EEL2020 Digital Design Lab Report
Sem II AY 2023-24

Experiment No.	:	**04**
Name	:	**Maulik Desai**
Roll No.	:	**B22CS033**
Partner Name (Partner Roll No.)	:	**Jay Mehta (B22CS034)**

Objective

Simulate and implement a BCD Adder/Subtractor circuit.
The selection of addition/subtraction operation should be done by another input. The sign (in case of subtraction) and carry (in case of addition) should be shown on an LED, while the BCD difference or sum should be shown on a 7-segment display. You may use Verilog operators of addition and subtraction.

Logic Design

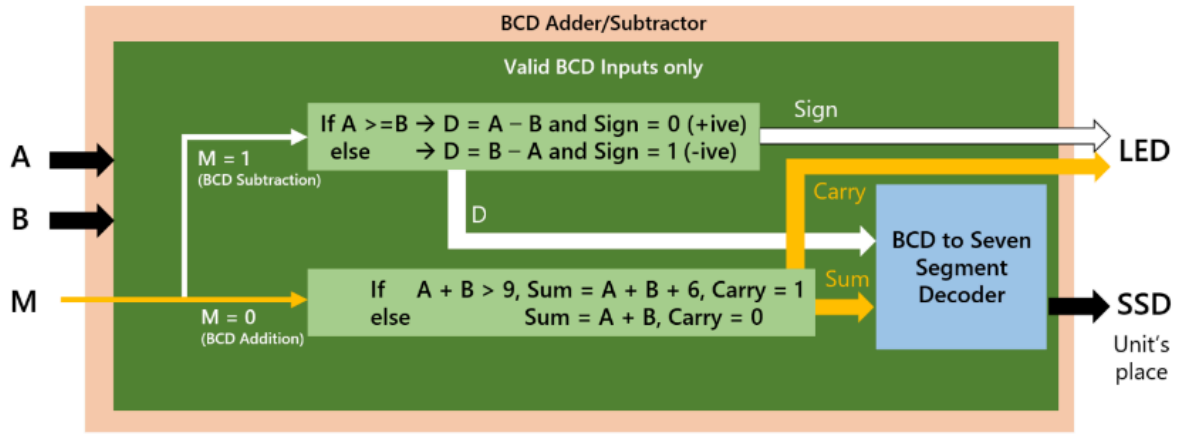


Figure 1 Module structure

High level logic design for the experiment

We take in 3 inputs, two 4-bit integers and one control. When the control is 0, the circuit acts as a BCD adder, else at 1 it acts as a BCD Subtractor. During addition, if the sum goes beyond 9, then the LED glows to depict carry. During subtraction, the LED depicts negative sign.

Source Description

- Design source

The source file is a bcd_AS.v module that generates an output to display the sum or subtraction of two four-bit numbers based on the value of the control input given.

- Constraint file

The PYNQ XDC file was updated with the following changes:

Ports (from Verilog module)	**Designation** (Input/Output)	PYNQ Component Type (Button/LED/Switch etc. along with number, eg. LD01, BTN2, etc.)	Pin Configuration (from the PYNQ User Manual)
seg_out[0]	Output	JB1_P	W14
seg_out[1]	Output	JB1_N	Y14
seg_out[2]	Output	JB2_P	T11
seg_out[3]	Output	JB2_N	T10
seg_out[4]	Output	JB3_P	V16
seg_out[5]	Output	JB3_N	W16
seg_out[6]	Output	JB4_P	V12
c_out_sign	Output	LED0	R14

The RPI XDC file was updated with the following changes:

Ports (from Verilog module)	**Designation** (Input/Output)	RPI Component Type (Button/LED/Switch etc. along with number, eg. LD01, BTN2, etc.)	Pin Configuration (from the RPI User Manual)
B[0]	Input	RPIO_14	V6
B[1]	Input	RPIO_15	Y6
B[2]	Input	RPIO_16	B19
B[3]	Input	RPIO_17	U7
A[0]	Input	RPIO_18	C20

A[1]	Input	RPIO_19	Y8
A[2]	Input	RPIO_20	A20
A[3]	Input	RPIO_26	W9

- Simulation source

We have used a test bench which tests for the following cases (each case runs for 40ns):-

1) **M = 1**
   A = 1001
   B = 0100
   (9-4=5 ⇒ LED won't glow, c_out=5)

2) **M = 1**
   A = 0100
   B = 1000
   (4-8=-4) ⇒ LED will glow, c_out=4)

3) **M = 1**
   A = 1011
   B = 0101
   (A is invalid – Blank SSD output)

4) **M = 0**
   A = 0010
   B = 0110
   (2+6=8 ⇒ LED won't glow, c_out=8)

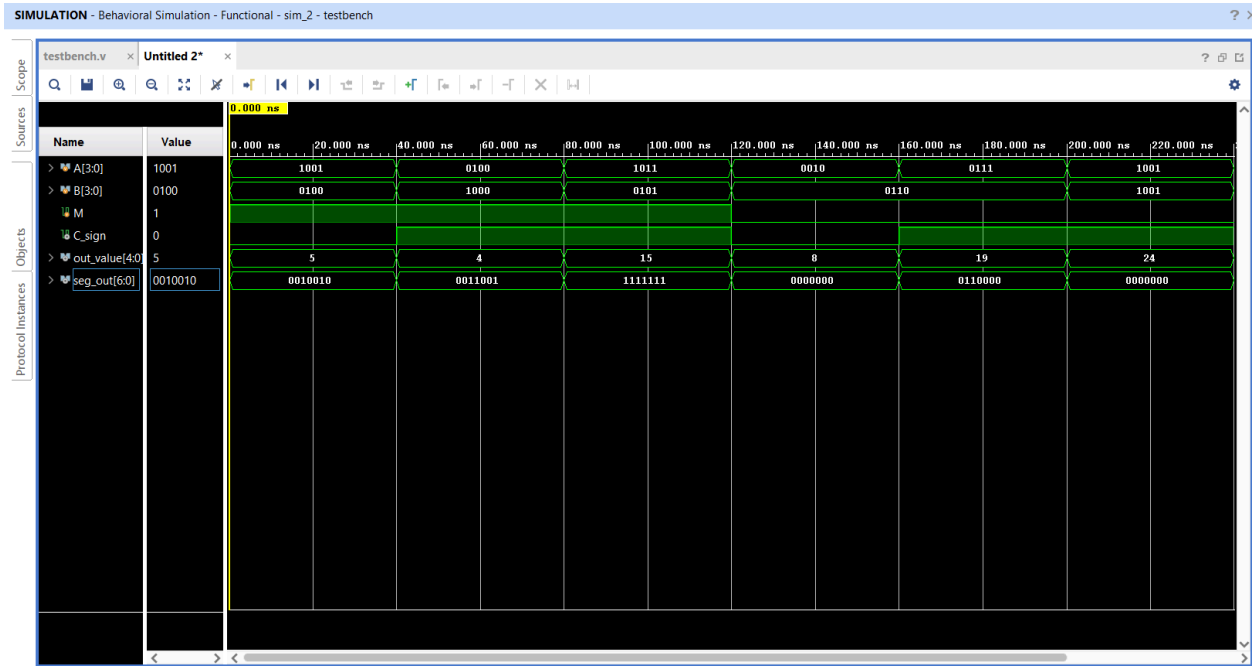
5) **M = 0**
   A = 0111
   B = 0110
   (7+6=13 ⇒ LED will glow, c_out=13+6=19)

6) **M=0**
   A = 1001
   B = 1001
   (9+9=18 ⇒ LED will glow, c_out=18+6=24)

Following is a tabular representation of the used test cases in the test bench. The results can be cross-verified through the timing diagram.

Input			Output	
A	***B***	***M***	***c_out***	***c_out_sign***
1001	**0100**	**1**	**5**	**0**
0100	**1000**	**1**	**4**	**1**
1011	**0101**	**1**	**- NA -**	**- NA -**
0010	**0110**	**0**	**8**	**0**
0111	**0110**	**0**	**19**	**1**
1001	**1001**	**0**	**24**	**1**

Simulation Results (Timing diagram)



Codes:

1. **bcd_AS.v**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 27.02.2024 19:38:49
// Design Name:
// Module Name: bcd_AS
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module bcd_AS (
    input [3:0] A,
    input [3:0] B,
    input M,
    output reg [4:0] out_value,
    output reg [6:0] seg_out,
    output reg C_sign
);

always @(*)
begin
    if (A <= 4'b1001 && B <= 4'b1001) begin
        if (M == 1) begin
```

```
        if (A >= B) begin
          out_value = A - B;
          C_sign = 0;
        end
        else begin
          out_value = B - A;
          C_sign = 1;
        end
      end
      else begin
        out_value = A + B;
        if (out_value > 4'b1001) begin
          C_sign = 1;
          out_value = out_value + 4'b0110;
        end
        else
          C_sign = 0;
      end
      $display ("A = %b, B = %b, Subtraction?: %b, C_out_Sign: %b, Output value: %d", A,
B, M, C_sign, out_value);
    end
    else begin
      out_value = 4'b1111;
      $display ("Invalid BCD input");
    end
end

always @(*)
begin
    case (out_value[3:0])
      4'b0000: seg_out = 7'b1000000; // Display 0
      4'b0001: seg_out = 7'b1111001; // Display 1
      4'b0010: seg_out = 7'b0100100; // Display 2
      4'b0011: seg_out = 7'b0110000; // Display 3
      4'b0100: seg_out = 7'b0011001; // Display 4
      4'b0101: seg_out = 7'b0010010; // Display 5
      4'b0110: seg_out = 7'b0000010; // Display 6
      4'b0111: seg_out = 7'b1111000; // Display 7
      4'b1000: seg_out = 7'b0000000; // Display 8
      4'b1001: seg_out = 7'b0010000; // Display 9
      default: seg_out = 7'b1111111; // Blank display for other values
    endcase
```

```
end

endmodule


module add_sub (input [3:0] A, [3:0] B, input M, output [6:0] seg_out, output c_out_sign);
BCD_Addition_Subtraction addition_sub (A, B, M, out_val, seg_out, c_out_sign);
endmodule
```

---

2. testbench.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 27.02.2024 19:59:41
// Design Name:
// Module Name: testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module testbench;
  reg [3:0] A;
  reg [3:0] B;
  reg M;
  wire C_sign;
```

```
wire [4:0] out_value;
wire [6:0] seg_out;

bcd_AS uut (
    .A(A),
    .B(B),
    .M(M),
    .out_value(out_value),
    .seg_out(seg_out),
    .C_sign(C_sign)
);


initial begin
    // Test case 1
    M = 1;      // Subtractor
    A = 4'b1001; // 9
    B = 4'b0100; // 4
    // 9 - 4 = 5
    // out_value = 5, c_sign = 0
    #40;

    // Test case 2
    M = 1;      // Subtractor
    A = 4'b0100; // 4
    B = 4'b1000; // 8
    // 4 - 8 = -4
    // out_value = 4, c_sign = 1
    #40;

    // Test case 3
    M = 1;      // Subtractor
    A = 4'b1011; // Invalid BCD
    B = 4'b0101; // 5
    // Blank SSD display
    #40;

    // Test case 4
    M = 0;      // Adder
    A = 4'b0010; // 2
    B = 4'b0110; // 6
    // 2 + 6 = 8
```

```
        // out_value = 8, c_sign = 0
        #40;

        // Test case 5
        M = 0;      // Adder
        A = 4'b0111; // 7
        B = 4'b0110; // 6
        // 7 + 6 = 13
        // out_value = 13+6 = 19, c_sign = 1
        #40;

        // Test case 6
        M = 0;      // Adder
        A = 4'b1001; // 9
        B = 4'b1001; // 9
        // 9 + 9 = 18
        // out_value = 18+6 = 24, c_sign = 1
        #40;

        $finish;
    end
endmodule
```
