Université Concordia University

Department of Electrical & Computer Engineering

# COEN 212: DIGITAL SYSTEMS DESIGN I

## Lecture 5.2: Hardware Description Language (HDL)

**Instructor:** Dr. Reza Soleymani, Office: EV-5.125,
Telephone: 848-2424 ext.: 4103.

## Lecture 5.2: Objectives of this lecture

- In this lecture, we see:
  - Basic Definitions.
  - Design Steps.
  - Simulation.
  - Timing.

## Lecture 5.2: Reading for this lecture

- Digital Design by M. Morris R. Mano and Michael D. Ciletti, 6th Edition, Pearson, 2018:
  - Section 3.9

# Lecture 5.2:
# Hardware Description Language (HDL)

- HDL software is the tool for designing digital circuits.

- Developing a digital circuit using HDL consists of the following steps:
  - Design entry: creating an HDL based definition of the functionality of the circuit
  - Logic Simulation: verifying the functionality of the logic using a computer
  - Logic synthesis: deriving the list of physical components need to implement the circuit and their interconnection (a net list)
  - Timing Verification: to make sure that the fabricated circuit operates at the specified speed. In this stage, we need to take into consideration the gate delays.

# Lecture 5.2: HDL: Module Definition

Department of Electrical & Computer Engineering

- Primitives: **module**, **input**, **output**, **wire**, **and**, **or**, **not**

// Description of the 3 gate circuit

**module** my_circuit (A, B, C, D, E);

**output** D, E;

**input** A, B, C;

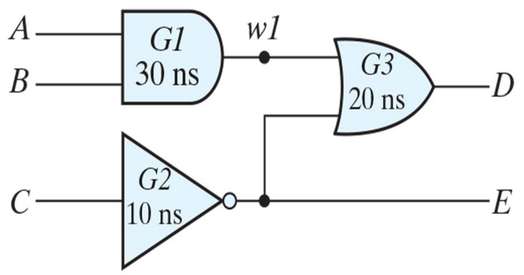
**wire** $w_1$;

**and** $G_1(w_1, A, B)$;

**not** $G_2(E, C)$;

**or** $G_3(D, w_1, E)$;

**end module**



Department of Electrical & Computer Engineering

# Lecture 5.2: HDL: Gate delays

- To simulate our circuit, we need to specify gate delays.
- Assume: propagation delays for $G_1$, $G_2$, and $G_3$ are 30 ns, 10 ns, and 20 ns, respectively.

```
// Verilog model with propagation delay
module my_circuit_prop_delay (A, B, C, D, E);
output D, E;
input A, B, C;
wire w1;
and #(30)G1(w1,A,B);
not #(10)G2(E,C);
or #(20)G3(D,w1,E);
end module
```

# Lecture 5.2: HDL: Simulation

- To simulate our circuit, we need to write a **testbench.**

// Test bench for my-circuit

Module t_my_circuit_prop_delay;

wire D, E;

reg A, B, C;

my_circuit_prop_delay $M_1(A, B, C, D, E)$;

initial

begin

$A = 1'b0;\ B = 1'b0; C = 1'b0;$

$\#100\ A = 1'b1;\ B = 1'b1; C = 1'b1;$

end

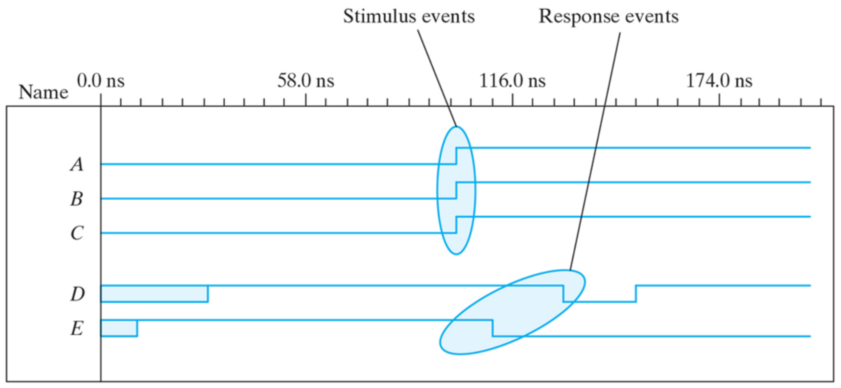
initial #200 $ finish;

end module

# Lecture 5.2: HDL: Simulation

Université Concordia University
Department of Electrical & Computer Engineering

- The result of the simulation will be:



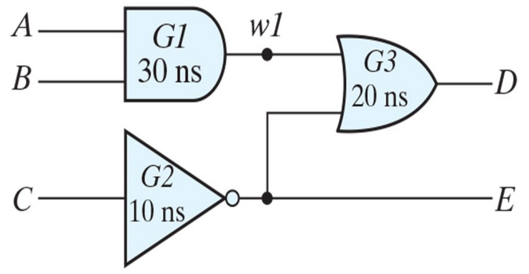
- Note that $D$ is not defined for the first 20 ns, and $E$ for the first 10 ns.

# Lecture 5.2: Boolean expressions in HDL

- Boolean expressions are defined using
  - & for AND
  - | for OR
  - ~ for NOT

using the keyword **assign**.

Example: For



*The expression is:*

$$\underline{\textbf{assign}}\ D = (A \& B) | {\sim} C$$

## Lecture 5.2: Boolean expressions in HDL

Concordia University — Department of Electrical & Computer Engineering

- **Example**: write an HDL program implementing

$$E = A + BC + B'D$$
$$F = B'C + BC'D'$$

```
// Verilog model: My Boolean Circuit
module my_Boolean_Circuit (E, F, A, B, C, D);
output E, F;
input A, B, C, D;
assign E = A|(B&C)|(~B&D);
assign F = (~B&C)|(B&~C&~D);
end module
```

# Lecture 5.2: User-Defined Primitives

Université Concordia University
Department of Electrical & Computer Engineering

- **User-Defined Primitives (UDP):**

- Some basic logic operations, e.g., AND, OR, NOT are included in the language. We can add other primitives (UDP) using a table. These user-defined primitives can be later instantiated when designing more complex circuits.

- **Example:** Introduce a UDP implementing
- $X(A,B,C) = \sum(0,2,4,6,7)$

## Lecture 5.2: User-Defined Primitives

Department of Electrical & Computer Engineering

```
// User Defined Primitive (UDP) example
primitive crctp (X,A,B,C);
output X;
input A,B,C;
// Truth Table for X(A,B,C)=∑(0,2,4,6,7)
Table
//A   B   C   :    X
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;
end tabel
end primitive
```

## Lecture 5.2: User-Defined Primitives

Concordia University
Department of Electrical & Computer Engineering

- The newly introduced primitive can then be instantiated using the following lines of code:
- // Instantiate primitive
- module declare_crctp;
- reg $x, y, z$;
- wire $w$;
- crctp $(w, x, y, z)$;
- end module