ELECTRONICS
CLUB

Centre For Innovation

WELCOME TO ELECTROMANIA!

DAY - 2

ELECTRONICS
CLUB

Centre For Innovation

ELECTRONICS CLUB

Centre For Innovation

TABLE OF CONTENTS

ELECTRONICS CLUB

Centre For Innovation

01

LOGIC MINIMIZATION

ELECTRONICS CLUB

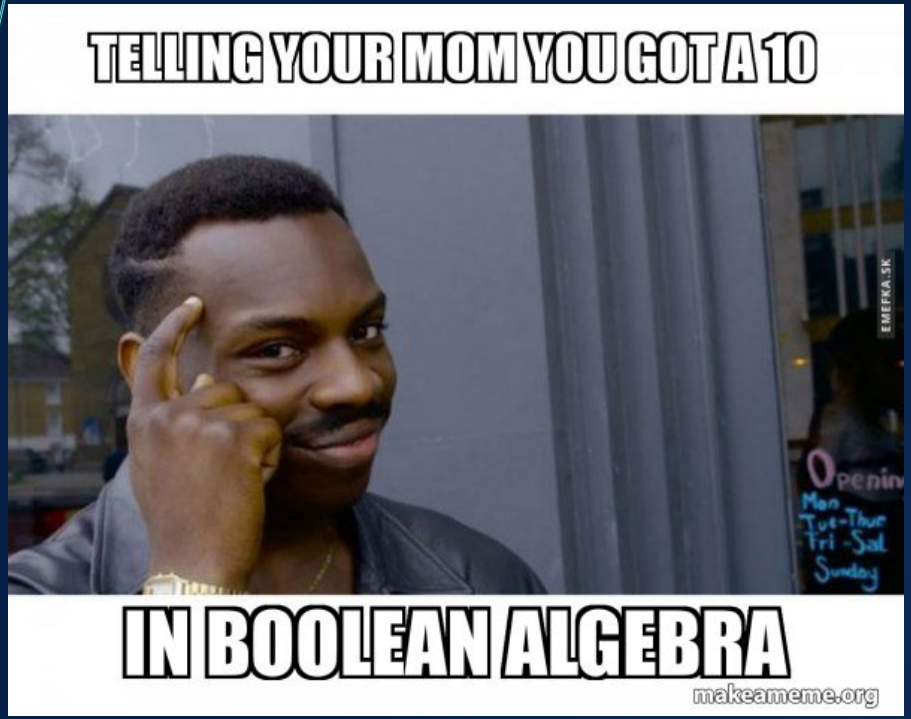
Centre For Innovation

The need for a Standard Method

- Gate-level minimization is the design task of finding an optimal gate-level implementation of the Boolean functions describing a digital circuit. This task is well understood, but is difficult to execute by manual methods when the logic has more than a few inputs.
- And the manual method is not standardized but depends on the person doing it. Hence different people might represent the same equation in different ways.
- So we need a standard method to keep the simplification uniform

ELECTRONICS CLUB

Centre For Innovation



ELECTRONICS CLUB

Why Gate Minimization?

Centre For Innovation

- Gate-level minimization is the design task of finding an optimal gate-level implementation of the Boolean functions describing a digital circuit
- The complexity of the digital logic gates that implement a Boolean function is directly related to the complexity of the algebraic expression from which the function is implemented.
- This is in-turn related to the cost of the IC needed for this computation

ELECTRONICS CLUB

Centre For Innovation

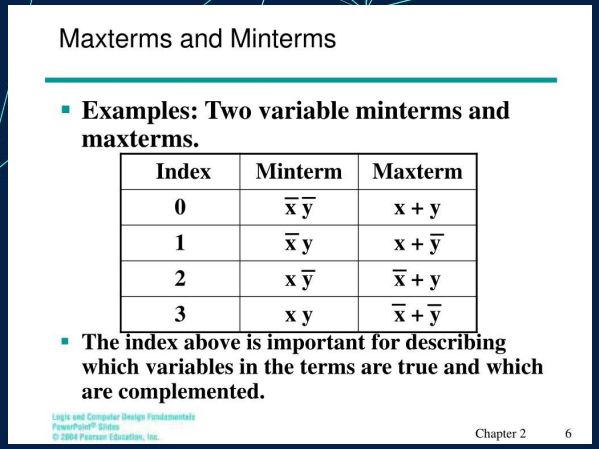
SOP and POS

- A binary variable may appear either in its normal form (x) or in its complement form (x'). Now consider two binary variables x and y combined with an AND operation.
- Since each variable may appear in either form, there are four possible combinations: xy, x'y, xy', and x'y'. Each of these four AND terms is called a minterm, or a standard product
- A symbol for each minterm is also shown in the table and is of the form mj , where the subscript j denotes the decimal equivalent of the binary number of the minterm designated.

ELECTRONICS CLUB

Centre For Innovation

SOP and POS



Maxterms and Minterms

- **Examples: Two variable minterms and maxterms.**

Index	Minterm	Maxterm
0	$\bar{x}\,\bar{y}$	$x + y$
1	$\bar{x}\,y$	$x + \bar{y}$
2	$x\,\bar{y}$	$\bar{x} + y$
3	$x\,y$	$\bar{x} + \bar{y}$

- **The index above is important for describing which variables in the terms are true and which are complemented.**

Logic and Computer Design Fundamentals PowerPoint® Slides © 2004 Pearson Education, Inc.

Chapter 2 6

- The minterms whose sum defines the Boolean function are those which give the 1's of the function in a truth table
- It is sometimes convenient to express a Boolean function in its sum-of-minterms form. If the function is not in this form, it can be made so by first expanding the expression into a sum of AND terms.
- In a similar fashion, n variables forming an OR term, with each variable being primed or unprimed, provide 2n possible combinations, called maxterms, or standard sums. The eight maxterms for three variables, together with their symbolic designations

SOP and POS

- The last conversion follows from the definition of minterms and maxterms as shown in the table. From the table, it is clear that the following relation holds: mj' = Mj

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \bar{x}_1\bar{x}_2x_3$	$M_1 = x_1 + x_2 + \bar{x}_3$
2	0	1	0	$m_2 = \bar{x}_1x_2\bar{x}_3$	$M_2 = x_1 + \bar{x}_2 + x_3$
3	0	1	1	$m_3 = \bar{x}_1x_2x_3$	$M_3 = x_1 + \bar{x}_2 + \bar{x}_3$
4	1	0	0	$m_4 = x_1\bar{x}_2\bar{x}_3$	$M_4 = \bar{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\bar{x}_2x_3$	$M_5 = \bar{x}_1 + x_2 + \bar{x}_3$
6	1	1	0	$m_6 = x_1x_2\bar{x}_3$	$M_6 = \bar{x}_1 + \bar{x}_2 + x_3$
7	1	1	1	$m_7 = x_1x_2x_3$	$M_7 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

w	x	y	z	F
0	0	0	0	m0
0	0	0	1	m1
0	0	1	0	m2
0	0	1	1	m3
0	1	0	0	m4
0	1	0	1	m5
0	1	1	0	m6
0	1	1	1	m7
1	0	0	0	m8
1	0	0	1	m9
1	0	1	0	m10
1	0	1	1	m11
1	1	0	0	m12
1	1	0	1	m13
1	1	1	0	m14
1	1	1	1	m15

Truth Table 4-variable function

wx \ yz	00	01	11	10
00	m0 w'x'y'z' 0	m1 w'x'y'z 1	m3 w'x'yz 3	m4 w'x'yz' 2
01	m4 w'xy'z' 4	m5 w'xy'z 5	m7 w'xyz 7	m6 w'xyz' 6
11	m12 wxy'z' 12	m13 wxy'z 13	m15 wxyz 15	m14 wxyz' 14
10	m8 wx'y'z' 8	m9 wx'y'z 9	m11 wx'yz 11	m10 wx'yz' 10

K-Map 4-Variable

ELECTRONICS CLUB

Centre For Innovation

Don’t Care

- The logical sum of the minterms associated with a Boolean function specifies the conditions under which the function is equal to 1. The function is equal to 0 for the rest of the minterms.
- Functions that have unspecified outputs for some input combinations are called incompletely specified functions . In most applications, we simply don’t care what value is assumed by the function for the unspecified minterms. For this reason, it is customary to call the unspecified minterms of a function don’t-care conditions
- They are denoted with an X in the output for that particular input.

ELECTRONICS CLUB

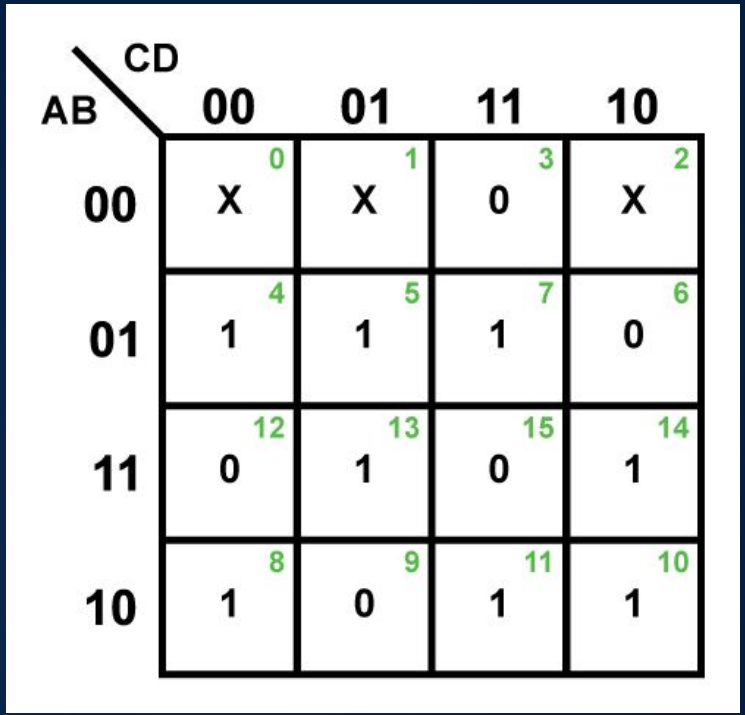
Centre For Innovation

ELECTRONICS CLUB

Centre For Innovation

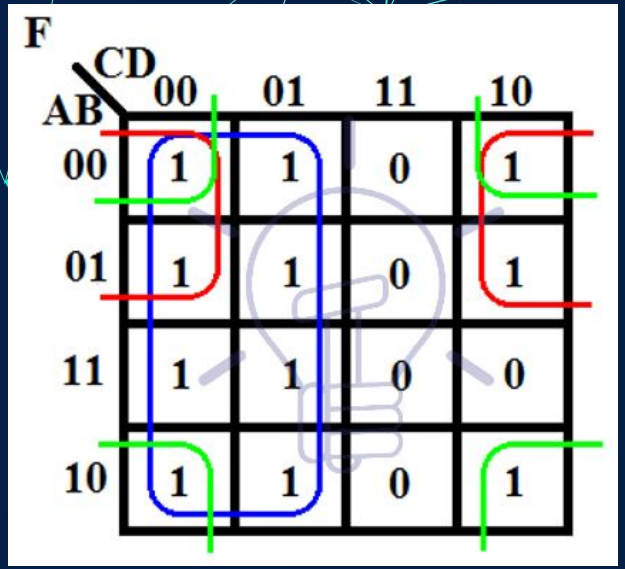
K-MAPS

A Karnaugh map (K-map) is a pictorial method used to minimize Boolean expressions without having to use Boolean algebra theorems and equation manipulations. A K-map can be thought of as a special version of a truth table .



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

K-Maps



Types of grouping:

- Pairs
- Quads: A group of 4 one's that are horizontally or vertically adjacent. End to end or in form of a square. A quad eliminates two variables and their complements.
- Octets: An octet eliminates three variables and their complements.

ELECTRONICS CLUB

K-Maps

Centre For Innovation

Process:

1. Draw the Karnaugh map
2. Look for octets and encircle them.
3. Look for quads and encircle them.
4. Look for pairs and encircle them.
5. Simplify and write down the equation.

ELECTRONICS CLUB

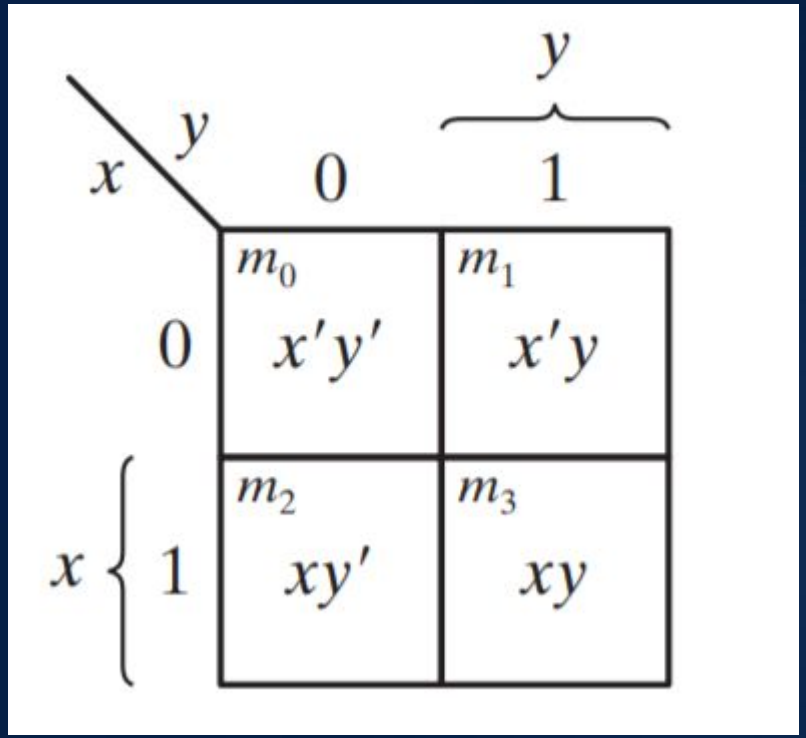
K-Maps

Centre For Innovation

A few key terms:

- Redundant: A groups of 1s or 0s whose all members are overlapped by other groups is called redundant group. We don't consider this group while writing the simplified equations from the K-map.
- Map-rolling: You can consider 2 edges of the map to be adjacent.
- Gray code

2 Variable K-Map



3 variable K-Maps

x \ yz	00	01	11	10
0	m_0 $x'y'z'$	m_1 $x'y'z$	m_3 $x'yz$	m_2 $x'yz'$
1	m_4 $xy'z'$	m_5 $xy'z$	m_7 xyz	m_6 xyz'

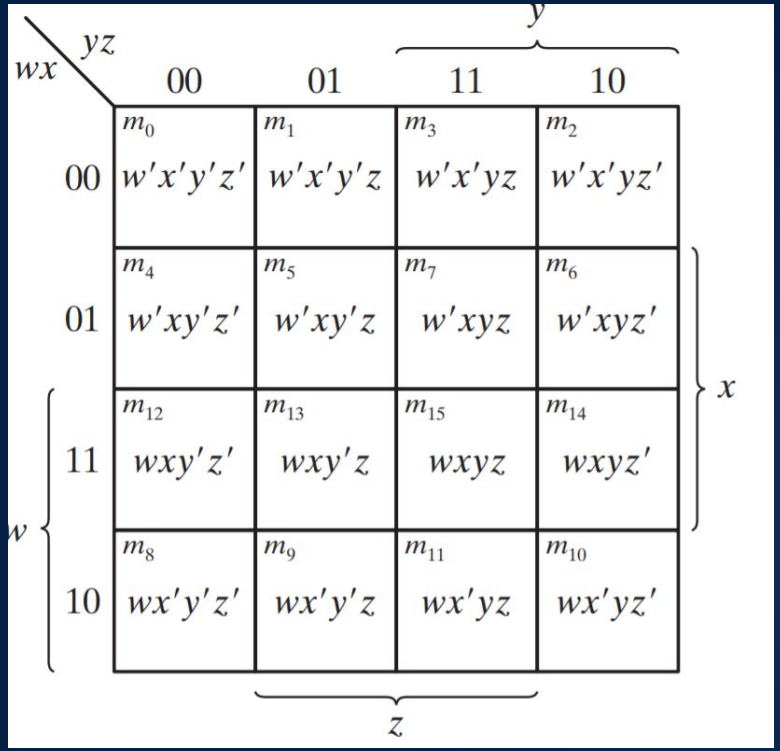
y (11, 10)

z (01, 11)

ELECTRONICS CLUB

4 variable K-Maps

Centre For Innovation



ELECTRONICS CLUB

Centre For Innovation

ELECTRONICS CLUB

Centre For Innovation

02

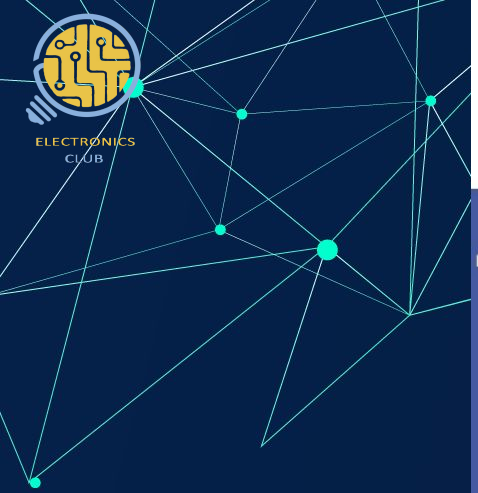
VERILOG

ELECTRONICS CLUB

WHAT IS VERILOG?

Centre For Innovation

- **Verilog**(Verification + Logic) is a *hardware description language* (HDL). A HDL is a computer-based language that describes the hardware of digital logic circuits in a textual form.
- Verilog is intended to be used for verification through simulation, for timing analysis, for test analysis (testability analysis and fault grading) and for logic synthesis.
- HDL describes a relationship between signals that are the inputs to a circuit and the signals that are the outputs of the circuit.
- Verilog supports a design at many levels of abstraction. The major three are –
    - Behavioral level
    - Register-transfer level
    - Gate level



HDL VERSUS SOFTWARE LANGUAGE

HDL	SOFTWARE LANGUAGE
A specialized computer language used to describe the structure and behavior of electronic circuits, most commonly, digital logic circuits	A computer language used to write a set of instructions to allow the CPU to perform a specific task
More complex	Not as complex
Verilog and VHDL are common examples	Java, C, C++, Python, PHP, etc. are common examples
Describe the behavior of digital circuits	Helps to develop various applications

Visit www.PEDIAA.com

Centre For Innovation

WHERE IS VERILOG USED?

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks.

Advantages of FPGAs:

- Can execute many things parallely
- Re-programmable
- Cheaper

Applications:

Aerospace & Defence, ASIC Prototyping, Automobile Industries, High Performance Computing and Data Storage and many more...

ELECTRONICS CLUB

Centre For Innovation

ELECTRONICS CLUB

Centre For Innovation

EDA PLAYGROUND

EDA Playground gives engineers immediate hands-on exposure to simulating and synthesizing SystemVerilog, Verilog, VHDL, C++/SystemC, and other HDLs. All you need is a web browser!

EDA Playground is maintained by Doulos.

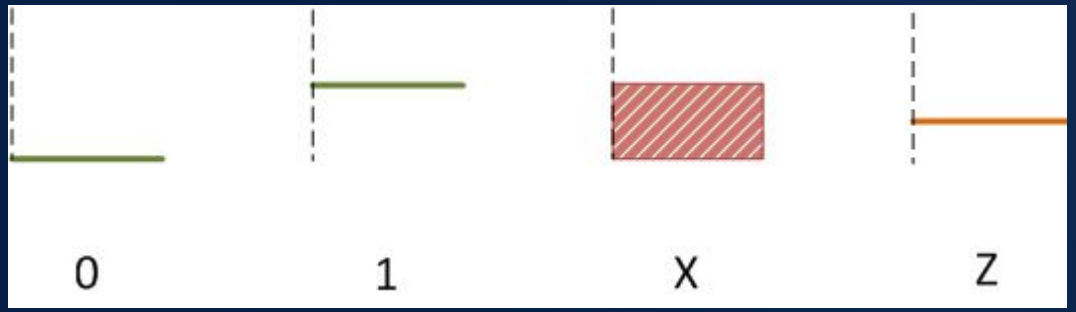
You can refer to the documentation here

Simulator link : https://www.edaplayground.com/

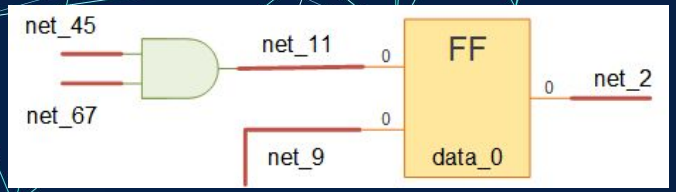
DATA TYPES IN VERILOG

Similar to any other programming language, Verilog also contains a variety of data types. However, unlike programming languages, in Verilog, data can exist only in 4 forms:

1. 0 (Logical LOW)
2. 1 (Logical HIGH)
3. X (Undefined logical value)
4. Z (High impedance tri-state gate or floating wire)



There are 2 main groups of data types : *nets and variables*.



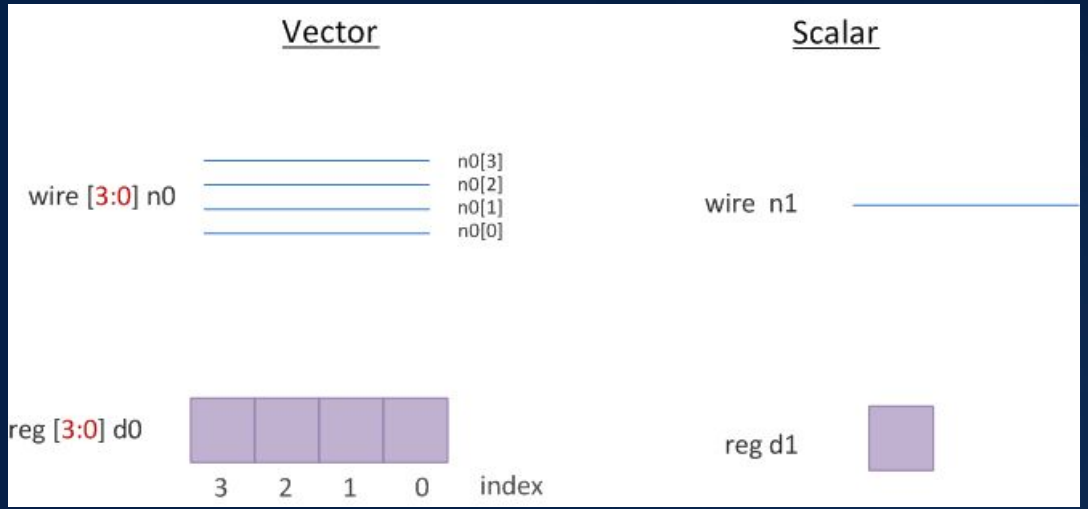
- **Nets** are used to connect between hardware entities like logic gates and hence do not store any value on it's own. There are different types of nets each with different characteristics but most common one is *Wire*. Some common nets are: Wand (AND of drivers), Wor (Or of drivers), Tri (three-state), Supply0 and Supply1 (Logical LOW and HIGH voltage terminals).

- Variables on other hand is an abstraction of a data storage element and can hold values. The data type reg(register) can be used to store values.

- There are also other data types like byte (signed 8 bits), shortint (signed 16 bits), time (unsigned 64 bit), shortreal (similar to float in C), strings, etc.

VERILOG SCALAR AND VECTOR

A *wire* or *reg* declaration without a range specification is called a **scalar** and is 1-bit wide.
If a range is specified then it is called a vector.



The range gives ability to address individual bits in a vector. The left number is the MSB and the right number is the LSB

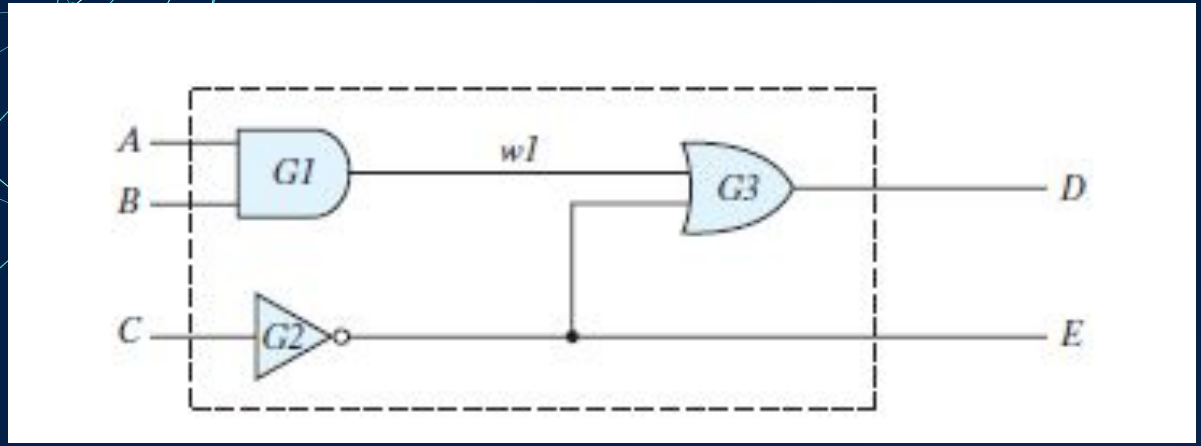
ELECTRONICS CLUB

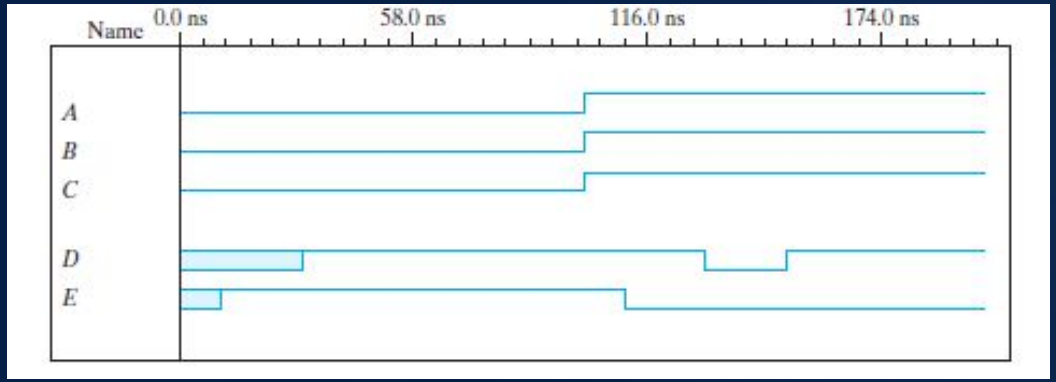
Centre For Innovation

TERMINOLOGY OF VERILOG

1. **Module :** A module is a block of Verilog code that implements a certain functionality. Modules can be embedded within other modules and a higher level module can communicate with lower level modules using their inputs and outputs ports.
   Syntax:
   module <module_name>(*list of ports*)
   //Contents of module
   endmodule

2. **Ports** : Ports allow communication between a module and its environment.
   3 types of ports:
   Input : The design module can only receive values from outside using its input ports
   Output : The design module can only send values to the outside using its output ports
   Inout : The design module can either send or receive values using its inout ports





ELECTRONICS CLUB

Centre For Innovation

3. Boolean Expressions and assign :

- Boolean equations describing combinational logic are specified in Verilog with a continuous assignment statement consisting of the keyword *assign* followed by a Boolean expression. To distinguish arithmetic operators from logical operators, Verilog uses the symbols (&), (/), and (&) for AND, OR, and NOT (complement), respectively.
- For example, **assign** D = (A && B) || (!C);
- The value of D during simulation are determined dynamically by the values of A, B, and C .

3. User-Defined Primitives :

- The logic gates used in Verilog descriptions with keywords *and, or, etc.*, are defined by the system and are referred to as *system primitives*. The user can create additional primitives by defining them in tabular form. These types of circuits are referred to as *user-defined primitives* (UDPs). One way of specifying a digital circuit in tabular form is by means of a truth table.
- They do not use the keywords module and endmodule but instead use the keyword pairs **primitive ... endprimitive**.
- Syntax:

```
primitive <name>(port_list)
//declaring output and inputs
table
//truth table here
endtable
endprimitive
```

```
// Verilog model: User-defined Primitive
primitive UDP_02467 (D, A, B, C);
 output D;
 input  A, B, C;
//Truth table for D 5 f (A, B, C) 5 Σ(0, 2, 4, 6, 7);
 table
//     A     B     C     :     D     // Column header comment
       0     0     0     :     1;
       0     0     1     :     0;
       0     1     0     :     1;
       0     1     1     :     0;
       1     0     0     :     1;
       1     0     1     :     0;
       1     1     0     :     1;
       1     1     1     :     1;
 endtable
endprimitive

// Instantiate primitive

// Verilog model: Circuit instantiation of Circuit_UDP_02467

module Circuit_with_UDP_02467 (e, f, a, b, c, d);
 output       e, f;
 input        a, b, c, d

 UDP_02467          (e, a, b, c);
 and                (f, e, d);          // Option gate instance name omitted
endmodule
```

ELECTRONICS CLUB

Centre For Innovation

4. Test bench :

- A test bench is a program used for exercising and verifying the correctness of a design.
- It has three main purposes.
    1. To generate stimulus for simulation(waveforms)
    2. To apply this stimulus to the module under test and collect output responses
    3. To compare output responses with expected values

Procedural Statements are executable **statements**. The **statements** are executed in sequence except as indicated. Two types of procedural statements are available in Verilog

5. **Initial Statement:**

An initial statement executes only once. It begins its execution at start of simulation which is at time 0.

Syntax for initial statement is:

```
initial        #5 a = 2
```

6. **Always Statement:**

In contrast to initial statement, an always statement executes periodically. Just like an initial statement, an always statement also begins execution at time 0.

Syntax for always statement is:

```
always @(*) out2 = a & b | c ^ d;
```

ELECTRONICS CLUB

Centre For Innovation

Centre For Innovation

THANKS

Do you have any questions?

https://forms.gle/X6ry7PcFWmkF9RGA8

SANJIV B N - EP20B034
ANEESH KANDI - EE20B009

ELECTRONICS CLUB

RESOURCES

Centre For Innovation

- https://www.chipverify.com/verilog/verilog-tutorial
- http://www.asic-world.com/verilog/index.html
- You can read the book - Digital Design - written by Morris Mano