Note: Lab Report sample is given to you in Lab folder. Do check it.

1. Addition of two usigned interger binary number.

Design:

```
module UnsignedAdder(
    input [3:0] A, // 4-bit input A
    input [3:0] B, // 4-bit input B
  output [3:0] Sum, // 5-bit output for the sum (to handle carry)
    output CarryOut // Output for the carry out
);
    // Internal signals
  reg [3:0] sum_temp; // Temporary sum to handle overflow
    reg carry_temp; // Temporary carry

    // Add A and B
  always @(A,B) begin
        {carry_temp, sum_temp} = A + B;
    end

    // Output signals
    assign CarryOut = carry_temp; // Output the carry out
    assign Sum = (carry_temp) ? {1'b1, sum_temp} : sum_temp; //
Adjust sum based on carry
endmodule
```

Testbench:

```
module testbench;

    // Inputs
    reg [3:0] A;
    reg [3:0] B;

    // Outputs
  wire [3:0] Sum;
    wire CarryOut;

    // Instantiate the UnsignedAdder module
```

```
  UnsignedAdder dut (A,B,Sum,CarryOut);

    // Stimulus generation
    initial begin

      $dumpfile("dump.vcd");
      $dumpvars(1);

        // Test case 1: A = 3, B = 5
        A = 4'b0011;
        B = 4'b0101;
        #10; // Wait for 10 time units
        // Display simulation results
        $display("A = %b, B = %b, Sum = %b, CarryOut = %b", A, B,
Sum, CarryOut);

        // Test case 2: A = 7, B = 9
        A = 4'b0111;
        B = 4'b1111;
        #10; // Wait for 10 time units
      // Display simulation results
        $display("A = %b, B = %b, Sum = %b, CarryOut = %b", A, B,
Sum, CarryOut);

    end
endmodule
```

2. Subtraction of two usigned interger binary number.

Design:

```
module BinarySubtractor4Bit(
    input [3:0] A,       // 4-bit input for the first binary number
    input [3:0] B,       // 4-bit input for the second binary number
    output reg [3:0] diff,  // 4-bit output for the difference
    output reg borrow_out   // Output for the final borrow
);

  reg borrow;  // Temporary variable to track borrow between stages
```

```
  integer i;

  always @* begin
    borrow = 0;
    for (i = 0; i < 4; i = i + 1) begin
      // Subtract A[i] - B[i] - borrow
      diff[i] = A[i] ^ B[i] ^ borrow;

      // Calculate borrow for the next bit
      borrow = (~A[i] & B[i]) | ((~A[i] | B[i]) & borrow);
    end
    borrow_out = borrow; // Final borrow after 4-bit subtraction
  end
endmodule
```

Testbench:

```
module testbench;

    // Inputs
    reg [3:0] A;
    reg [3:0] B;

    // Outputs
  wire [3:0] diff;
    wire borrow_out;

    // Instantiate the UnsignedAdder module
  BinarySubtractor4Bit dut (A,B,diff,borrow_out);

    // Stimulus generation
    initial begin

      $dumpfile("dump.vcd");
      $dumpvars(1);

        // Test case 1: A = 3, B = 5
        A = 4'b0101;
        B = 4'b0011;
```

```
        #10; // Wait for 10 time units
        // Display simulation results
      $display("A = %b, B = %b, diff = %b, borrow_out = %b", A, B,
diff, borrow_out);

        // Test case 2: A = 7, B = 9
        A = 4'b0111;
        B = 4'b1111;
        #10; // Wait for 10 time units
      // Display simulation results
      $display("A = %b, B = %b, diff = %b, borrow_out = %b", A, B,
diff, borrow_out);

      // Test case 2: A = 4, B = 7
        A = 4'b0100;
        B = 4'b0111;
        #10; // Wait for 10 time units
      // Display simulation results
      $display("A = %b, B = %b, diff = %b, borrow_out = %b", A, B,
diff, borrow_out);

    end
endmodule
```