



# SNS COLLEGE OF TECHNOLOGY

Coimbatore-35  
An Autonomous Institution



Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+ +' Grade  
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

## DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

### 19ECB302–VLSI DESIGN

III YEAR/ V SEMESTER

### UNIT 5-SPECIFICATION USING VERILOG HDL

### TOPIC 1 –BASIC CONCEPTS & IDENTIFIERS

10/30/2023



# OUTLINE



- Introduction-History
- Programming Language VS. Verilog HDL
- Different Levels of Abstraction
- Top Down ASIC Design Flow
- Verilog-HDL Simulators
- Overview of Verilog Module
- Activity
- Identifiers of Verilog
- Escaped Identifiers-Case Sensitivity
- Verilog-HDL Structural Language
- Nets and Registers
- Logic Level & switch level Modelling
- Example Programs
- Summary



# INTRODUCTION



## ■ Brief history of Verilog HDL

- 1985: Verilog language and related simulator Verilog-XL were developed by Gateway Automation.
- 1989: Cadence Design System purchased Gateway Automation.
- 1990: Open Verilog International formed.
- 1995: IEEE standard 1364 adopted.

## ■ Features of Verilog HDL

- Ability to mix different levels of abstract freely.
- One language for all aspects of design, testing, and verification.



# VERILOG HDL



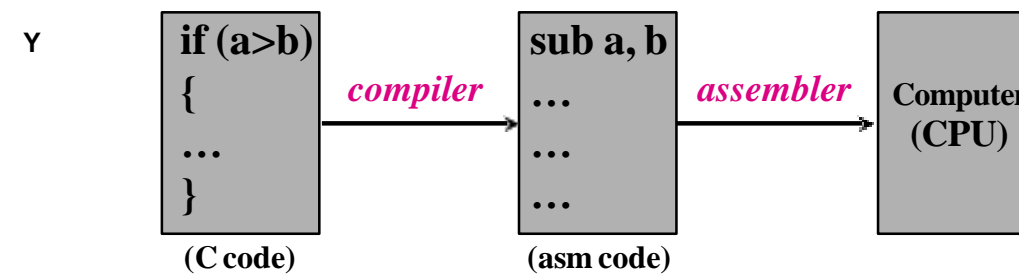
- **Verifying the logic HDL** – Hardware Description Language
  - A programming language that can describe the functionality and timing of the hardware.
- Why use an HDL?
  - It is becoming very difficult to design directly on hardware.
  - It is easier and cheaper to different design options.
  - Reduce time and cost.



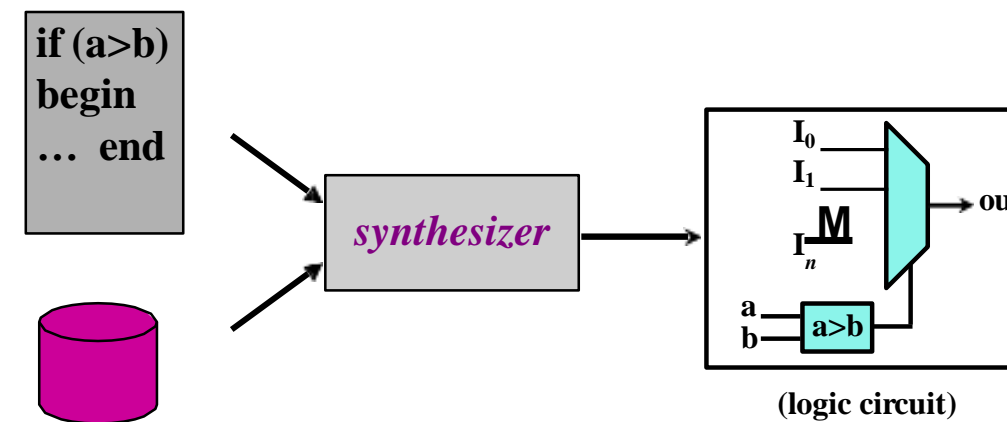
# PROGRAMMING LANGUAGE VS. VERILOG HDL



## ➤ Programming Language



## ➤ Verilog HDL





## DIFFERENT LEVELS OF ABSTRACTION



- Verilog-XL is an **event-driven** simulator that can emulate the hardware described by Verilog HDL.
- Verilog-HDL allows you to describe the design at various levels of abstractions within a design.
  - **Behavioral Level**
  - **RTL Level**
  - **Gate Level**
  - **Switch Level**

10/30/2023





## DIFFERENT LEVELS OF ABSTRACTION



### ➤ **Architecture / Algorithmic (Behavior)**

- A model that implements a design algorithm in high-level language construct.
- A behavioral representation describes how a particular design should respond to a given set of inputs.

### ➤ **Register Transfer Logic (RTL)**

- A model that describes the flow of data between registers and how a design process these data.

### ➤ **Gate Level (Structure)**

- A model that describes the logic gates and the interconnections between them.

### ➤ **Switch Level**

- A model that describes the transistors and the interconnections between them.



# THREE LEVELS OF VERILOG-HDL

## ▪ Gate Level

```

xor u0(.z(hs), .a1(A),
.a2(B));
xor u1(.z(Sum),
.a1(Ci), .a2(hs));
and u2(.z(hc0),
.a1(A), .a2(B));
and u3(.z(hc1),
.a1(Ci), .a2(hs));
or u4(.z(Co),
.a1(hc0), .a2(hc1));

```

BASIC CONCEPTS & IDENTIFIERS /16EC303-  
DESIGN/Dr.B.Sivasankari/PROFESSOR/ECE/SNSCT

## ▪ Switch Level

```

// AND gate of u2
pmos p0(VDD, nand, A), p1(VDD, nand, B);
nmos n0(nand, wire1, A), n1(wire1, GND, B);
      pmos p2(VDD, hc0, nand); nmos n2(hc0, GND, nand);
                                     M

```

## ▪ Behavioral Level (RTL)

```

assign {Co, Sum} = A + B + Ci

```





## VERILOG-HDL SIMULATORS



### ■ VCS (Synopsys)

#### ➤ Platform

Windows NT/XP, SUN Solaris (UNIX), Linux.

### ■ Modelsim (Mentor)

#### ➤ Platform

Windows NT/XP, SUN Solaris (UNIX), Linux.

### ■ NC-Verilog (Cadence)

#### ➤ Platform

Windows NT/XP, SUN Solaris (UNIX), Linux.

### ■ Verilog-XL (Cadence)

#### ➤ Platform

SUN Solaris (UNIX)

### ■ Other Simulators

#### ➤ MAX+PLUS II,

Quartus II (Altera)

#### ➤ Active HDL (Aldec), Silos (Silvaco),



## OVERVIEW OF VERILOG MODULE



- A Verilog module includes the following parts:

**module** module\_name (port\_name);

port declaration   data type declaration

Task & function declaration   module functionality or declaration

timing specification

**endmodule**



# ACTIVITY



## Group Discussion



## IDENTIFIERS OF VERILOG

- Identifiers are user-provided name for Verilog objects within a description.
- Legal characters in identifiers: **a-z, A-Z, 0-9, \_, \$**
- The first character of an identifier must be an alphabetical character (**a-z, A-Z**) or an underscore (**\_**). **A-Z** or an underscore (**\_**).
- Identifiers can be up to 1024 characters long.

### Example:

Mux\_2\_1 abc123 ABC123

Sel\_ A\$b\$10



## ESCAPED IDENTIFIERS

- Escaped Identifiers start with a backslash (\) and end with a white space.
- They can contain any printable ASCII characters.
- Backslash and identifiers.
- white space are not part of the identifiers.
- Example:

```
module \2:1mux(out, a, b, sel);
```

```
not u0(\~out, in);
```



## CASE SENSITIVITY

- Verilog is a case-sensitive language.
- You can run Verilog in case-insensitive mode by specifying **-u** command line option.

Example:

```
module inv(out, in);  
... end module
```

```
module Inv(out, in);  
... end module
```

// Both *inv* and *Inv* are viewed as two different modules.





# VERILOG-HDL STRUCTURAL LANGUAGE



## • Verilog Module

- Modules are basic building blocks in hierarchy.
- Every module description starts with module name(output\_ports, input\_ports), and ends with endmodule.

## • Module Ports

- Module ports are equivalent to the pins in hardware.
- Declare ports to be input, output, or inout (bidirectional) in the module description.



# SYNTAX OF VERILOG

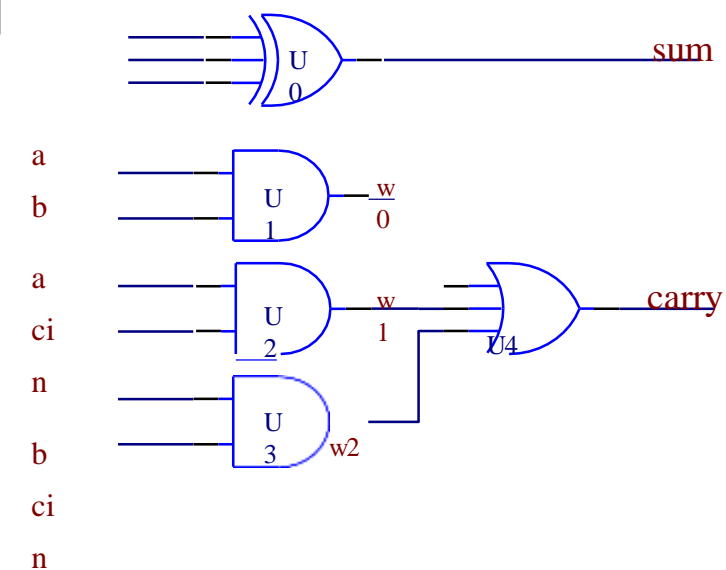
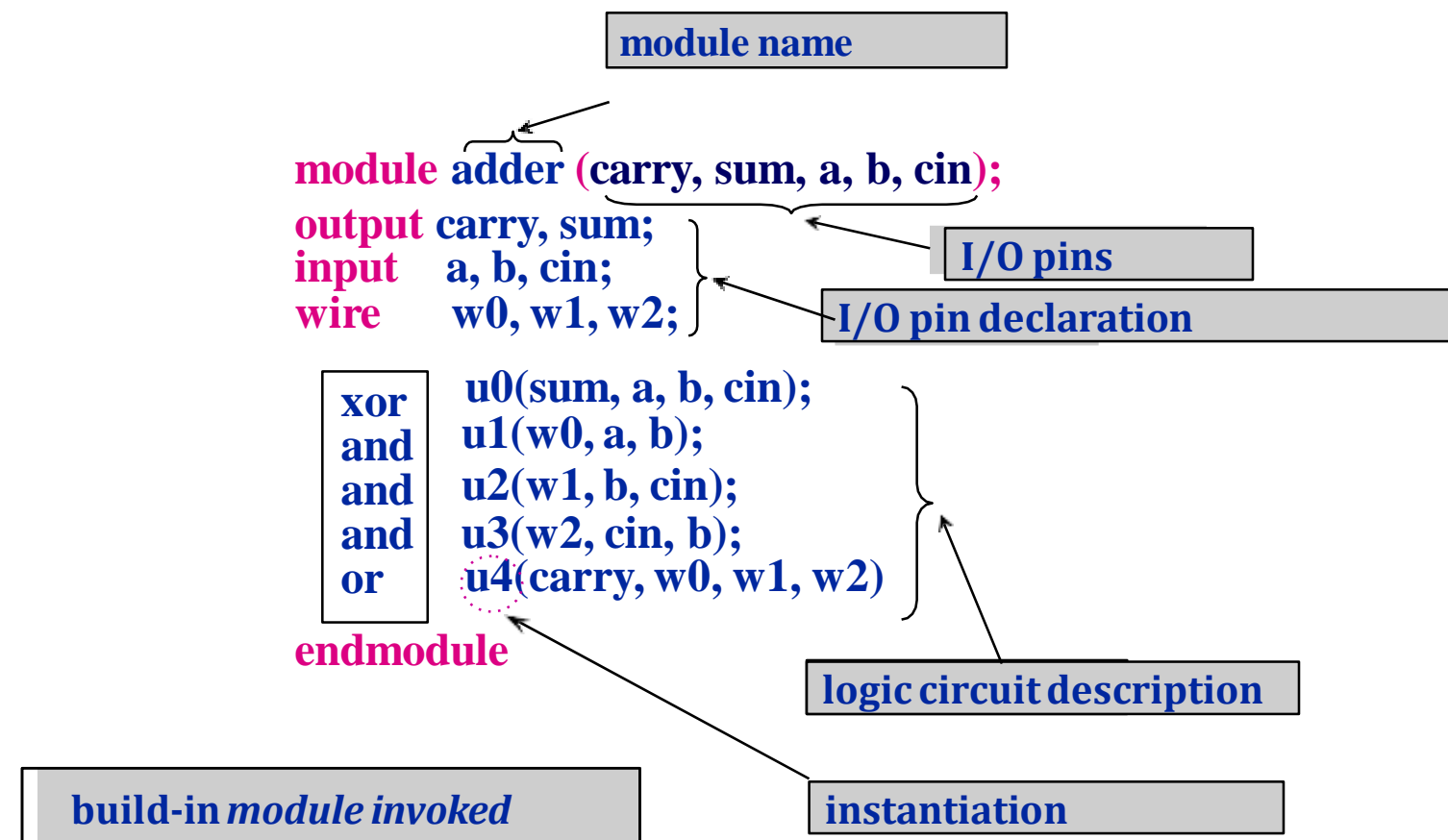


- C-like structural language
  - statement: **begin ... end, if ... else...**, and so on.
  - free writing format: statement ended with **;**.
  - remark: between **/\*** **\*/** or **//** until the line feed.
  - hierarchical modules.



# EXAMPLE OF ADDER

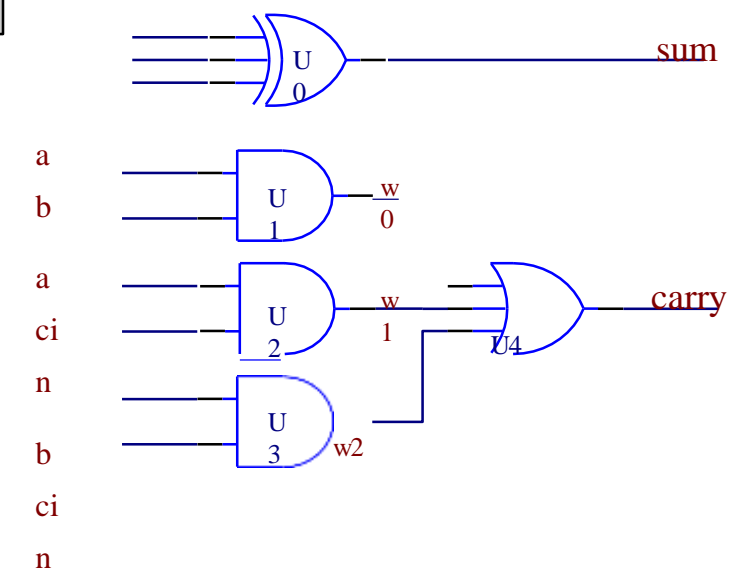
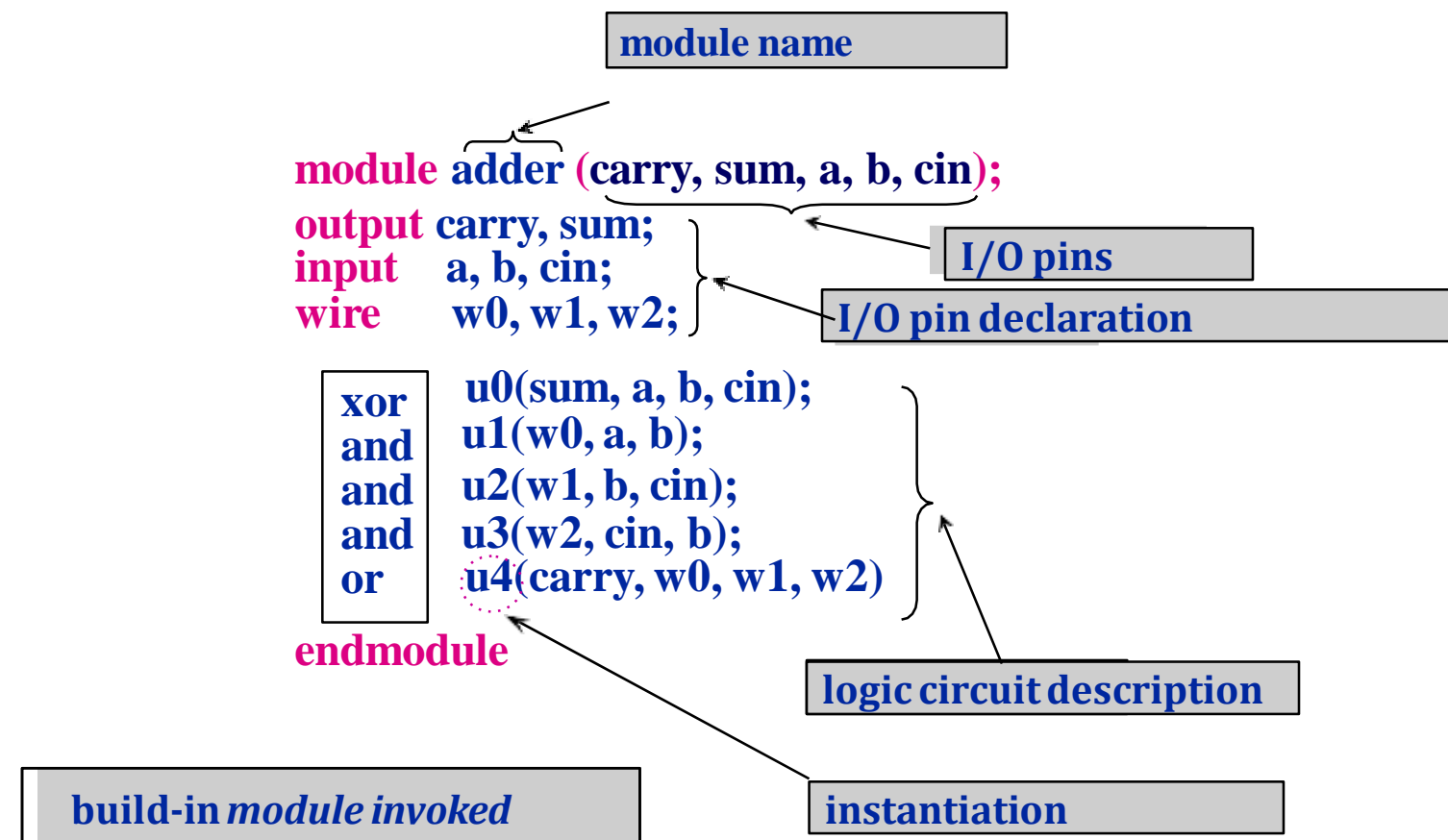
## A Full Adder





# EXAMPLE OF ADDER

## A Full Adder





# EXAMPLE OF ADDER

## A Full Adder

```
module adder (carry, sum, a, b, cin); output carry,
sum;
input  a, b, cin; reg          sum, carry;

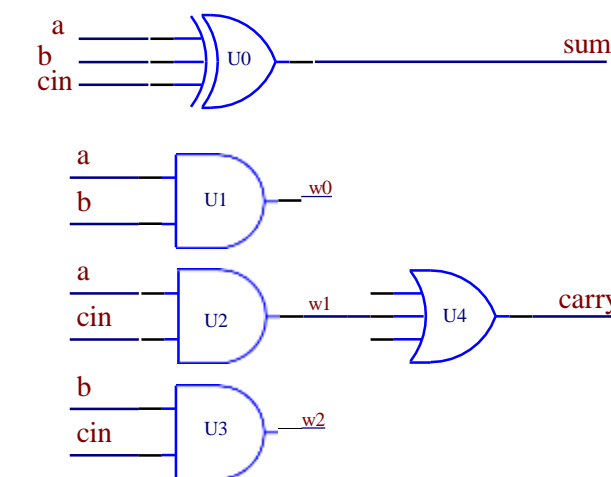
always @(a or b or cin)
{carry, sum} = a + b + cin; endmodule //behavioral
level
```

```
module adder (carry, sum, a, b, cin); output carry,
sum;
input  a, b, cin; wire          w0, w1, w2;

xor    u0(sum, a, b, cin);
and    u1(w0, a, b);
and    u2(w1, b, cin);
and    u3(w2, cin, b);
or     u4(carry, w0, w1, w2)
endmodule //gate level
```

```
module adder (carry, sum, a, b,
cin); output carry, sum;
input  a, b, cin;

assign {carry, sum} = a + b + cin;
endmodule //RTL level
```





## ASSESSMENT



- 1.Verilog HDL-Abbreviation
- 2.List out any four VERILOG HDL simulators
- 3.List out the design abstraction level
- 4.List out the types of modeling styles
- 5.Write the syntax of verilog HDL module.
- 6.what are the needs of identifiers?
- 7.Write Verilog HDL Code for full adder using 3 modeling style.





## SUMMARY & THANK YOU