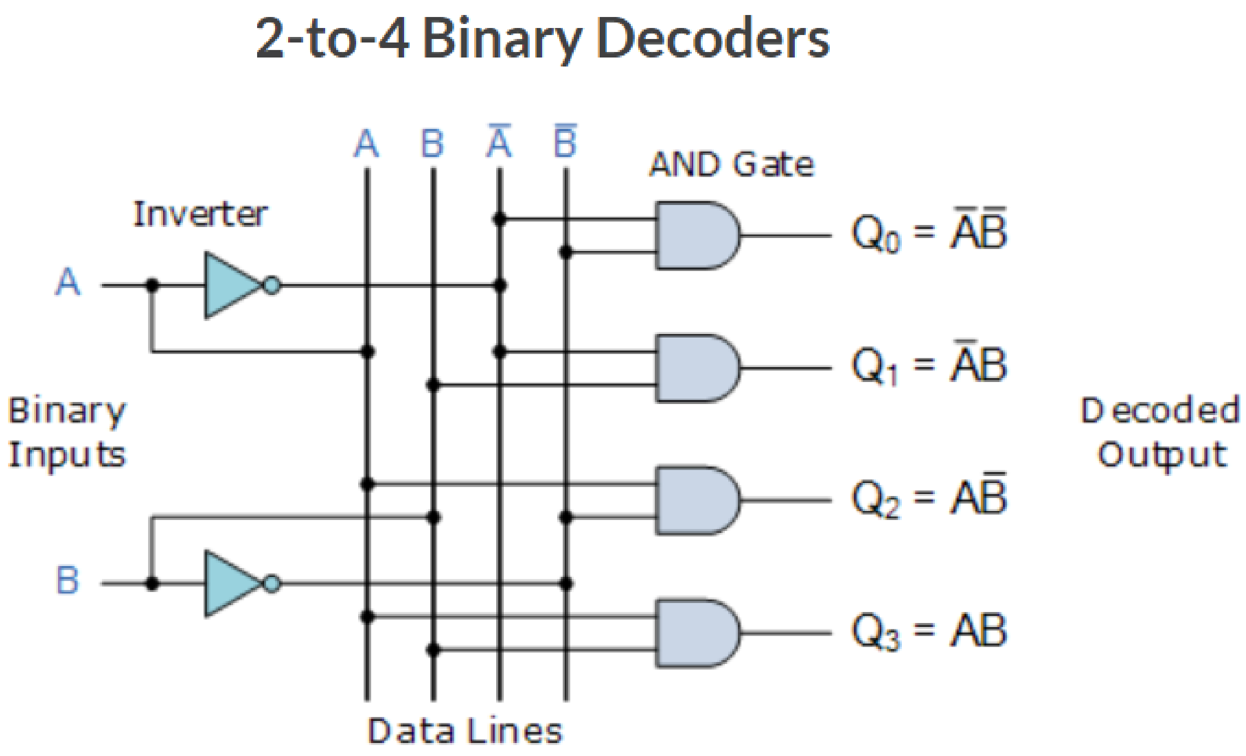
- The name "Decoder" means to translate or decode coded information from one format into another, so a binary decoder transforms "n" binary input signals into an equivalent code using 2^n outputs.



Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Decoder

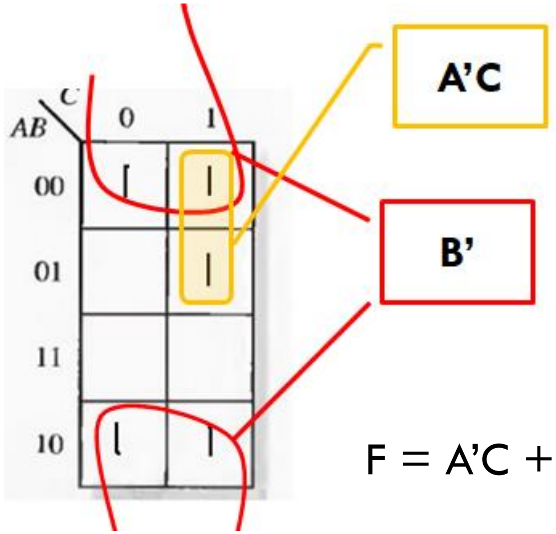


2-to-4 Binary Decoders
A
B
$\bar{A}$
$\bar{B}$
AND Gate
Inverter
A
$Q_0 = \bar{A}\bar{B}$
$Q_1 = \bar{A}B$
Binary Inputs
Decoded Output
$Q_2 = A\bar{B}$
B
$Q_3 = AB$
Data Lines

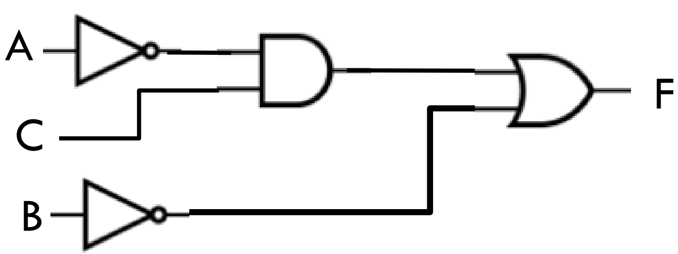
Exercise

- Realize the following function
  $F(A,B,C) = \sum(0,1,3,4,5) = A'B'C' + A'B'C + A'BC + AB'C' + AB'C$

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



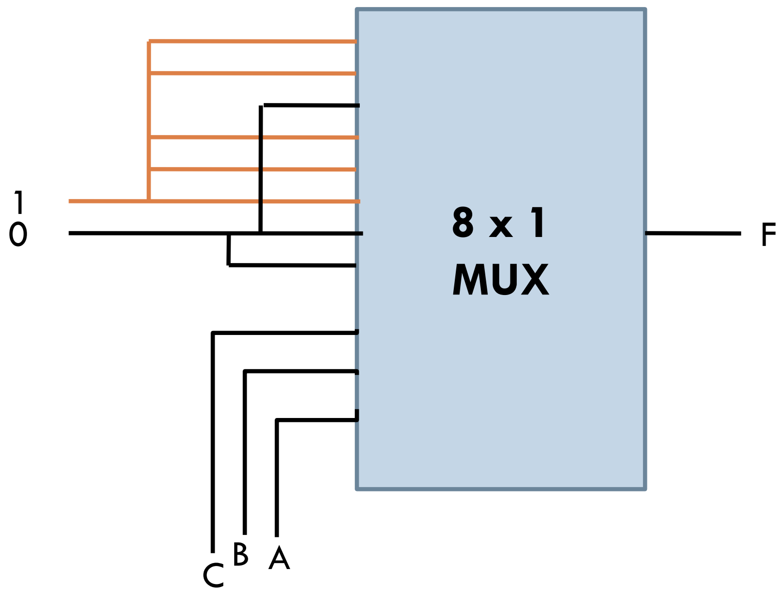
$F = A'C + B'$



Exercise

- Realize the following function with 8x1 MUX
  F(A,B,C) = ∑(0,1,3,4,5)

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



- Realize the following function with 4x1 MUX
  F(A,B,C) = ∑(0,1,3,4,5)

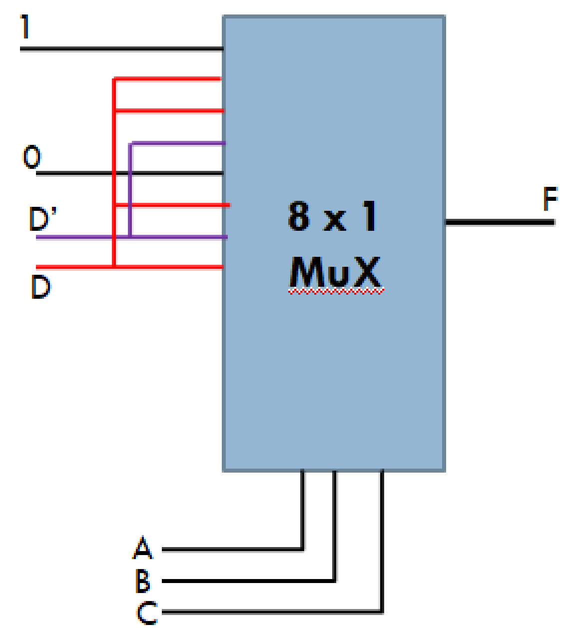
Exercise

Realize the following function with the help of 8x1 MUX

$F=\sum(0,1,3,5,611,12,15)$

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

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



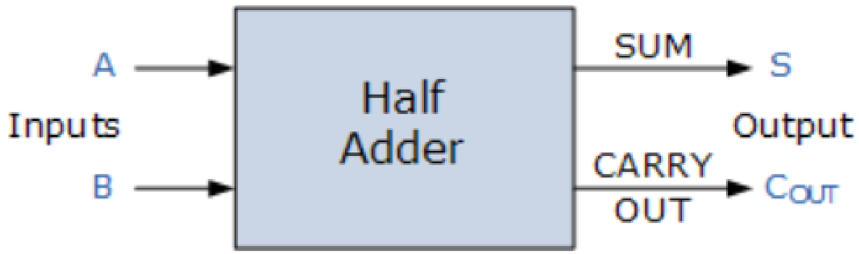
Exercise

- 1) Design Full adder with the help of Multiplexers
- 2) Implement following using Decoder

$F=\sum(0,2,4,6,7,13,15)$

Half Adder

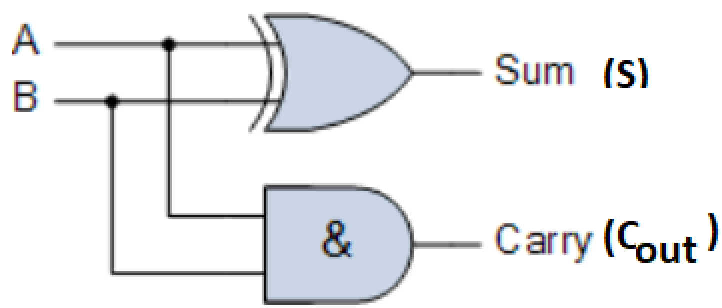
- Half Adder is a combinational Circuit which adds two inputs A and B and produces two out puts Sum (S) and Carry (C_{out})



Input		Output	
A	B	S	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

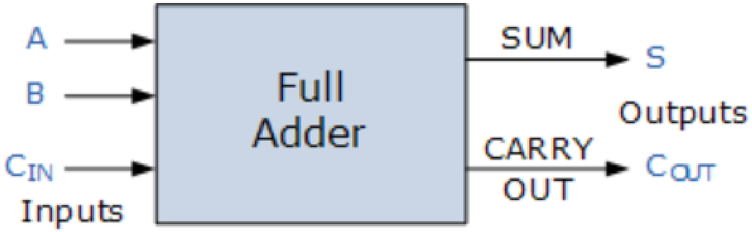
$S = A'B + AB'$ → A XOR B → $A \oplus B$

$C_{out} = AB$



Full Adder

- Full Adder is the combinational Circuit which adds three input bits A, B and C_{in} and produces two outputs S and C_{out}



$S = A'B'C_{in} + A'BC'_{in} + AB'C'_{in} + ABC_{in}$

$C_{out} = A'BC_{in} + AB'C_{in} + ABC'_{in} + ABC_{in}$

$S = A \oplus B \oplus C$

$C_{out} = A.B + C_{in}(A \oplus B)$

Input			Output	
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

Full Adder

$S = A'B'C_{in} + A'BC'_{in} + AB'C'_{in} + ABC_{in}$

$= (A'B'+AB)\ C_{in} + (A'B + AB')\ C'_{in}$

$= (A'B'+AB)\ C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= [(A'B') + (BB') + (AB) + (AA')]\ C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= [(A+B').(A'+B)]\ C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= [(A'.B'')'.(A''.B')']\ C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= [(A'B)'.(AB')']\ C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= [((A'.B)'' + (A.B')'')']C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= [((A'.B) + (A.B'))']\ C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= (A \text{ XOR } B)'\ C_{in} + (A \text{ XOR } B)\ C'_{in}$

$= (A \text{ XOR } B) \text{ XOR } C_{in}$

$= A \text{ XOR } B \text{ XOR } C_{in}$

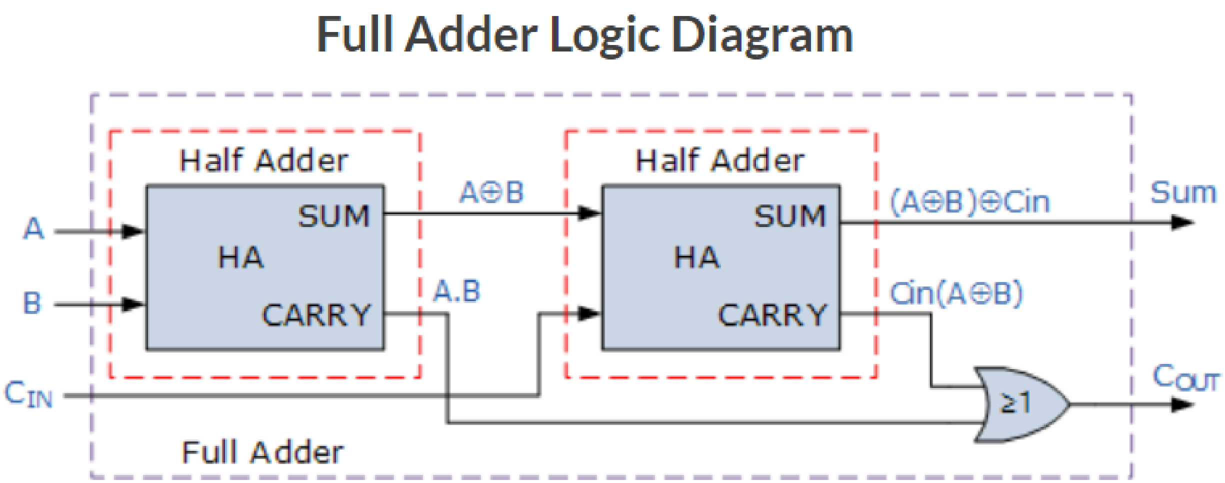
Full Adder

$$C_{out} = A'BC_{in} + AB'C_{in} + ABC'_{in} + ABC_{in}$$

$$= (A'B + AB')\,C_{in} + AB\,(C'_{in} + C_{in})$$

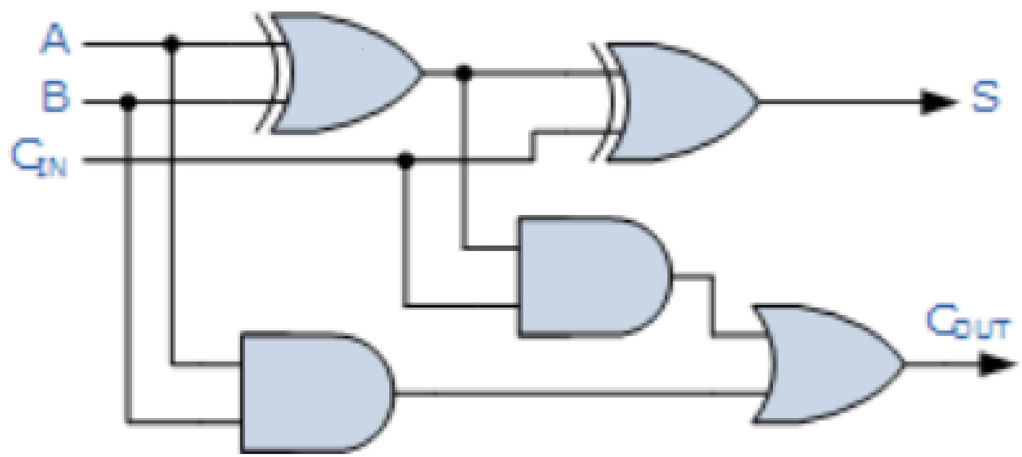
$$= (A \text{ XOR } B)\,C_{in} + AB$$

Full Adder



$S = A \oplus B \oplus C$

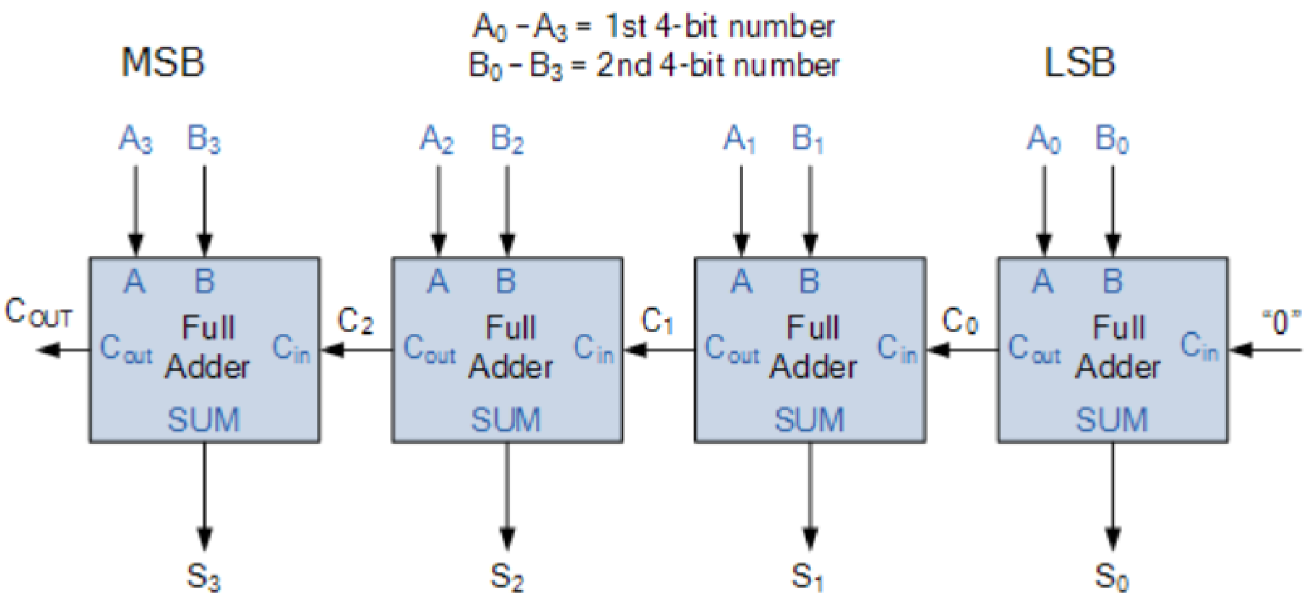
$C_{out} = A.B + C_{in}(A \oplus B)$



Ripple Adder

- n bit Ripple Adder: A “ripple carry adder” is simply “n“, full adders cascaded together with each full adder representing a single weighted column in a long binary addition. It is called a ripple carry adder because the carry signals produce a “ripple” effect through the binary adder from right to left, (LSB to MSB).
- For example, suppose we want to “add” together two 4-bit numbers, the two outputs of the first full adder will provide the first place digit sum (S) of the addition plus a carry-out bit that acts as the carry-in digit of the next binary adder.
- The second binary adder in the chain also produces a summed output (the 2nd bit) plus another carry-out bit and we can keep adding more full adders to the combination to add larger numbers, linking the carry bit output from the first full binary adder to the next full adder, and so forth.

4- bit Ripple Carry Adder



A 4-bit Ripple Carry Adder

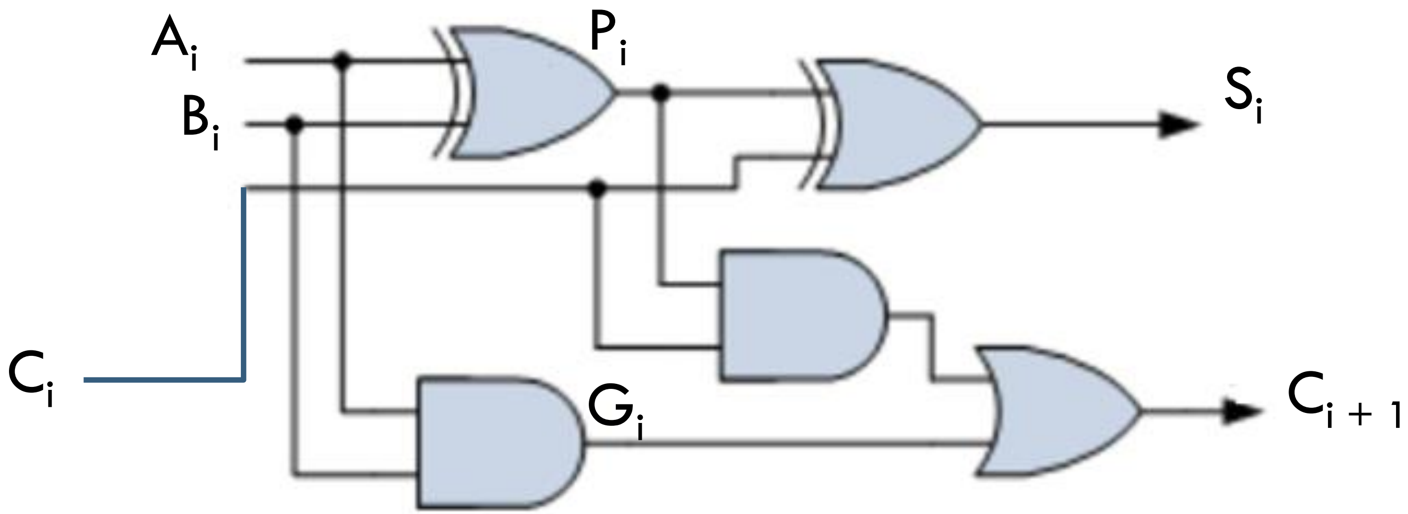
Ripple Carry Adder- Disadvantage

- One main disadvantage of “cascading” together full adders to add large binary numbers is that if inputs A and B change, the sum at its output will not be valid until any carry-input has “rippled” through every full adder in the chain because the MSB (most significant bit) of the sum has to wait for any changes from the carry input of the LSB (less significant bit). Consequently, there will be a finite delay before the output of the adder responds to any change in its inputs resulting in a accumulated delay.

- When the size of the bits being added is not too large for example, 4 or 8 bits, or the summing speed of the adder is not important, this delay may not be important. However, when the size of the bits is larger for example 32 or 64 bits used in multi-bit adders, or summation is required at a very high clock speed, this delay may become prohibitively large with the addition processes not being completed correctly within one clock cycle.

Carry Look Ahead Binary Adder

- Consider the Circuit of full addres



$Pi = Ai \oplus Bi$

$Gi = Ai\ Bi$

Output Sum and Carry can be expressed as

$Si = Pi \oplus Ci$

$Ci+1 = Gi + Pi\ Ci$

Carry Look Ahead Binary Adder

- Gi is called carry generator and it produces a carry of 1 when both Ai and Bi are 1, regardless of the input carry Ci.
- Pi is called a carry propagation because it is the term associated with the propagation of carry from Ci to Ci+1

C0 = input carry

C1 = G0 + P0C0

C2= G1 + P1C1= G1 + P1(G0 + P0C0) = G1 + P1G0 + P1P0C0

C3 = G2 + P2C2 = G2 + P2G1+ P2P1G0 + P2P1P0C0

Since the boolean function for each output carry is expressed in sum of products, each function can be implemented with one level of AND gates followed by an OR gate(or by two levels NANAD). The three boolean functions for C1, C2, and C3 are implemented in the Carry lookahead generator shown in figure.

Note that C3 does not have to wait for C2 and C1 to propagate. C3 is propagated at the same time as C1 and C2

Carry Look Ahead Binary Adder

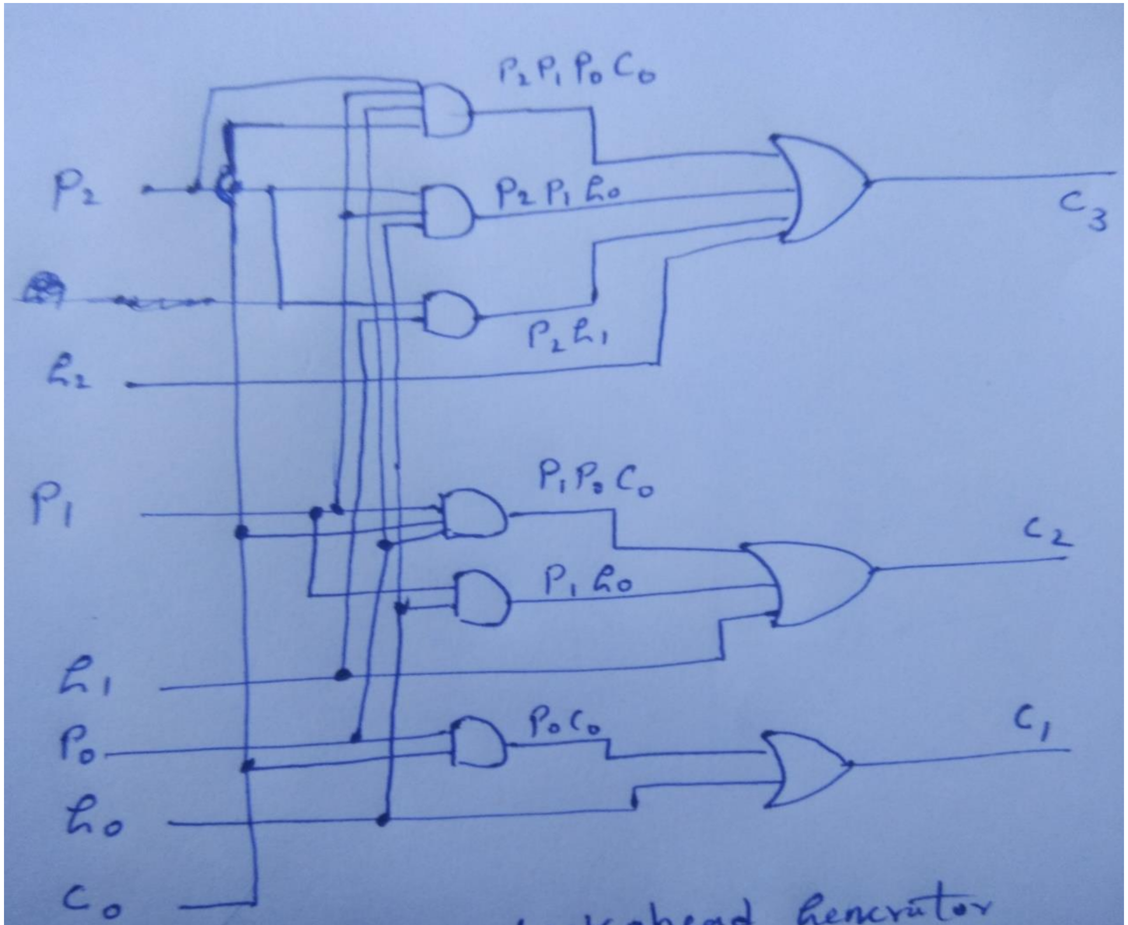


Figure: Carry lookahead Generator

Carry Look Ahead Binary Adder

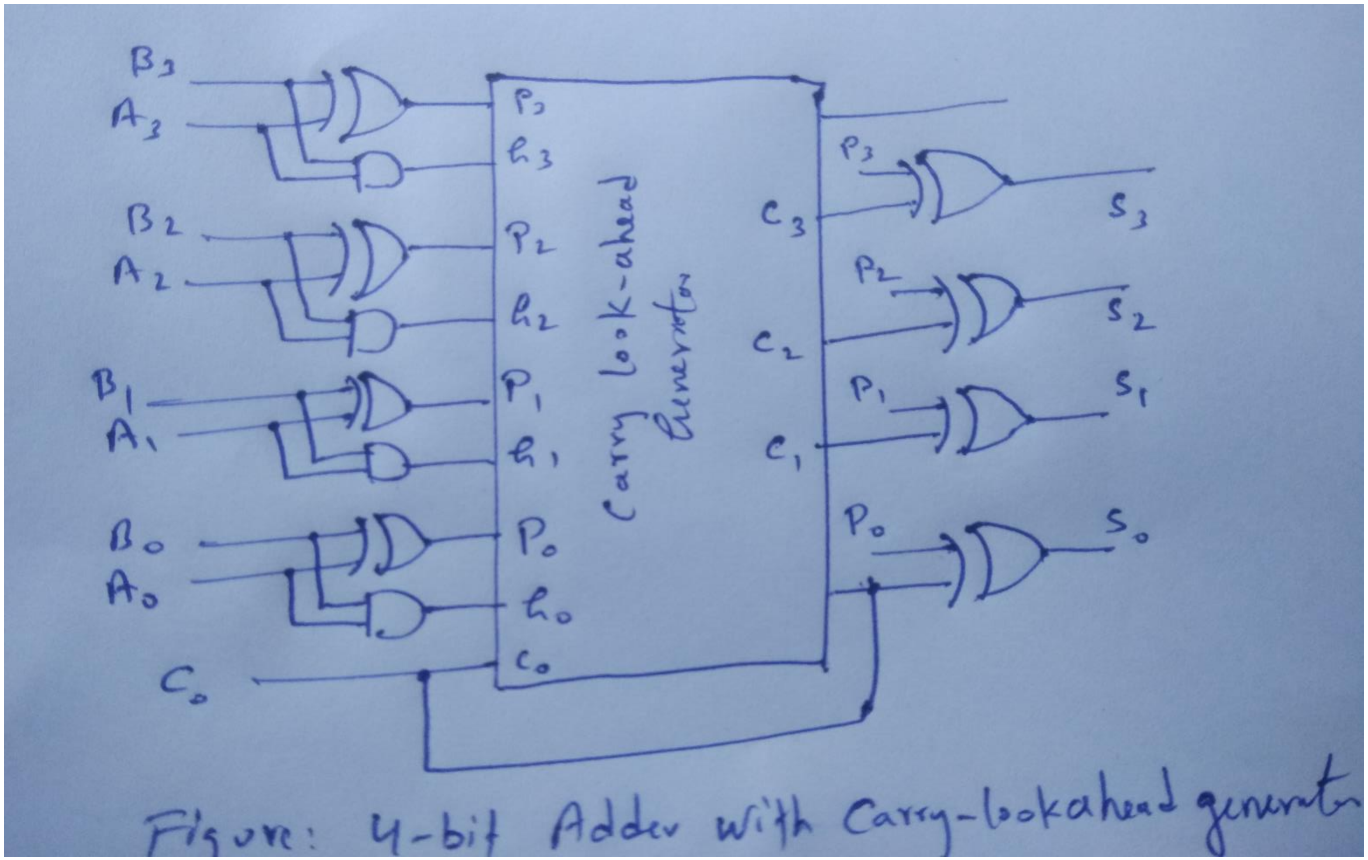


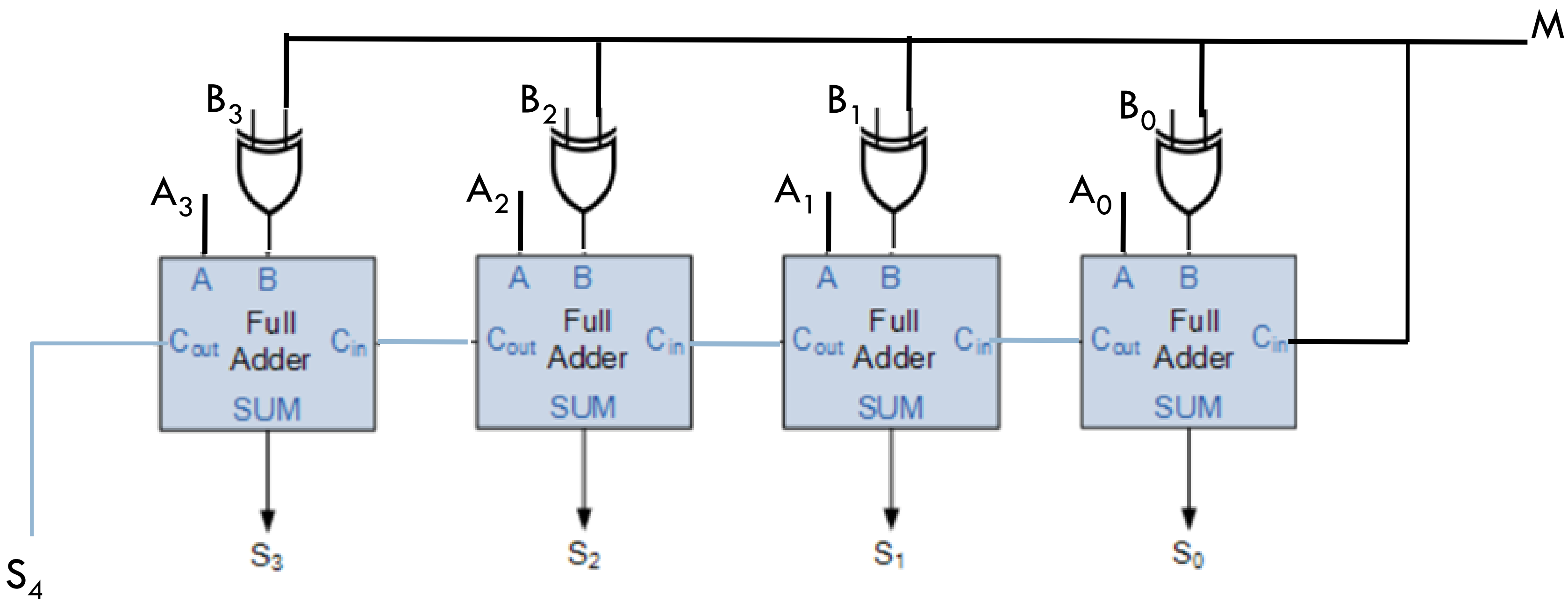
Figure: 4-bit Adder with Carry-lookahead generator

- All output carries are generated after a delay through two levels of Gates. Thus, S1 through S2 have equal propagation delay times. The Two level circuit for C4 is not shown

Adder-Subtractor Circuit

- The subtraction of unsigned numbers can be done by means of complements. Subtraction of A- B can be done by taking 2's complement of B and adding it to A

Adder-Subtractor Circuit



M signal is given as low(0) when circuit is to be used as binary adder and it is given as high(1) when circuit is to be used as subtractor.

When M=0 → S= A+B

When M= 1→ S= A + (2's Complement of B) = S= A-B

Adder-Subtractor Circuit

- Output V is for detecting overflow.
  - Overflow occurs when carry into the msb is not equal to carry out of msb

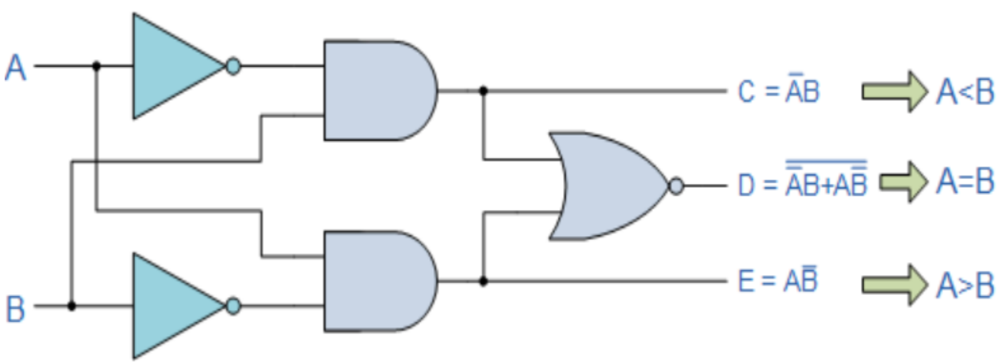
BCD to 7 Segment Decoder

- Self Study

Digital Comparator

- Comparators are made up from standard AND, NOR and NOT gates that compare the digital signals present at their input terminals and produce an output depending upon the condition of those inputs.

1-bit Digital Comparator Circuit

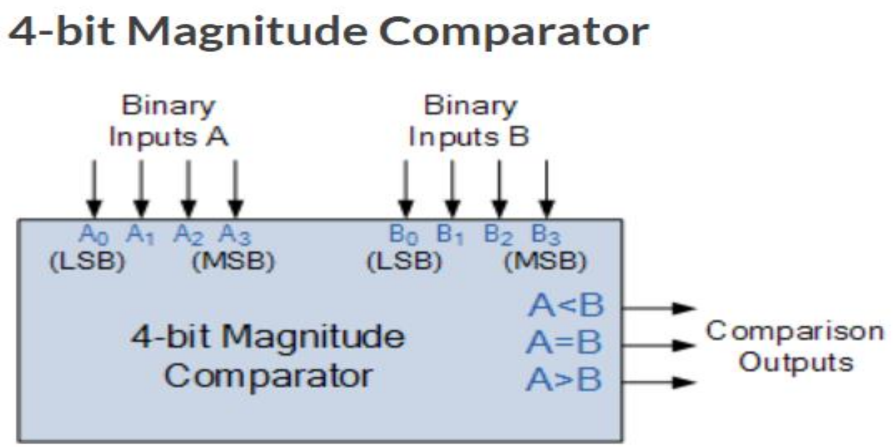


Digital Comparator Truth Table

Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Digital Comparator

- we can design larger bit comparators by cascading together n of these and produce a n-bit comparator. Multi-bit comparators can be constructed to compare whole binary or BCD words to produce an output if one word is larger, equal to or less than the other.



Some commercially available digital comparators such as the TTL 74LS85 or CMOS 4063 4-bit magnitude comparator have additional input terminals that allow more individual comparators to be "cascaded" together to compare words larger than 4-bits with magnitude comparators of "n"-bits being produced. These cascading inputs are connected directly to the corresponding outputs of the previous comparator

Exercise

- Design a Combinational Circuit with three inputs and one output. The output is 1 when the binary value of inputs is less then 3.
- Solution:

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

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

Exercise

- Design a binary Multiplier that multiplies two 2-bit numbers use AND gate and Half Adders

More Combinational Circuits

- BCD to Decimal Decoder
- BCD to Seven Segment Decoder
- Decimal to BCD encoder
- Octal to Binary Priority Encoder
- Parity Generator/Checker
- .
- .

References

- M. Morris .Mano, Digital Design, Pearson, 2016
- D. K. Kaushik, Digital Electronics, D. R. Publ., 2005
- Floyed, Digital Fundamentals, 10th Ed, Pearson, 2011
- https://www.electronics-tutorials.ws/logic/logic_1.html

Thank you