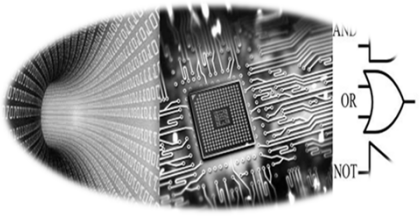
DIGITAL ELECTRONICS

Lecture 06: Combinational Logic



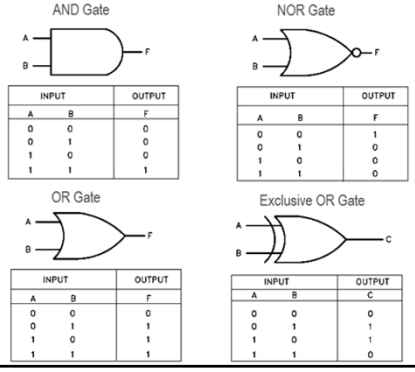
Dr. Tushar Kanti Bera

Department of Electrical Engineering
National Institute of Technology Durgapur (NITD)
Mahatma Gandhi Rd, A-Zone, Durgapur, West Bengal 713209, India

Date: Jan-Jun, 2021

Combinational Logic

- Standard Truth Tables can be converted into boolean logic circuit very easily.



INPUT A	INPUT B	OUTPUT F
0	0	0
0	1	0
1	0	0
1	1	1

INPUT A	INPUT B	OUTPUT F
0	0	1
0	1	0
1	0	0
1	1	0

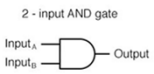
INPUT A	INPUT B	OUTPUT F
0	0	0
0	1	1
1	0	1
1	1	1

INPUT A	INPUT B	OUTPUT C
0	0	0
0	1	1
1	0	1
1	1	0

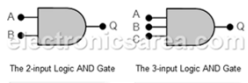
Combinational Logic

- Standard Truth Tables can be converted into boolean logic circuit very easily.

2 - input AND gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



The 2-input Logic AND Gate

Truth Table			
B	A	Q	
0	0	0	
0	1	0	
1	0	0	
1	1	1	

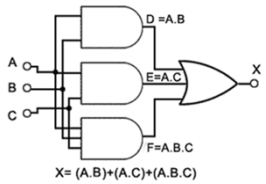
The 3-input Logic AND Gate

Truth Table			
C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Why Minterm and Maxterm?

- Developing a digital logic circuit using logic gates is very straight forward when a Boolean Expression is given.

X= (A.B)+(A.C)+(A.B.C)



X= (A.B)+(A.C)+(A.B.C)

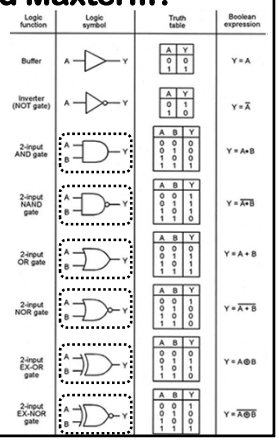
- Even for a large or complex BE the design is possible

$$f(A,B,C) = \left(\overline{AB + AB\bar{C} + BCD + \bar{E}(A+\bar{B}) + A\bar{B}E}\right)F$$

IS IT POSSIBLE TO DESIGN A DIGITAL CIRCUIT FROM A TRUTH TABLE?

Why Minterm and Maxterm?

- The function of each logic gates could be possible to represent through its Truth Table.
- Hence:
  Designing a digital circuit from a Truth Table could be possible for a known or standard digital logic.
- Thus, if a Truth table is provided the gate could be designed.



Combinational Logic: Ex 01

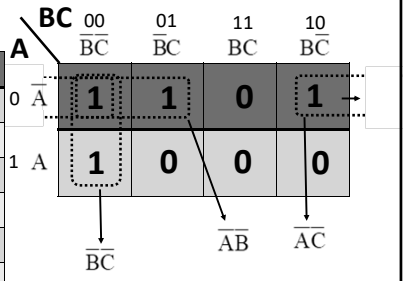
- Design a logic circuit with three input variables producing logic high for more than one inputs are at logic 0.

Inputs			Output
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Combinational Logic: Ex 01

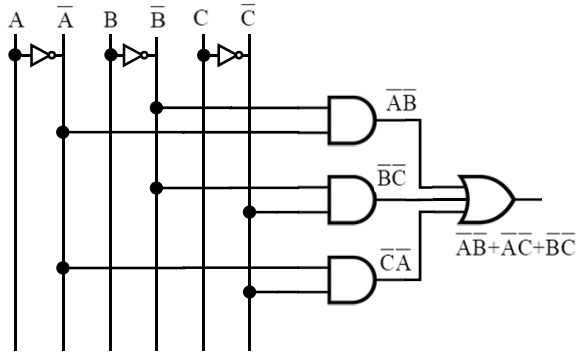
Boolean expression from Truth Table:

Inputs			Output
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0



$f(A,B,C) = \overline{A}\overline{B}+\overline{A}\overline{C}+\overline{B}\overline{C}$

Combinational Logic: Ex 01



$f = \overline{A}\overline{B}+\overline{A}\overline{C}+\overline{B}\overline{C}$

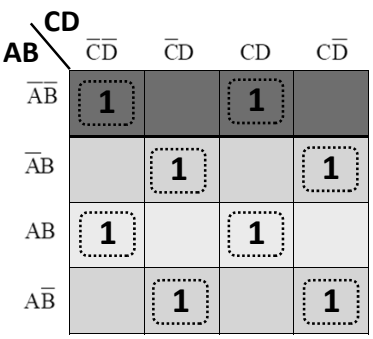
Combinational Logic: Ex 02

- Design a logic circuit when the no. of high inputs are even

Inputs				Output
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
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	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Combinational Logic: Ex 02

Inputs				Output
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
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	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



$f(A,B,C) = \overline{A}\overline{B}\overline{C}\overline{D}+AB\overline{C}\overline{D}+\overline{A}B\overline{C}D+A\overline{B}\overline{C}D$
$+\overline{A}\overline{B}CD+ABCD+\overline{A}BC\overline{D}+A\overline{B}C\overline{D}$

4 Variables K-Map: Example 02

$f(A,B,C) = A\overline{B}\overline{C}\overline{D} + AB\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D$

$+\overline{A}\overline{B}CD + ABCD + \overline{A}BC\overline{D} + ABC\overline{D}$

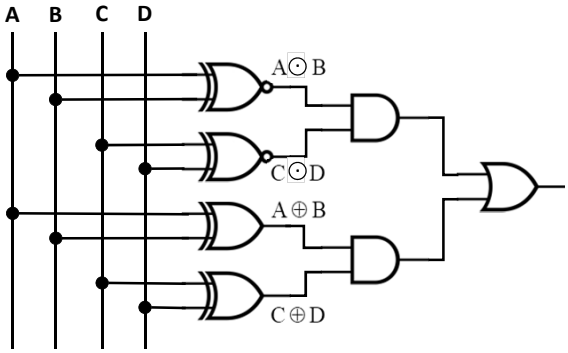
$f(A,B,C) = \overline{C}\overline{D}(\overline{A}\overline{B}+AB) + \overline{C}D(\overline{A}B+A\overline{B})$
$+ CD(\overline{A}\overline{B}+AB) + C\overline{D}(\overline{A}B+A\overline{B})$

$f(A,B,C) = \overline{C}\overline{D}(A \odot B) + \overline{C}D(A \oplus B)$
$+ CD(A \odot B) + C\overline{D}(A \oplus B)$

$f(A,B,C) = (A \odot B)(\overline{C}\overline{D}+CD) + (A \oplus B)(\overline{C}D+C\overline{D})$

$f(A,B,C) = (A \odot B)(C \odot D) + (A \oplus B)(C \oplus D)$

Combinational Logic: Ex 02



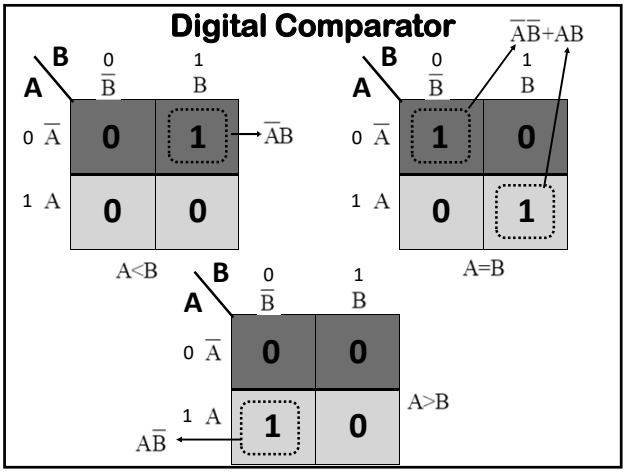
$f(A,B,C) = (A \odot B)(C \odot D) + (A \oplus B)(C \oplus D)$

Combinational Logic: Digital Comparator

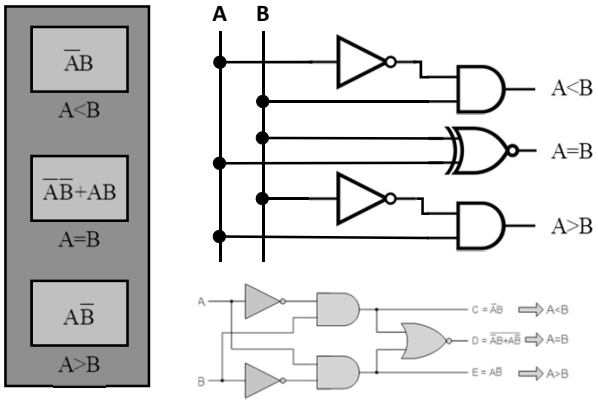
- Comparing a **Number (say, A)** with another **Number (say, B)**
- One bit comparison
  - A<B
  - A=B
  - A>B

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

Digital Comparator

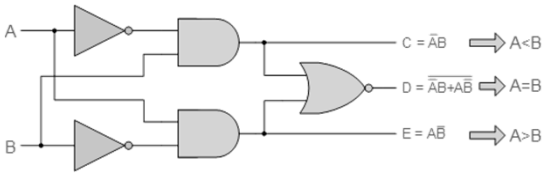


1-Bit Digital Comparator



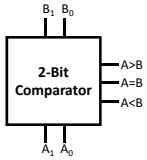
1-Bit Digital Comparator

- The Digital Comparator is another very useful combinational logic circuit used to compare the value of two binary digits



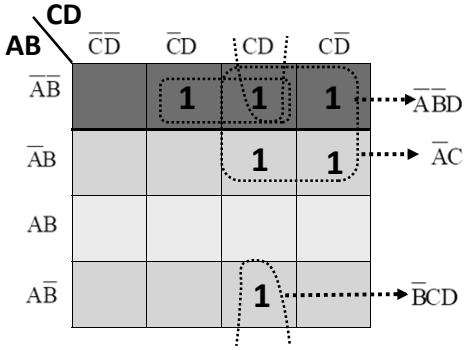
2-Bit Digital Comparator

- **2-Bit Comparator:** compares 2 bit binary data (A_1A_0 vs B_1B_0)



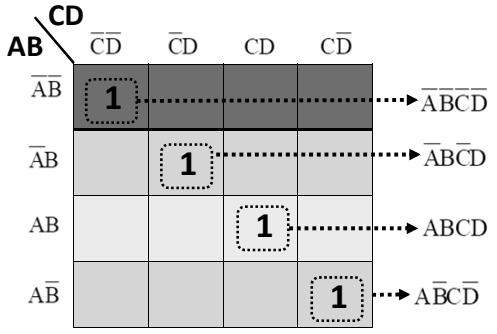
Inputs			
D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

2-Bit Digital Comparator



$$f(A,B,C) = \overline{A}\overline{B}D + \overline{A}B\overline{D} + A\overline{B}\overline{D}$$

2-Bit Digital Comparator



$$f(A,B,C)=\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}\overline{B}\overline{C}D+ABCD+A\overline{B}C\overline{D}$$

2-Bit Digital Comparator

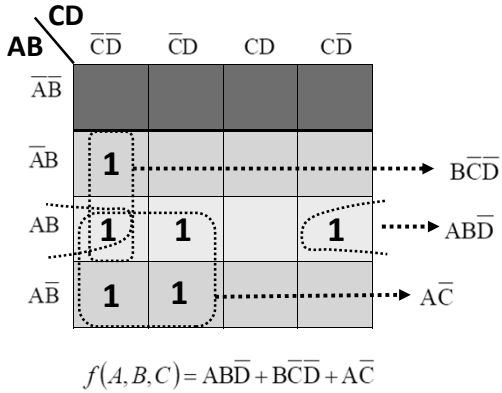
$$f(A,B,C)=\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}\overline{B}\overline{C}D+ABCD+A\overline{B}C\overline{D}$$

$$f(A,B,C)=\overline{A}\overline{C}\left(\overline{B}\overline{D}+BD\right)+AC\left(BD+\overline{B}\overline{D}\right)$$

$$f(A,B,C)=\left(\overline{B}\overline{D}+BD\right)\left(\overline{A}\overline{C}+AC\right)$$

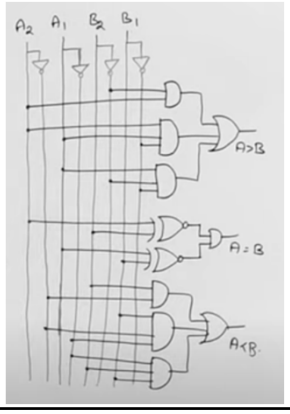
$$f(A,B,C)=(B\odot D)(A\odot C)$$

2-Bit Digital Comparator

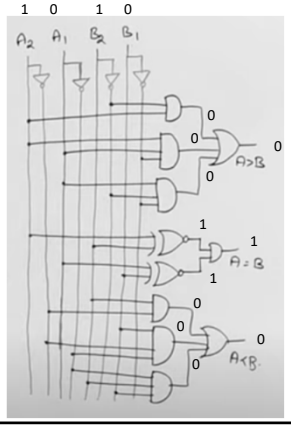


$$f(A,B,C)=AB\overline{D}+B\overline{C}\overline{D}+A\overline{C}$$

2-Bit Digital Comparator

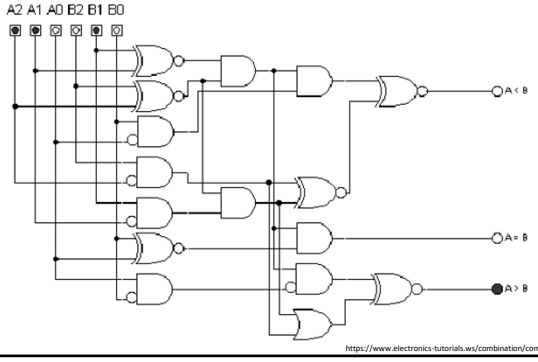


2-Bit Digital Comparator



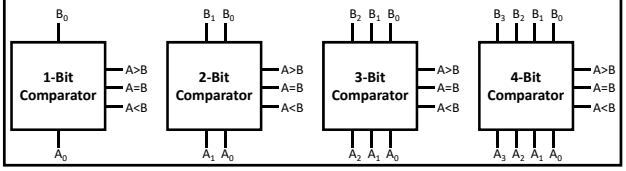
3-Bit Comparator: Assignment

- **Make the truth table of 3-Bit Comparator and derive the Boolean expression for the outputs and send it to me.**



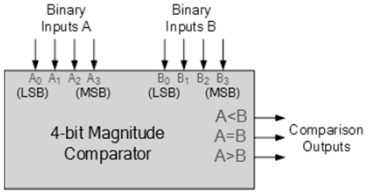
n-Bit Digital Comparator

- The Digital Comparator is another very useful combinational logic circuit used to compare the value of two binary digits.
- **1-Bit Comparator:** compares 1 bit binary data (A_0 vs B_0)
- **2-Bit Comparator:** compares 2 bit binary data (A_1A_0 vs B_1B_0)
- **3-Bit Comparator:** compares 3 bit binary data ($A_2A_1A_0$ vs $B_2B_1B_0$)
- **4-Bit Comparator:** compares 4 bit binary data ($A_3A_2A_1A_0$ vs $B_3B_2B_1B_0$)



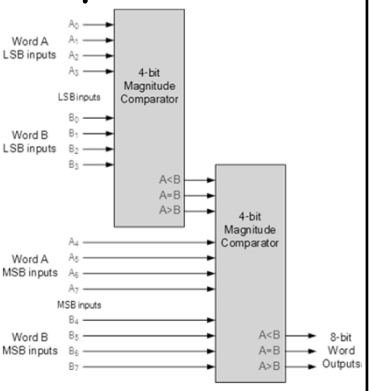
4-Bit Digital Comparator

- As well as comparing individual bits, we can design larger bit comparators by cascading together n of these and produce a n-bit comparator just as we did for the n-bit adder in the previous tutorial. 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.
- A very good example of this is the 4-bit **Magnitude Comparator**. Here, two 4-bit words ("nibbles") are compared to each other to produce the relevant output with one word connected to inputs A and the other to be compared against connected to input B as shown below.



Word Comparator

- 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 as shown to compare 8, 16 or even 32-bit words.



Combinational Logic: Adder Circuit

- As the name suggests *half-adder* is an arithmetic circuit block by using this circuit block we can be used to add two bits.
- As we know it can add two bit number so it has two inputs terminals and as well as two outputs terminals, with one producing the SUM output and the other producing the CARRY.

- Addition in Binary System:
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 10
- Addition in Binary System:
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 0 with a carry 1

Combinational Logic: Adder Circuit

Addition in Binary System:

- **0 + 0 = 0**
- **0 + 1 = 1**
- **1 + 0 = 1**
- **1 + 1 = 0** with a carry 1

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

A	B	Sum
0	0	0
0	1	1
1	0	1
1	1	0

Summing Logic

A	B	Carry
0	0	0
0	1	0
1	0	0
1	1	1

Carry generation logic

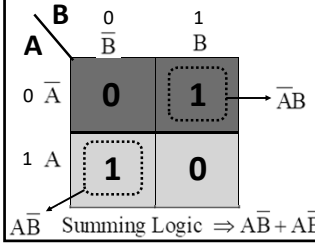
Adder Circuit: Half Adder

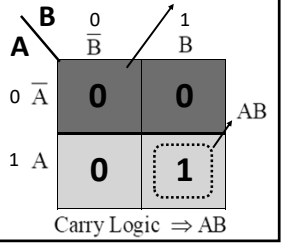
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Summing Logic

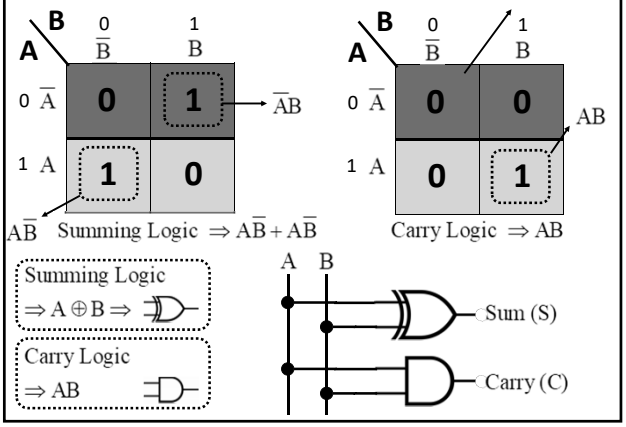
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Carry generation logic

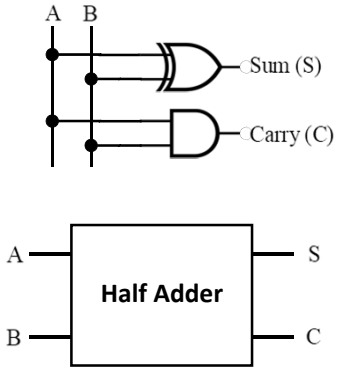




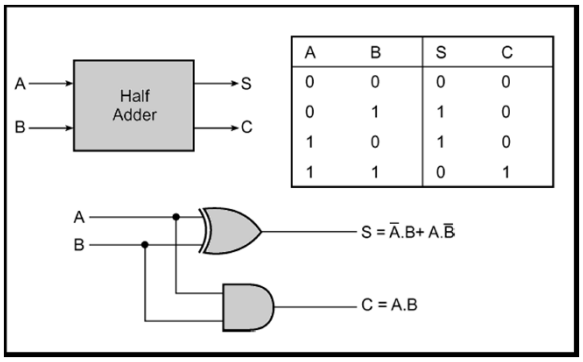
Adder Circuit: Half Adder



Adder Circuit: Half Adder



Adder Circuit: Half Adder

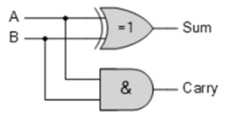


A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

http://www.electronicsengineering.nbcafe.in/half-adder-and-full-adder-circuit/

Half Adder: Summary

Half Adder Truth Table with Carry-Out

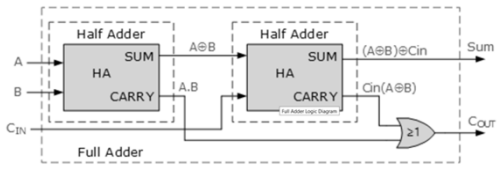
Symbol	B	A	SUM	CARRY
	0	0	0	0
	0	1	1	0
	1	0	1	0
	1	1	0	1

https://www.electronics-tutorials.ws/combination/comb_7.html

Adder Circuit: Full Adder

- The main difference between the **Full Adder** and the previous **Half Adder** is that a full adder has three inputs.
- The same two single bit data inputs A and B as before plus an additional *Carry-in* (C-in) input to receive the carry from a previous stage as shown below.

Adder Logic Diagram

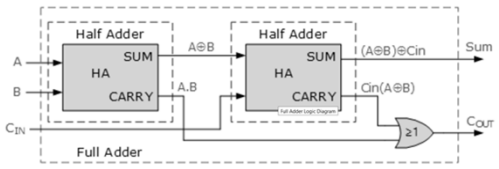


https://www.electronics-tutorials.ws/combination/comb_7.html

Transistor-Resistance Logic: NOR Gate

- The main difference between the **Full Adder** and the previous **Half Adder** is that a full adder has three inputs. The same two single bit data inputs A and B as before plus an additional *Carry-in* (C-in) input to receive the carry from a previous stage as shown below.

Full Adder Logic Diagram

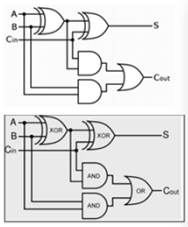


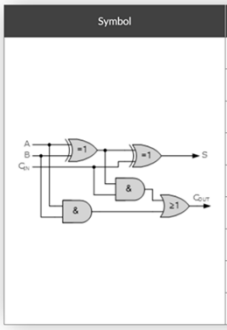
https://www.electronics-tutorials.ws/combination/comb_7.html

Adder Circuit: Full Adder

- Abc

Full Adder Truth Table with Carry





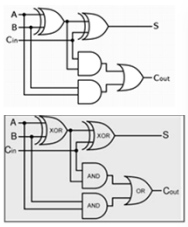
Symbol	Truth Table				
	C-in	B	A	Sum	C-out
	0	0	0	0	0
	0	0	1	1	0
	0	1	0	1	0
	0	1	1	0	1
	1	0	0	1	0
	1	0	1	0	1
	1	1	0	0	1
	1	1	1	1	1

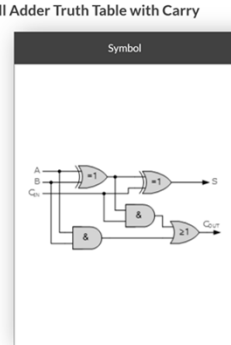
https://www.electronics-tutorials.ws/combination/comb_7.html

Adder Circuit: Assignment

- **Using the K-Map derive the Boolean expression for a full adder circuit and send it to me.**

Full Adder Truth Table with Carry



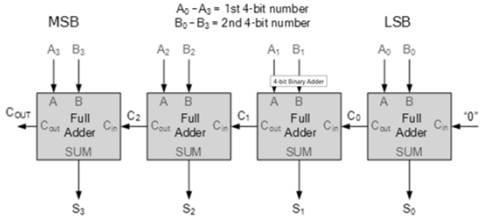


Symbol	Truth Table				
	C-in	B	A	Sum	C-out
	0	0	0	0	0
	0	0	1	1	0
	0	1	0	1	0
	0	1	1	0	1
	1	0	0	1	0
	1	0	1	0	1
	1	1	0	0	1
	1	1	1	1	1

https://www.electronics-tutorials.ws/combination/comb_7.html

4-Bit Full Adder

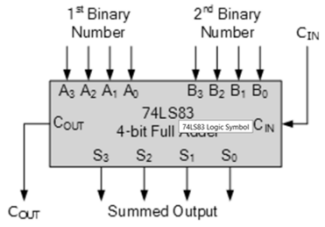
- We have seen above that single 1-bit binary adders can be constructed from basic logic gates. But what if we wanted to add together two n-bit numbers, then n number of 1-bit full adders need to be connected or "cascaded" together to produce what is known as a **Ripple Carry Adder**.
- A "ripple carry adder" is simply "n", 1-bit 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).



https://www.electronics-tutorials.ws/combination/comb_7.html

4-Bit Full Adder

- 4-bit full adder circuits with carry look ahead features are available as standard IC packages in the form of the TTL 4-bit binary adder 74LS83 or the 74LS283 and the CMOS 4008 which can add together two 4-bit binary numbers and generate a SUM and a CARRY output as shown.



https://www.electronics-tutorials.ws/combination/comb_7.html

Parity Generator

- **Parity: the fact of being even or odd.**
- Parity bit is an extra bit associated with any binary information to detect the error in the information.
- To detect the error the parity bit is added to the binary bits such that the total number of 1s are even or odd depending on the parity used.
- **Parity Generator**
  - Parity Generator is a digital circuit which generates the parity bit.
- **Parity Generator Classification**
  - **Even Parity Generator (EPV).**
    - Makes the total number of 1s in the data even (including parity bit)
    - That means, EPV generates 1 when total number of 1s is found odd in the binary data (ABCD or else)
  - **Odd Parity Generator.**
    - Makes the total number of 1s in the data even (including parity bit)
    - That means, OPV generates 1 when total number of 1s is found even in the binary data (ABCD or else)

Even Parity Generator

- Even Parity Generator It is a combinational logic circuit which generates the parity bit such that the number of 1s in the message become even.
- It checks the number of 1s in the message and generates a bit either 0 or 1 to make the total number of 1s in the data even (including parity bit).
- Let us consider a 4-bit binary data with a even Parity generator as shown in the truth table.

4-Bit Information				Even Parity
A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Example of a Combinatorial Circuit:
A Multiplexer (MUX)

Consider an integer 'm', which is constrained by the following relation:

$$m = 2^n,$$

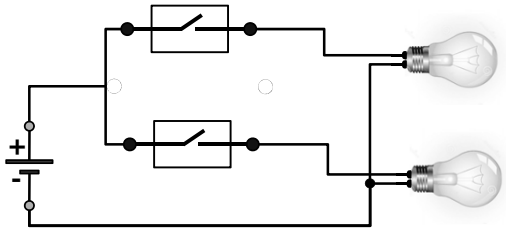
where m and n are both integers.

- **A m-to-1 Multiplexer** has
  - m Inputs: $I_0, I_1, I_2, \ldots\ldots\ldots\ldots I_{(m-1)}$
  - one Output: Y
  - n Control inputs: $S_0, S_1, S_2, \ldots\ldots S_{(n-1)}$
  - One (or more) Enable input(s)

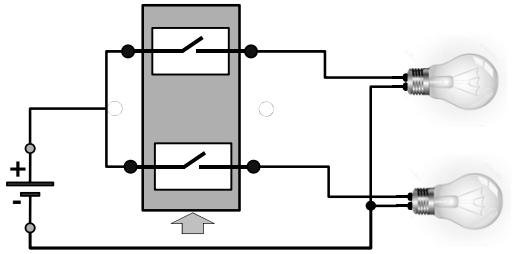
such that Y may be equal to one of the inputs, depending upon the control inputs.

43

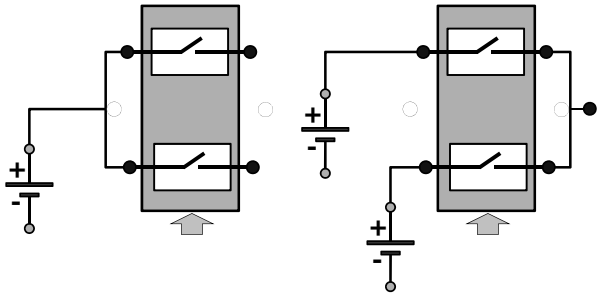
- **A switch is a electrical/electronic component which make or break a circuit.**



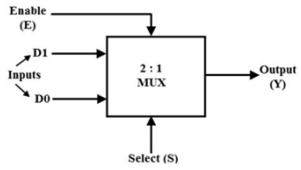
- **A switch is a electrical/electronic component which make or break a circuit.**



- **Demultiplexer**
- **Multiplexer**



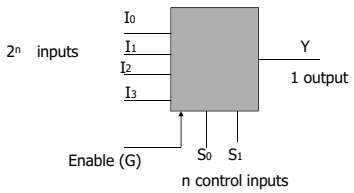
- **2:1 Multiplexer**



Selector	Output
0	D0
1	D1

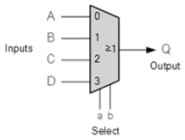
Example: A 4-to-1 Multiplexer

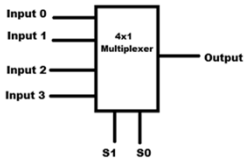
A 4-to-1 Multiplexer:



48

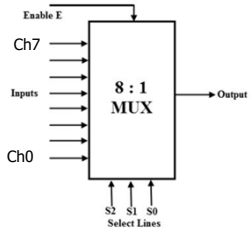
4:1 Multiplexer: Truth Table





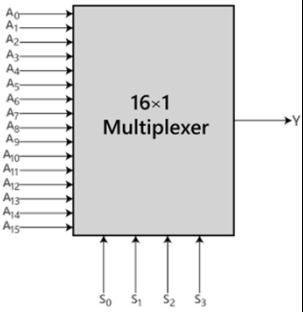
S1	S0	Output
0	0	Ch_0
0	1	Ch_1
1	0	Ch_2
1	1	Ch_3

4:1 Multiplexer



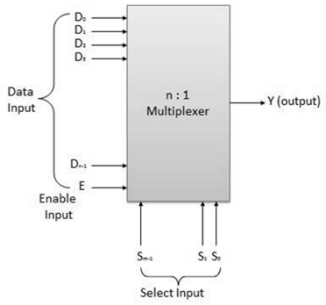
S2	S1	S0	Output
0	0	0	Ch_0
0	0	1	Ch_1
0	1	0	Ch_2
0	1	1	Ch_3
1	0	0	Ch_4
1	0	1	Ch_5
1	1	0	Ch_6
1	1	1	Ch_7

16:1 Multiplexer



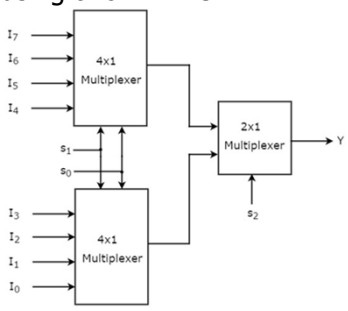
S3	S2	S1	S0	Output
0	0	0	0	Ch_0
0	0	0	1	Ch_1
0	0	1	0	Ch_2
0	0	1	1	Ch_3
0	1	0	0	Ch_4
0	1	0	1	Ch_5
0	1	1	0	Ch_6
0	1	1	1	Ch_7
1	0	0	0	Ch_8
1	0	0	1	Ch_9
1	0	1	0	Ch_{10}
1	0	1	1	Ch_{11}
1	1	0	0	Ch_{12}
1	1	0	1	Ch_{13}
1	1	1	0	Ch_{14}
1	1	1	1	Ch_{15}

n:1 Multiplexer



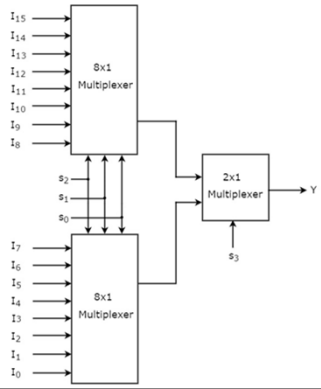
8:1 Multiplexer from 4:1 Multiplexers

- 8:1 MUX using two 4:1 MUX



16:1 Multiplexer from 8:1 Multiplexers

- 8:1 MUX using two 4:1 MUX



8:1 Multiplexer from 4:1 Multiplexers

- 8:1 MUX using two 4:1 MUX

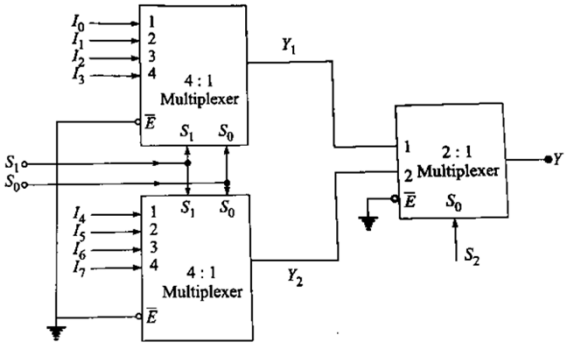
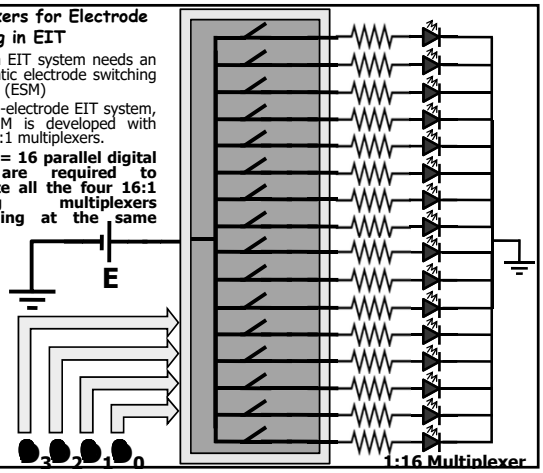


Fig. 4.56 8:1 multiplexer using 4:1 and 2:1 multiplexers

Multiplexers for Electrode Switching in EIT

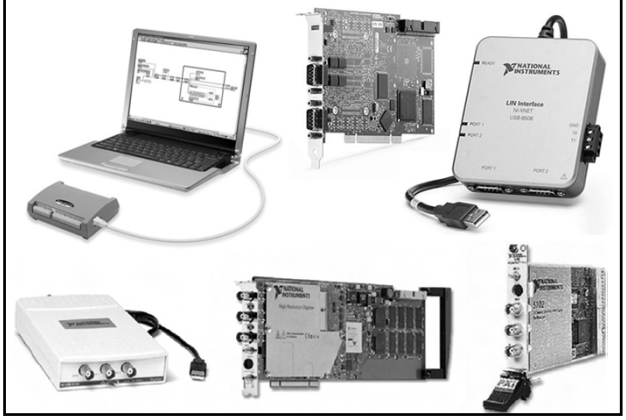
- Modern EIT system needs an automatic electrode switching module (ESM)
- In a 16-electrode EIT system, the ESM is developed with four 16:1 multiplexers.
- **4 × 4 = 16 parallel digital bits are required to operate all the four 16:1 analog multiplexers switching at the same time.**



LabVIEW Based Graphical User Interface (GUI)

57

NI Hardware: USB, PCI, PXI



Introduction to LabVIEW

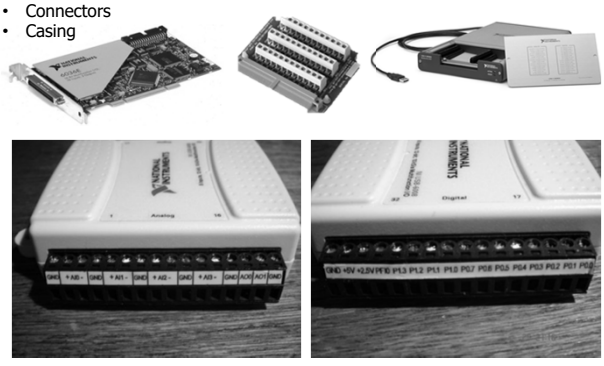
LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) is a system design platform and development environment for a visual programming language from

Software:

Hardware: NI DAQ or

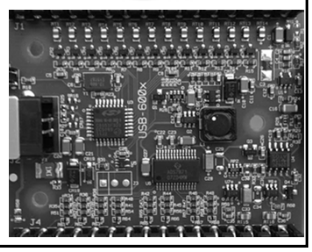
DAQ Accessories

- Electronics
- Connectors
- Casing

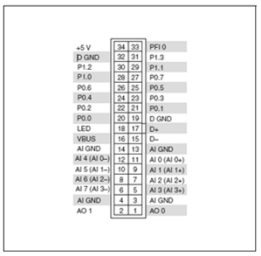


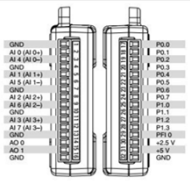
The analog I/O terminals with labels assuming differential signal

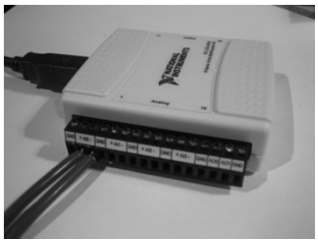
The digital I/O terminals


Ni USB-6009 provides basic data acquisition functions for simple data recording, portable measurement and university laboratory experiments







LabVIEW Applications

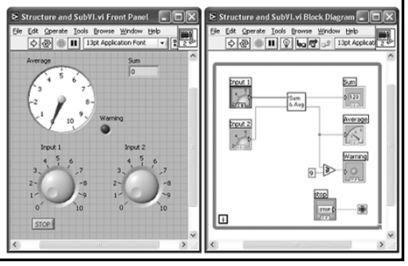
Of course, everything has a good and bad side, so you can't just brush off something as the best or worst. LabVIEW finds optimal usage in a couple of sectors, and is used, as mentioned earlier, all over the world. Here's some applications of LabVIEW:



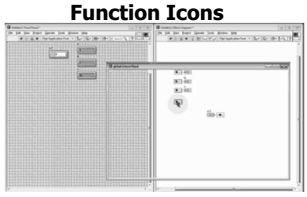
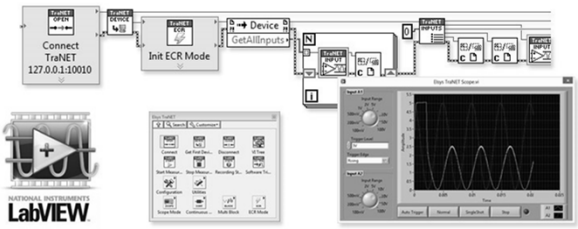
1. Wind turbine monitoring
2. Oil and gas pressure monitoring
3. Power quality monitoring
4. Machine control monitoring
5. Commercial and industrial robotics
6. Customized control and measurement
7. Machine condition monitoring
8. Data acquisition
9. Signal processing
10. Instrumentation control
11. Aerospace and transportation systems control and monitoring
12. Embedded systems testing
13. Device and component prototyping
14. Industrial automation
15. Weather monitoring and warning systems
16. UI design for instrumentation
17. Industrial and commercial result/output generation systems

LabVIEW Programming

- Front Panel
- Block Diagram

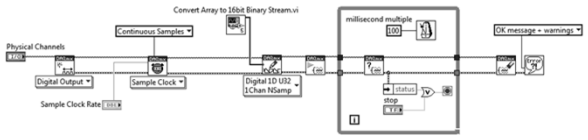


LabVIEW Programming: Dragging and Dropping the Function Icons

LabVIEW Programming: Uniqueness

- **Graphical programing + Conventional programing**



The Language

LabVIEW differs from most other development platforms in the regard that it depends on 'visual programming' more than actual coding. In simple words, you use either predefined or custom components to complete programming tasks. For example, to create a reiterating series, all you need to do is drag and drop the corresponding loop functionality onto the block diagram. Add your conditions, outputs, connections, and you're good to go!