**ET-3206 DD LABORATORY**

**JOURNAL**

**EXPERIMENT - 6**

**Simulation of Combinational Circuits in Verilog HDL**

**(Comparator & 8-Bit Adder)**

**Group No.: - *ETB-B3-G9***

**Group Members**

**1. Raghavi Narayanan(12311308)**

**Objectives -**

1. **To implement combinational logic circuits** using Verilog HDL, focusing on fundamental digital design principles where outputs depend solely on current inputs without memory elements.
2. **To design and verify a digital comparator circuit** that compares two multi-bit binary numbers and generates output signals indicating whether the first number is greater than, less than, or equal to the second number.
3. **To design and verify an 8-bit binary adder circuit** capable of performing addition operations on two 8-bit binary numbers, including carry generation and propagation.
4. **To develop comprehensive testbenches** for simulation and verification of the designed circuits, ensuring correct functionality across various input combinations and edge cases.

**EDA Tool Used:**

* Synopsis VCS and VERDI

1. **COMPARTOR**

**Description**

A **digital comparator** is a combinational logic circuit that compares two binary numbers and determines their relative magnitudes. It produces output signals that indicate whether one number is greater than, less than, or equal to the other.

**Working Principle**

The comparator takes two n-bit binary numbers as inputs (typically denoted as A and B) and generates three output signals:

* **A > B (Greater Than)**: Active when the magnitude of A is greater than B
* **A < B (Less Than)**: Active when the magnitude of A is less than B
* **A = B (Equal To)**: Active when both numbers have the same magnitude

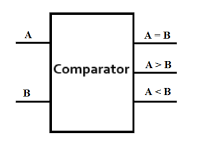
**Types of Comparators**

* **1-bit Comparator**: Compares single-bit inputs
* **Multi-bit Comparator**: Compares n-bit binary numbers (2-bit, 4-bit, 8-bit, etc.)
* **Magnitude Comparator**: Compares unsigned binary numbers (implemented in this experiment)
* **Equality Comparator**: Only checks if two numbers are equal

**RTL Description**

**Module Interface**

**Block Diagram**



**Design Module Code**

// comparator.v

module comparator(

input [3:0] A,

input [3:0] B,

output reg greater,

output reg equal,

output reg smaller

);

always @(\*) begin

if (A > B) begin

greater = 1;

equal = 0;

smaller = 0;

end

else if (A == B) begin

greater = 0;

equal = 1;

smaller = 0;

end

else begin

greater = 0;

equal = 0;

smaller = 1;

end

end

endmodule

**Testbench Code**

// comparator\_tb.v

`timescale 1ns/1ps

`include "comparator.v"

module comparator\_tb;

reg [3:0] A, B;

wire greater, equal, smaller;

// Instantiate the comparator

comparator uut (

.A(A),

.B(B),

.greater(greater),

.equal(equal),

.smaller(smaller)

);

initial begin

// Dump waveform

$fsdbDumpfile("comparator\_tb.fsdb");

$fsdbDumpvars(0, comparator\_tb);

// Test cases

A = 4'b0000; B = 4'b0000; #10;

A = 4'b0101; B = 4'b0011; #10;

A = 4'b0010; B = 4'b1010; #10;

A = 4'b1111; B = 4'b1111; #10;

A = 4'b1001; B = 4'b0111; #10;

$finish;

end

initial begin

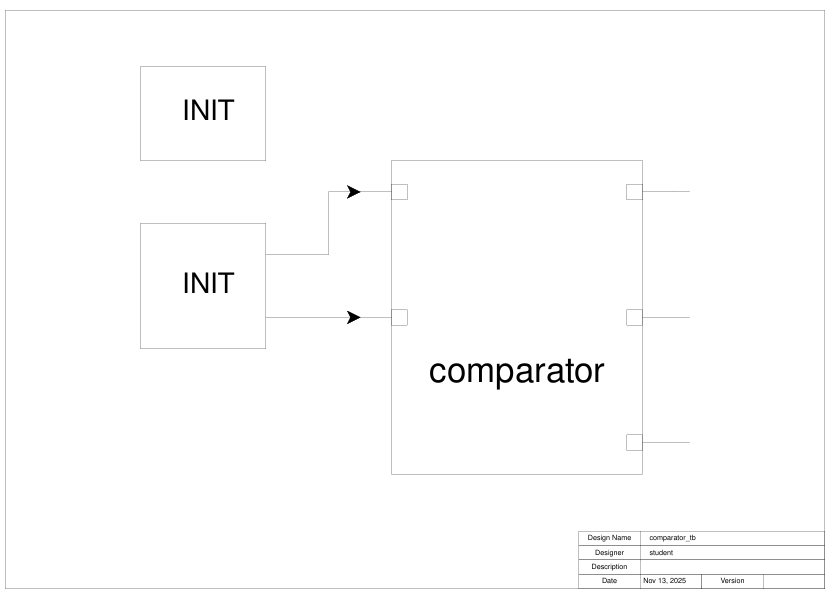
$monitor("Time=%0t | A=%b | B=%b | greater=%b | equal=%b | smaller=%b",

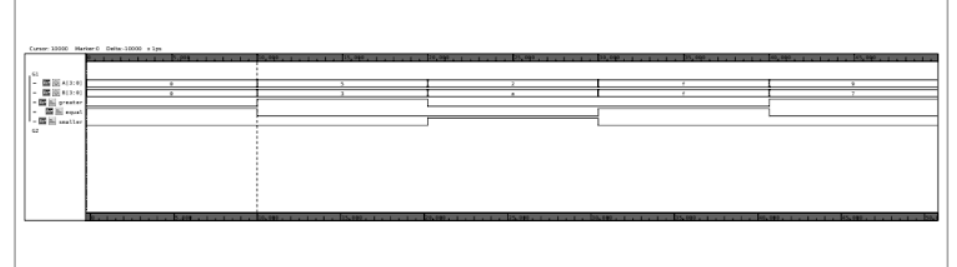
$time, A, B, greater, equal, smaller);

end

endmodule

**Results:   
  
RTL Block Diagram**

****



1. **8-bit Adder**

**Description**

An **8-bit adder** is a combinational arithmetic circuit that performs binary addition of two 8-bit numbers along with an optional carry input, producing an 8-bit sum output and a carry output. It is one of the most fundamental arithmetic circuits in digital systems and forms the backbone of arithmetic logic units (ALUs) in processors.

**Working Principle**

The 8-bit adder takes three inputs and produces two outputs:

**Inputs:**

* **A [7:0]**: First 8-bit binary number (augend)
* **B [7:0]**: Second 8-bit binary number (addend)
* **Cin**: Carry input (optional, typically used for multi-precision arithmetic)

**Outputs:**

* **Sum [7:0]**: 8-bit sum result
* **Cout**: Carry output (indicates overflow for unsigned addition)

The addition follows the binary addition rules:

* 0 + 0 = 0 (Carry = 0)
* 0 + 1 = 1 (Carry = 0)
* 1 + 0 = 1 (Carry = 0)
* 1 + 1 = 0 (Carry = 1)
* 1 + 1 + 1 = 1 (Carry = 1) [when carry-in exists]

**Types of Adder Architectures**

1. **Ripple Carry Adder (RCA)**:
2. **Carry Look-Ahead Adder (CLA)**:
3. **Carry Select Adder**:
4. **Carry Save Adder**:

**Full Adder Logic:**

* **Sum = A ⊕ B ⊕ Cin** (three-input XOR)
* **Cout = (A·B) + (Cin·(A ⊕ B))** (majority function)

Each full adder adds one bit from A, one bit from B, and the carry from the previous stage.

**RTL Description**

**Module Interface**

Module Name: adder\_8bit

Inputs: A [7:0] – First 8-bit binary number

B [7:0] - Second 8-bit binary number

Cin - Carry input (1-bit)

Outputs: Sum [7:0] - 8-bit sum result

Cout - Carry output (1-bit)

**Block Diagram**



**Verilog Code**

// Adder\_8Bit.v

module Adder\_8Bit (

input [7:0] A,

input [7:0] B,

output [7:0] SUM,

output COUT

);

assign {COUT, SUM} = A + B;

endmodule

**Testbench Code**

// Adder\_8Bit\_tb.v

`timescale 1ns/1ps

`include "Adder\_8Bit.v"

module Adder\_8Bit\_tb;

reg [7:0] A, B;

wire [7:0] SUM;

wire COUT;

// Instantiate the adder

Adder\_8Bit uut (

.A(A),

.B(B),

.SUM(SUM),

.COUT(COUT)

);

initial begin

// Dump waveform for Verdi

$fsdbDumpfile("Adder\_8Bit\_tb.fsdb");

$fsdbDumpvars(0, Adder\_8Bit\_tb);

// Test cases

A = 8'b00000000; B = 8'b00000000; #10;

A = 8'b00000001; B = 8'b00000010; #10;

A = 8'b00001111; B = 8'b00000001; #10;

A = 8'b11111111; B = 8'b00000001; #10;

A = 8'b10101010; B = 8'b01010101; #10;

$finish;

end

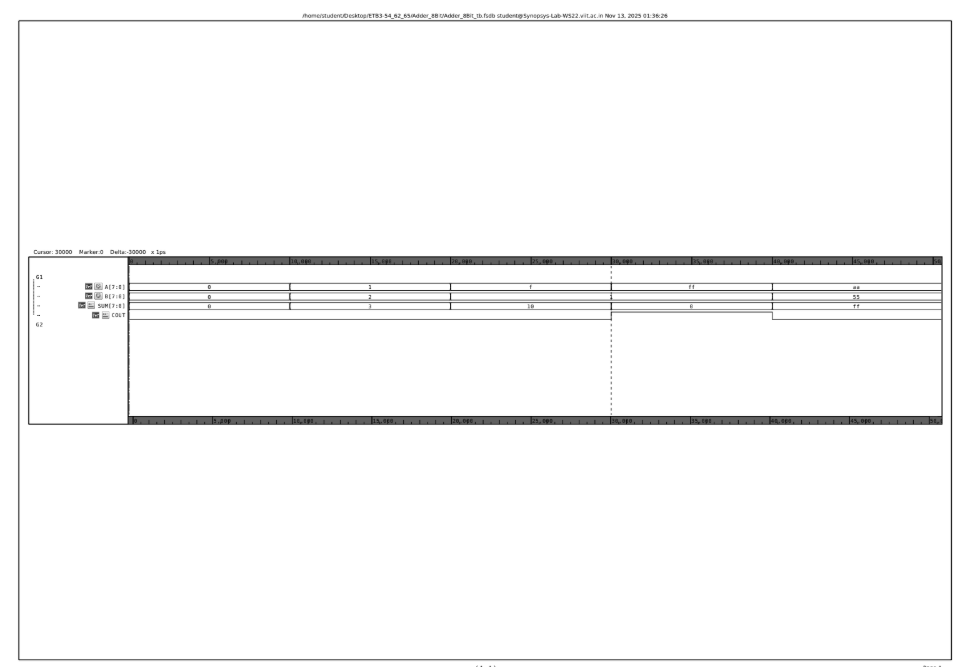
initial begin

$monitor("Time=%0t | A=%b | B=%b | SUM=%b | COUT=%b",

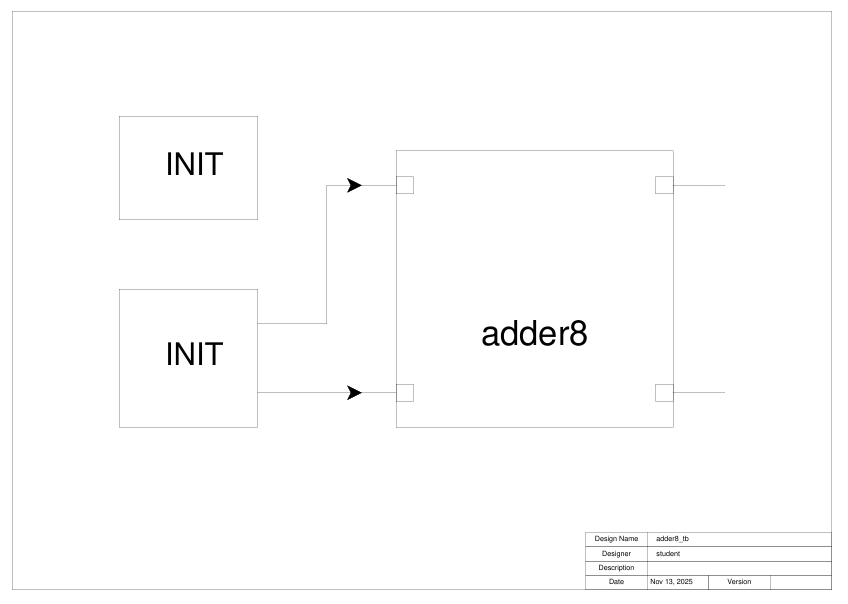
$time, A, B, SUM, COUT);

end

endmodule

**Result: Waveform** 

**RTL Block Diagram:**

****



**Conclusion:**

This experiment successfully demonstrated the design and implementation of two fundamental combinational circuits, a digital comparator and an 8-bit adder, using Verilog HDL. The comparator effectively compared multi-bit binary numbers and generated mutually exclusive outputs (greater than, less than, equal to), while the 8-bit adder accurately performed binary addition with carry propagation. Comprehensive testbench simulations verified functional correctness across all input combinations, edge cases, and boundary conditions, confirming zero-latency operation and proper logical behaviour for both circuits.

Through this experiment, essential skills in digital design were developed, including writing synthesizable Verilog code, creating self-checking testbenches, analysing waveforms, and understanding RTL design principles. Both circuits demonstrated efficient resource utilisation and scalable, parameterizable designs suitable for various applications. The comparator and adder serve as critical building blocks in microprocessors, ALUs, and arithmetic units, forming the foundation for more complex digital systems.

This experiment successfully bridged theoretical digital logic concepts with practical hardware implementation. The experience gained in HDL coding, simulation, verification, and performance analysis provides a solid foundation for advanced topics in computer architecture and VLSI design, preparing for future work in sequential circuits and system-level digital development.