EEL2020 Digital Design Lab Report

Sem II AY 2023-24

Experiment No.	:	**07**
Name	:	**Lavangi Parihar**
Roll No.	:	**B22EE044**
Partner Name (Partner Roll No.)	:	**Chaital V Ghan (B22CS020)**

Objective

(a) Designing a 4-bit shift Register
(b) Multiplier using shifter and adder

4-BIT SHIFT REGISTER

Logic Design

It takes a 4-bit input and extends it to a 6-bit output by appending two zeros at the beginning. This effectively creates a 4-bit shift register where the input is shifted by two positions to the right, and the vacated positions are filled with zeros.

Source Description

- Design source

```
module extend_input(
   input [3:0] in,  // 4-bit input
   output [5:0] out // 6-bit output
);

// Prepend two '0's to the 4-bit input to make it a 6-bit output
assign out = {2'b00, in};

endmodule
```

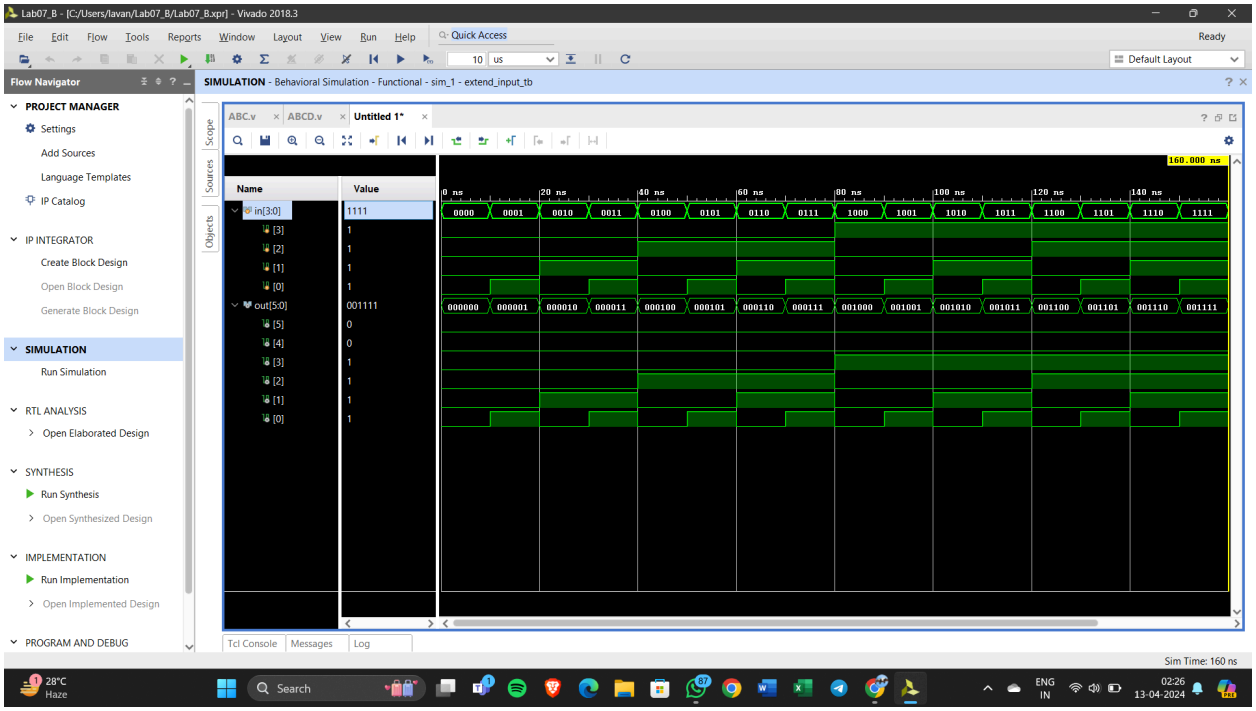
- Simulation source (if any)

```
`timescale 1ns / 1ps

module extend_input_tb;

   // Testbench signals
   reg [3:0] in;
   wire [5:0] out;
```

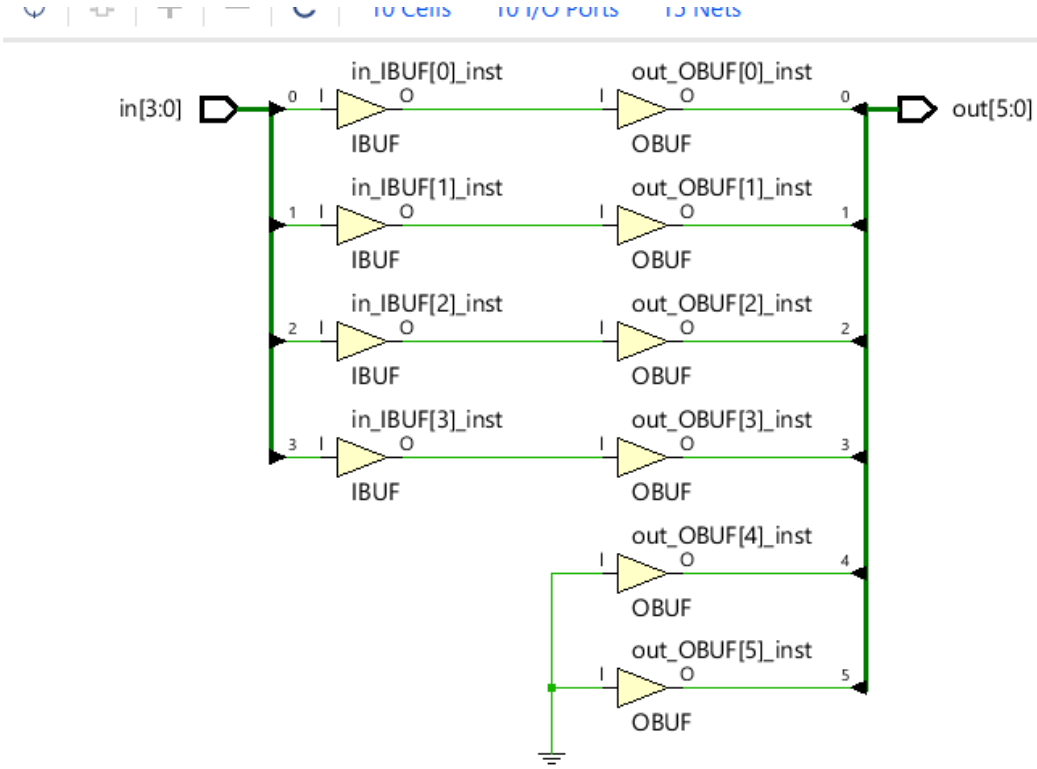
```
    // Instantiate the module under test
    extend_input dut(
        .in(in),
        .out(out)
    );

    initial begin
        // Initialize input
        in = 4'd0; // Start with 0

        // Apply different inputs and wait 10 time units between each
        #10 in = 4'd1;  // 1
        #10 in = 4'd2;  // 2
        #10 in = 4'd3;  // 3
        #10 in = 4'd4;  // 4
        #10 in = 4'd5;  // 5
        #10 in = 4'd6;  // 6
        #10 in = 4'd7;  // 7
        #10 in = 4'd8;  // 8
        #10 in = 4'd9;  // 9
        #10 in = 4'd10; // 10
        #10 in = 4'd11; // 11
        #10 in = 4'd12; // 12
        #10 in = 4'd13; // 13
        #10 in = 4'd14; // 14
        #10 in = 4'd15; // 15

        #10 $finish; // End simulation
    end

    // Monitor changes and print
    initial begin
        $monitor("Time = %t, Input = %4b, Output = %6b", $time, in, out);
    end

Endmodule
```

Video Implementation:
Link

Simulation Results (Timing diagram)



Elaborated Design



MULTIPLIER USING SHIFTER AND ADDER

Logic Design

It takes a 4-bit input and multiplies it by 6 by first multiplying it by 2 (shifting left by 1) and then by 4 (shifting left by 2), and finally adding the two shifted values together.

Source Description

- **Design source**

```
module six_multiplier(
    input [3:0] in, // 4-bit input
    output [6:0] out // 7-bit output
);
    // Intermediate values for shifted inputs
    wire [4:0] shift_one;
    wire [5:0] shift_two;

    // Shift operations
    assign shift_one = in << 1; // Multiply by 2
    assign shift_two = in << 2; // Multiply by 4

    // Sum the two shifted values
    // Since shift_one is 5 bits and shift_two is 6 bits,
    // we need to align them for addition.
    // We will use a 6-bit representation for shift_one_sum to align with shift_two.
    wire [5:0] shift_one_sum = {1'b0, shift_one}; // Prepend a 0 to make it 6 bits
    assign out = shift_one_sum + shift_two; // The sum will be a 7-bit number

Endmodule
```

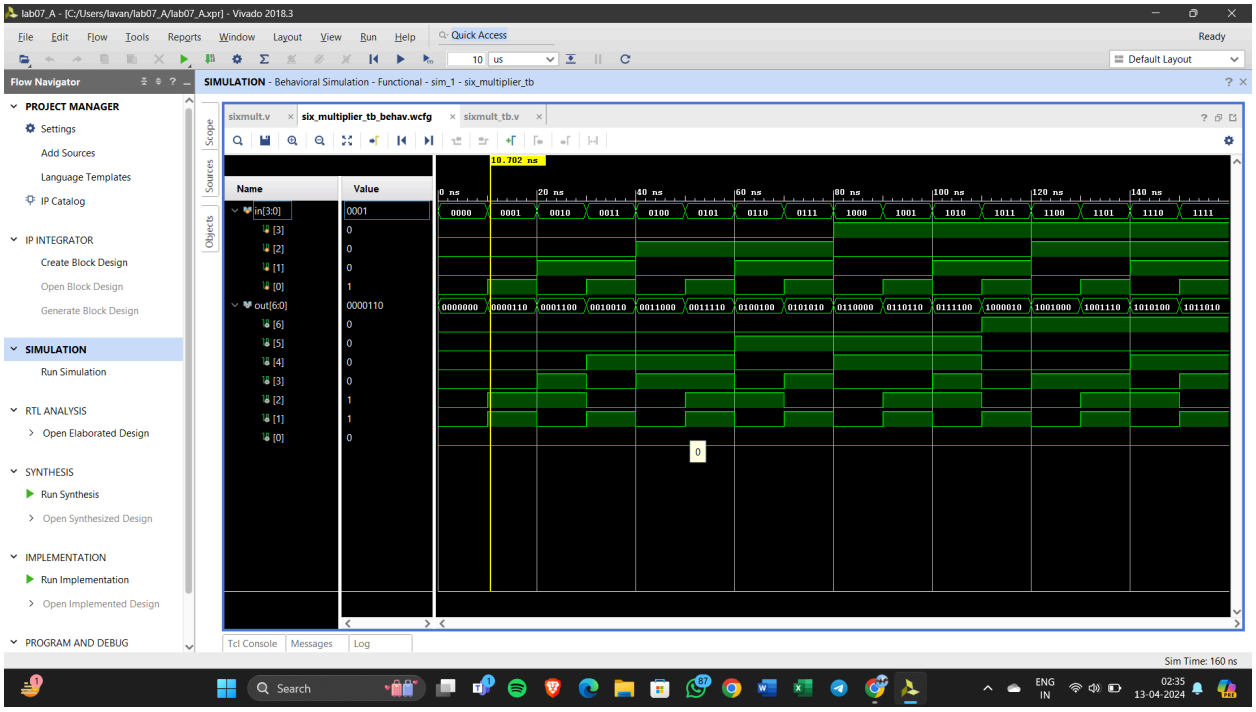
- **Simulation source (if any)**

```
`timescale 1ns / 1ps

module six_multiplier_tb;

    // Testbench signals
    reg [3:0] in;
```

```
    wire [6:0] out;

    // Instantiate the design under test
    six_multiplier dut(
        .in(in),
        .out(out)
    );

    initial begin
        // Initialize input
        in = 4'd0;

        // Apply different inputs and wait 10 time units between each
        #10 in = 4'd1;
        #10 in = 4'd2;
        #10 in = 4'd3;
        #10 in = 4'd4;
        #10 in = 4'd5;
        #10 in = 4'd6;
        #10 in = 4'd7;
        #10 in = 4'd8;
        #10 in = 4'd9;
        #10 in = 4'd10;
        #10 in = 4'd11;
        #10 in = 4'd12;
        #10 in = 4'd13;
        #10 in = 4'd14;
        #10 in = 4'd15;

        #10 $finish; // End simulation
    end

    // Monitor changes and print
    initial begin
        $monitor("Time = %t, Input = %b, Output = %b", $time, in, out);
    end

Endmodule
```

Video Implementation:
Link

Simulation Results (Timing diagram)



Elaborated Design