Digital Design Verilog Assignment

Submitted to Shashi Kant Sir

Submitted by

Name: AMIT

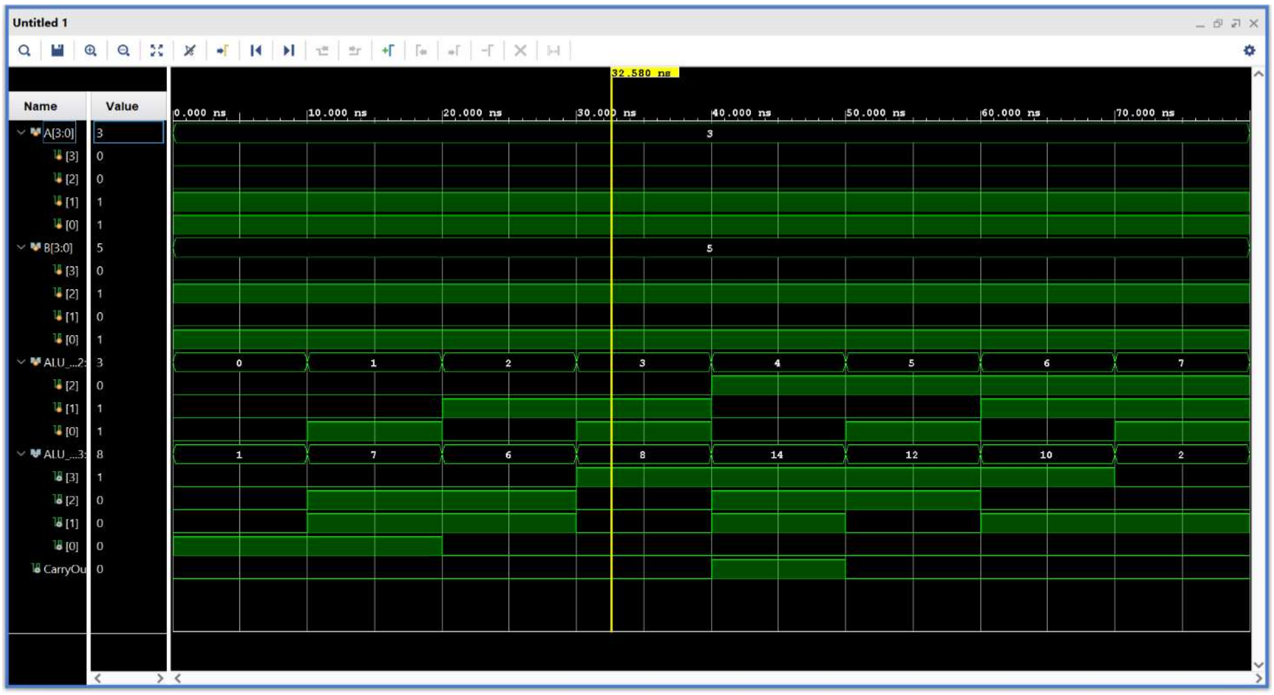
Roll No: KVLSI2501033

1.Write a Verilog code to implement ALU.

Solution Code:-
Design code

```
module alu_4bit_mux (

    input  [3:0] A,
    input  [3:0] B,
    input  [2:0] ALU_Sel,
    output [3:0] ALU_Out,
    output CarryOut);

wire [3:0] and_out, or_out, xor_out, add_out, sub_out, not_a, shl_b, shr_b;
wire add_carry, sub_carry;

// Individual operations
assign and_out   = A & B;
assign or_out    = A | B;
assign xor_out   = A ^ B;
assign {add_carry, add_out} = A + B;
assign {sub_carry, sub_out} = A - B;
assign not_a     = ~A;
assign shl_b     = B << 1;
assign shr_b     = B >> 1;

// 8-to-1 MUX for ALU_Out
assign ALU_Out = (ALU_Sel == 3'b000) ? and_out :
                 (ALU_Sel == 3'b001) ? or_out  :
                 (ALU_Sel == 3'b010) ? xor_out :
                 (ALU_Sel == 3'b011) ? add_out :
                 (ALU_Sel == 3'b100) ? sub_out :
                 (ALU_Sel == 3'b101) ? not_a   :
                 (ALU_Sel == 3'b110) ? shl_b   :
                 (ALU_Sel == 3'b111) ? shr_b   :
                 4'b0000;

// 8-to-1 MUX for CarryOut
assign CarryOut = (ALU_Sel == 3'b011) ? add_carry :
                  (ALU_Sel == 3'b100) ? sub_carry :
                  1'b0;

endmodule
```

```
Testbench Code:-
module tb_alu_4bit_mux;


reg  [3:0] A, B;
reg  [2:0] ALU_Sel;
wire [3:0] ALU_Out;
wire CarryOut;

alu_4bit_mux
uut(.A(A),.B(B),.ALU_Sel(ALU_Sel),.ALU_Out(ALU_Out),.CarryOut(CarryOut));

initial begin
    // Set up the dumpfile and dumpvars
    $dumpfile("alu_4bit_mux_waveform.vcd");  // Name of the VCD file
    $dumpvars(0, tb_alu_4bit_mux);  // Dump all the variables in the testbench
    // Initial values
    A = 4'b0011; B = 4'b0101;
    // Apply test cases
    ALU_Sel = 3'b000; #10;  // AND operation
    ALU_Sel = 3'b001; #10;  // OR operation
    ALU_Sel = 3'b010; #10;  // XOR operation
    ALU_Sel = 3'b011; #10;  // ADD operation
    ALU_Sel = 3'b100; #10;  // SUB operation
    ALU_Sel = 3'b101; #10;  // NOT A operation
    ALU_Sel = 3'b110; #10;  // B << 1 operation
    ALU_Sel = 3'b111; #10;  // B >> 1 operation

    $finish;
end
endmodule
```



Name	Value
A[3:0]	3
[3]	0
[2]	0
[1]	1
[0]	1
B[3:0]	5
[3]	0
[2]	1
[1]	0
[0]	1
ALU_...2:	3
[2]	0
[1]	1
[0]	1
ALU_...3:	8
[3]	1
[2]	0
[1]	0
[0]	0
CarryOu	0

2. Write a Verilog code to design and implement mod-100 counter.

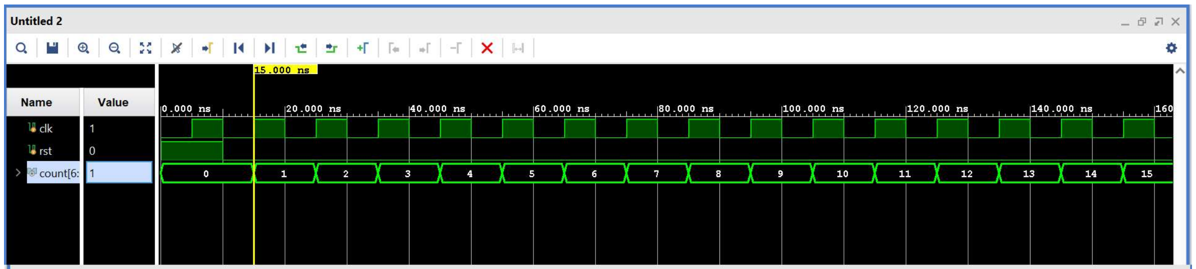
Solution:

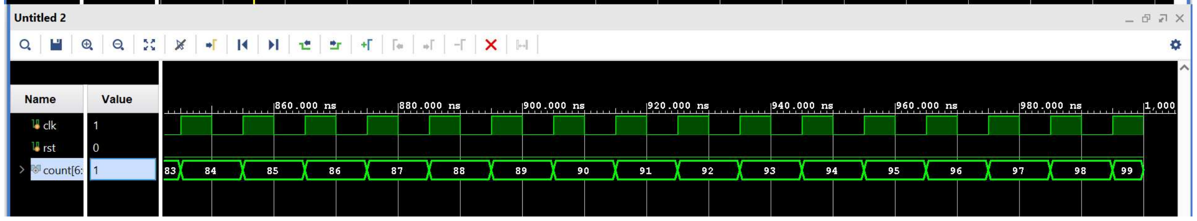
Design Code:

```
module mod100_counter(
    input clk,          // Clock input
    input rst,          // Reset input
    output reg [6:0] count // 7-bit counter to hold values from 0 to 99
);
    // Counter logic
    always @(posedge clk or posedge rst) begin
        if (rst)
            count <= 7'b0000000; // Reset the counter to 0
        else if (count == 99)
            count <= 7'b0000000; // Wrap around to 0 when count reaches 100
        else
            count <= count + 1; // Increment the counter
    end
endmodule
```

Simulation Code:

```
module tb_mod100_counter;

    reg clk;            // Clock signal
    reg rst;            // Reset signal
    wire [6:0] count;   // Counter value
    // Instantiate the mod100 counter
    mod100_counter uut (.clk(clk),.rst(rst),.count(count));
    // Clock generation
    always begin
        #5 clk = ~clk;  // Generate clock with a period of 10 time units
    end
    // Test sequence
    initial begin
        // Initialize signals
        clk = 0;
        rst = 0;
        // Apply reset and check counter
        rst = 1; #10;    // Assert reset for 10 time units
        rst = 0; #10;    // Deassert reset
        // Run counter and check its values
        #1000;  // Simulate for some time to observe counting
        $finish;  // End the simulation
    end
    // Display counter values in a readable format
    initial begin
        $monitor("Time: %0t | Count: %d", $time, count);
    end
endmodule
```



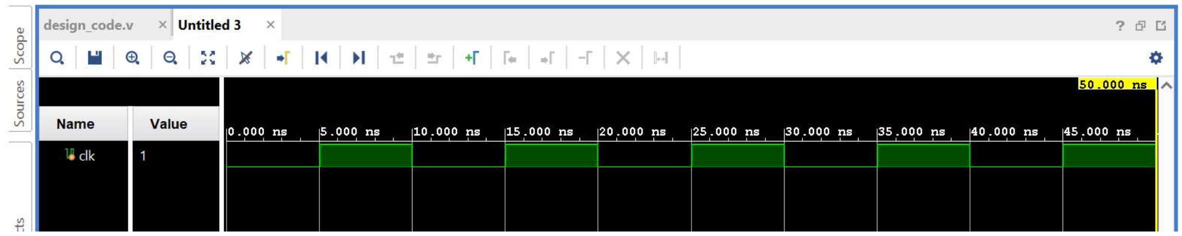


---

3. Write Verilog code in min 4 ways to generate clock.
Solution
Design Code:

```
//method-1 using always block

module clk_gen_always;
    reg clk;
    always #5 clk = ~clk;
    initial begin
        clk = 0;
        $monitor("Time=%0t | clk=%b", $time, clk);
        #50 $finish;
    end

endmodule

//method-2 using initial and forever
module clk_gen_forever;
    reg clk;
    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end
    initial begin
        $monitor("Time=%0t | clk=%b", $time, clk);
        #100 $finish;
    end
endmodule

//method-3 using for loop
module clk_gen_for;
    reg clk;
    integer i;
    initial begin
        clk = 0;
        for (i = 0; i < 10; i = i + 1) begin
```

```
            #15 clk = ~clk;
        end
    end
    initial begin
        $monitor("Time=%0t | clk=%b", $time, clk);
        #160 $finish;
    end
endmodule

//method-4 using repeat statement
module clk_gen_repeat;
    reg clk;
    initial begin
        clk = 0;
        repeat (10) begin
            #7 clk = ~clk;
        end
    end

    initial begin
        $monitor("Time=%0t | clk=%b", $time, clk);
        #100 $finish;
    end
endmodule
```


design_code.v
Untitled 3
Scope
Sources
50.000 ns
Name
Value
clk
1
0.000 ns
5.000 ns
10.000 ns
15.000 ns
20.000 ns
25.000 ns
30.000 ns
35.000 ns
40.000 ns
45.000 ns