**PRACTICALS**

**List of Experiments**

1. Study of simulation tools and synthesis tools
2. Simulation of basic logic gates using Xilinx Software and FPGA.
3. Design, simulate and synthesis of adders using Xilinx Software and FPGA.
4. Design, simulate and synthesis of Multiplexers & demultiplexers Xilinx Software and FPGA.
5. Design, simulate and synthesis of Encoders & Decoders using Xilinx Software and FPGA
6. Design, simulation and synthesis of flip flops using Xilinx Software and FPGA.
7. Design, simulation and synthesis of Counters using Xilinx Software and FPGA

**COURSE OUTCOMES:** At the end of the course, the students will be able to

**CO1:** apply the basic concepts and different levels of abstraction in Verilog HDL.

**CO2:** relate the characteristics of MOS transistors.

**CO3:** design and Illustrate CMOS based digital circuit designs, data path and arithmetic circuits for processor design.

**CO4:** write program using Verilog HDL for VLSI digital circuits

# Choice of Technology



- Two distinct types of technology are fabricated in silicon based upon
  - BJT (Bipolar Junction Transistor)
  - MOS (Metallic Oxide Semiconductor)
- Since processing of these technologies is very different, it is impractical to mix them up within a chip.
- MOS logic occupies much smaller area of silicon than the equivalent BJT logic.
- MOS technology has a much higher potential packing density.
- A MOS logic circuit requires appreciably less current and hence less power than its bipolar counter part.
- However, bipolar circuits operate faster than MOS circuits

# Most Commonly used HDLs



- **Verilog**
  - Verilog HDL is commonly used in the US industry. Major digital design companies in Pakistan use Verilog HDL as their primary choice.
  - most commonly used in the design, verification, and implementation of digital logic chips
- **VHDL** (VHSIC (Very High Speed Integrated Circuits) hardware description language)
  - VHDL is more popular in Europe.
  - commonly used as a design-entry language for *field-programmable gate arrays*. Field-Programmable Gate Array is a type of logic chip that can be programmed.

# Design Styles

Verilog, like any other hardware description language, permits a design in either Bottom-up or Top-down methodology.



## Bottom-Up Design

The traditional method of electronic design is bottom-up.

Each design is performed at the gate-level using the standard gates (refer to the Digital Section for more details).

With the increasing complexity of new designs this approach is nearly impossible to maintain.

New systems consist of ASIC or microprocessors with a complexity of thousands of transistors.

These traditional bottom-up designs have to give way to new structural, hierarchical design methods.

Without these new practices it would be impossible to handle the new complexity.

# Verilog Code

## Structural Method



```
module half_adder_structural (
  input a,    // Input 'a'
  input b,    // Input 'b'
  output s,   // Output 's' (Sum)
  output c    // Output 'c' (Carry)
);
xor gate_xor (s, a, b);  // XOR gate for sum
 and gate_and (c, a, b);  // AND gate for carry
endmodule
```

# Scalars and Vectors

**Scalars:** Single bit wire or register is known as a scalar.

```
wire a; reg a;
```

**Vectors:** The nets or registers can be declared as vectors to represent multiple bit widths. If bit width is not specified, it is a scalar.

wire [5:0] a; reg [5:0] a;

# Constants

The value of constants can not be changed. It is read-only in nature.

# Integer data type

The integers are general-purpose 32-bit register data types. They are declared by the ‘integer’ keyword.

```
integer count;
```

## Unsized

Numbers without a *base_format* specification are decimal numbers by **default**. Numbers without a *size* specification have a default number of bits depending on the type of simulator and machine.

```
1 | integer a = 5423;        // base format is not specified, a gets a decimal va
2 | integer a = 'h1AD7;      // size is not specified, because a is int (32 bits)
```

## Strings



A sequence of characters enclosed in a double quote `" "` is called a string. It cannot be split into multiple lines and every character in the string take 1-byte to be stored.

```
"Hello World!"          // String with 12 characters -> require 12 bytes
"x + z"                  // String with 5 characters

"How are you
feeling today ?"        // Illegal for a string to be split into multiple line
```

## String

An ordered collection of characters is called a string. They can be stored in reg data type. Each character in a string requires 1 byte (8 bits) for storage and is typically mentioned within double-quotes (" ").

```
reg [8*11:0] name = "Hello World"; // String "Hello World" //requires 11 bytes space.
```

## Identifiers

*Identifiers* are names of variables so that they can be referenced later on. They are made up of alphanumeric characters `[a-z][A-Z][0-9]`, underscores `_` or dollar sign `$` and are case sensitive. They cannot start with a digit or a dollar sign.

```
integer var_a;          // Identifier contains alphabets and underscore -> Vali
integer $var_a;         // Identifier starts with $ -> Invalid
integer v$ar_a;         // Identifier contains alphabets and $ -> Valid
integer 2var;           // Identifier starts with a digit -> Invalid
integer var23_g;        // Identifier contains alphanumeric characters and unde
integer 23;             // Identifier contains only numbers -> Invalid
```

# Escaped Identifiers



Verilog HDL allows any character to be used in an identifier by escaping the identifier.

Escaped identifiers provide a means of including any of the printable ASCII characters in an identifier (the decimal values 33 through 126, or 21 through 7E in hexadecimal).

- Escaped identifiers begin with the back slash ( \ )
- Entire identifier is escaped by the back slash.
- Escaped identifier is terminated by white space (Characters such as commas, parentheses, and semicolons become part of the escaped identifier unless preceded by a white space)
- Terminate escaped identifiers with white space, otherwise characters that should follow the identifier are considered as part of it.

# THANK YOU