ELL201 Project Report
Calculator

Aniket Gupta – 2023MT10667
Abdul Haleem Ahmed – 2023MT60069
Ashvath Hari Bhaskar - 2023EE10168
Jayesh Narayanan – 2023EE11048

April 29, 2025

Contents

1 Implementation Methodology

1.1 Problem Statement

To design and implement a Verilog simulation and run on CPLD Hardware a calculator which is capable of doing 3 types of operations namely, addition, subtraction, and multiplication on two 5-bit signed operands, to give a 5-bit output (in case of overflow the extra bits are clipped off).

1.2 Implementation Methodology

The calculator was implemented as a finite state machine with the following design:

1. **State Control**
   - Single state switch advances through all states
   - Four states: Operation → Input 1 → Input 2 → Output → Operation
   - Dedicated reset switch for initialization
2. **Operation Selection**
   - 2-bit encoding: 01=Add, 10=Subtract, 11=Multiply
   - Originally planned custom units but used Verilog built-in operators
3. **Data Handling**
   - 5-bit signed integers (-16 to +15) in two's complement
   - Same range for both input and output
4. **Hardware Constraints**
   - CPLD had only 64 macrocells total
   - Forced to use Verilog built-in arithmetic operators
   - Modulo operation completely omitted
5. **Verification**
   - State LEDs used to verify transitions.
   - Testbench simulation for all operations.
   - Physical testing on CPLD board for all operations.

1.3 State Transition diagram

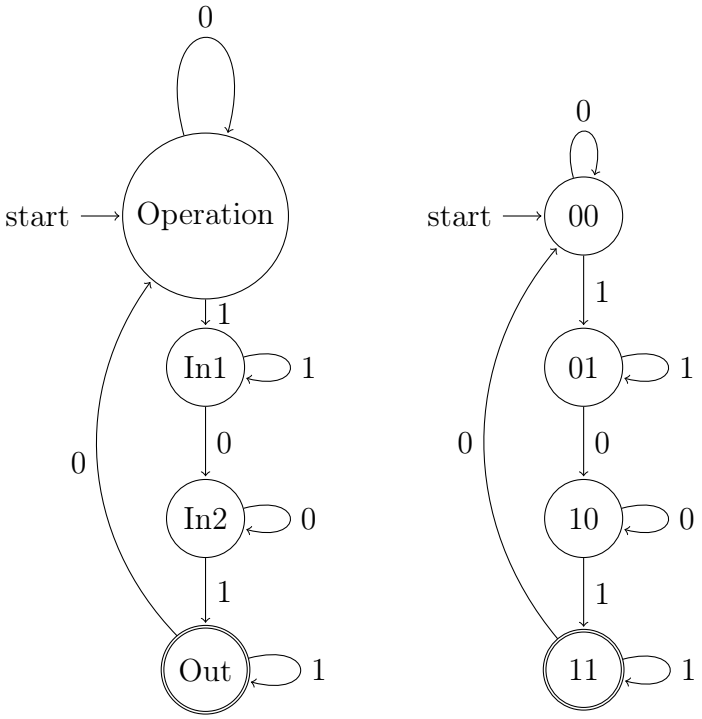

Figure 1: FSM diagrams corresponding to the design

Prev AB		Input x	Next AB	
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Table 1: State transition table

A\Bx	00	01	11	10
0	0	0	0	1
1	1	1	1	0

Figure 2: K-map for D_A with groupings

The simplified expression for D_A is:

$$D_A = AB' + Ax + A'Bx'$$

A \Bx	00	01	11	10
0	0	1	1	0
1	0	1	1	0

Figure 3: K-map for D_B with groupings

The simplified expression for D_B is:

$$D_B = x$$

1.4 Circuit Diagram for State Transitions

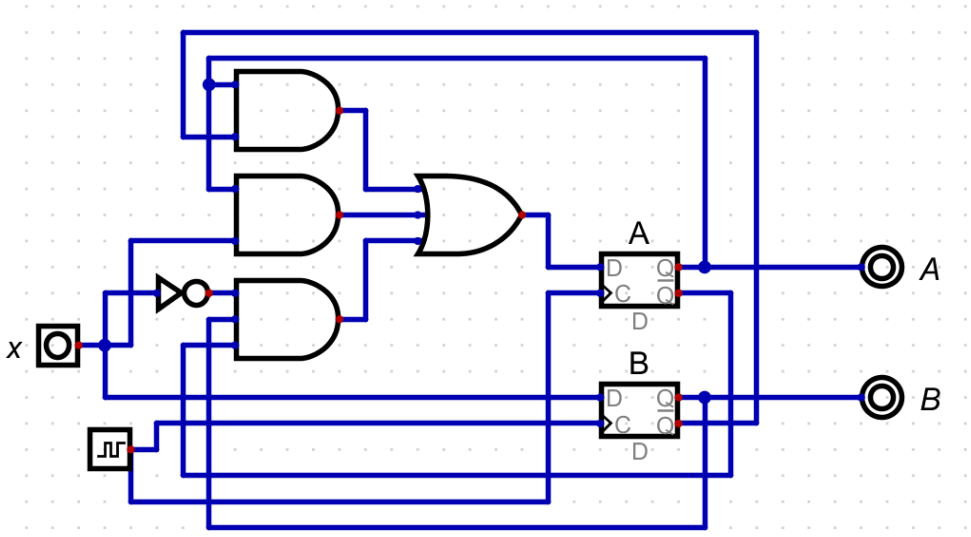

Figure 4: Circuit Diagram

2 Verilog Code

2.1 Design Code

```
module d_flipflop(
input D,clk,reset,
output reg Q);

always @(posedge clk)
begin
    if (~reset) begin
        Q<=0;
    end
    else begin
        Q <= D;
    end
end
endmodule

module calculator(
input signed [4 : 0] x,
input clk,s,rs,
output reg [5 :0] y,
output A_out,B_out
);


wire Da,Db;
reg signed [4:0] In1, In2;
reg [1:0] op;

assign Da = A_out&(~B_out) | A_out&s | (~A_out)&(B_out)&(~s);
assign Db = s;

d_flipflop d1(Da,clk,rs,A_out);
d_flipflop d2(Db,clk,rs,B_out);

always @(*)
begin
    case ({A_out,B_out})
        2'b00: begin
            In1 <= 5'b00000;
            In2 <= 5'b00000;
            op <= {x[1],x[0]};
            y <= {x[1],x[0]};
        end
        2'b01: begin
            In1 <= x;
            y <= x;
        end
        2'b10: begin
            In2 <= x;
```

```
            y <= x;
        end
        2'b11: begin
            case (op)
                2'b01: y <= In1 + In2;
                2'b10: y <= In1 - In2;
                2'b11: y <= In1*In2;
            endcase
            end
    endcase
end
endmodule
```

2.2 Testbench Code

```
module t_dff();
  reg clk, s, rs;
  reg signed [4 :0] in
  reg signed [5:0] out;
  wire A_out, B_out;
  calculator add(in, clk,s,rs,out,A_out,B_out);
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
      s=0;
      rs = 0;
      clk = 1;
      #25
      clk = 0;
      #25
      clk = 1;
      #12
      s = 0;
      rs = 1;
      #10
      in = 5'b000001;
      #3
      clk = 0;
      #25
      clk = 1;
      #12
      s = 1;
      #13
      clk = 0;
      #25
      clk = 1;
      #25
      in = 5'b00101;
      clk = 0;
      #25
```

```
37        clk = 1;
38        #12
39        s = 0;
40        #13
41        clk = 0;
42        #25
43        clk = 1;
44        #10
45        in = 5'b00111;
46        #15
47        clk = 0;
48        #25
49        clk = 1;
50        #25
51        clk = 0;
52        s = 1;
53        #25
54        clk = 1;
55        #25
56        clk = 0;
57        #25
58        clk = 1;
59
60      end
61  endmodule
```

3 Testbench Waveform

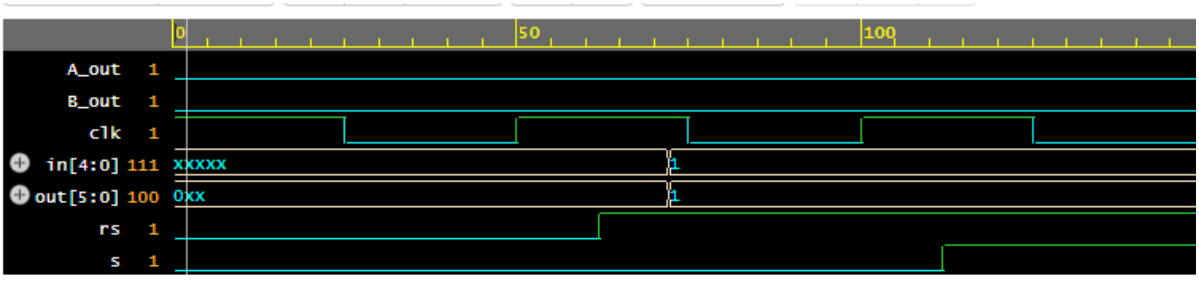

Figure 5: Testbench Waveform 0-150ms

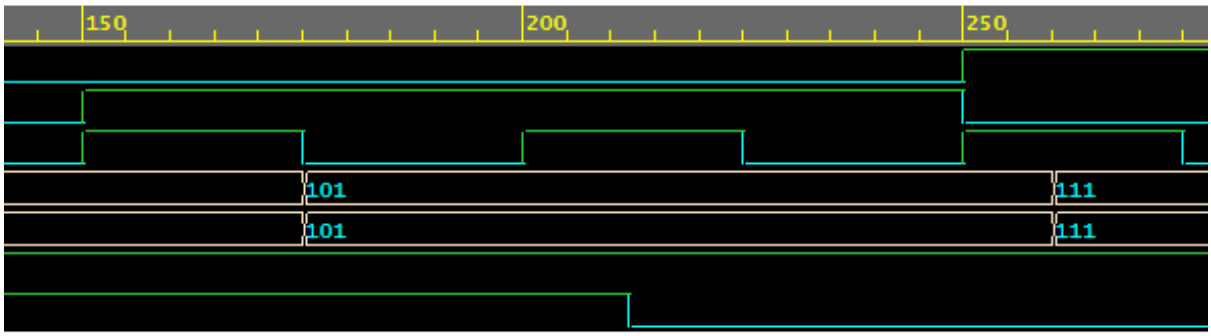

Figure 6: Testbench Waveform 150-300ms

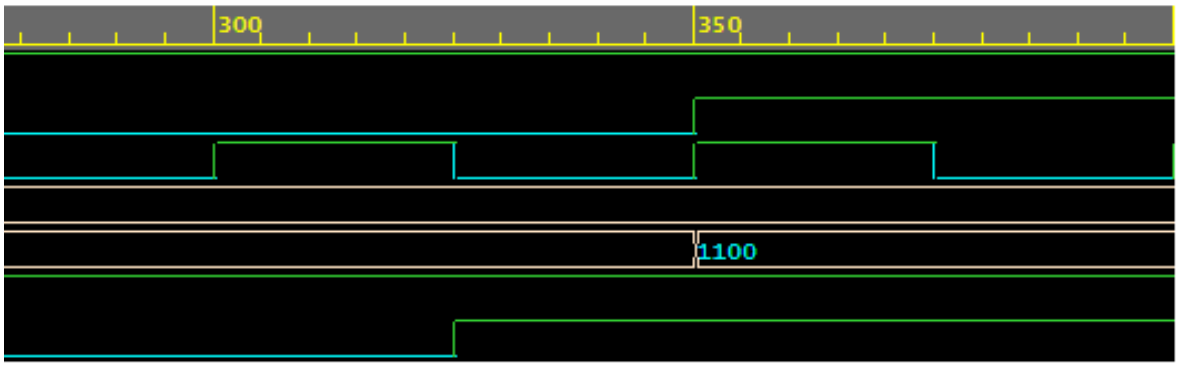

Figure 7: Testbench Waveform 300-400ms

4 Verification of Outputs on CPLD Board

4.1 CPLD Input and Output Assignments

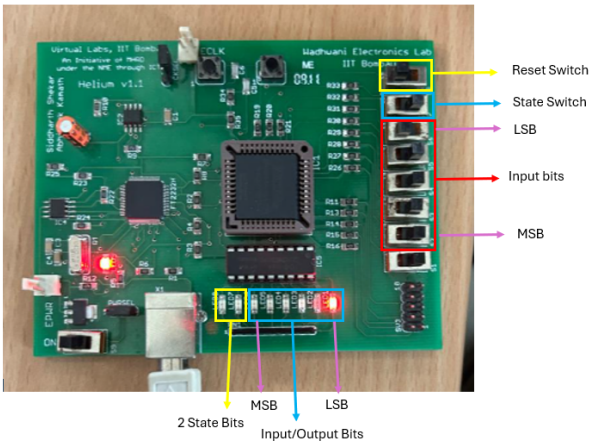

Figure 8: Input and Output Assignments

4.2 Test Cases

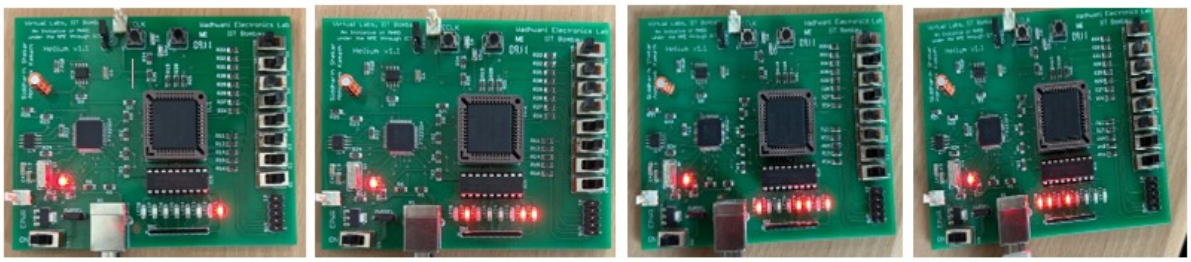

Figure 9: Addition: 3+13=16

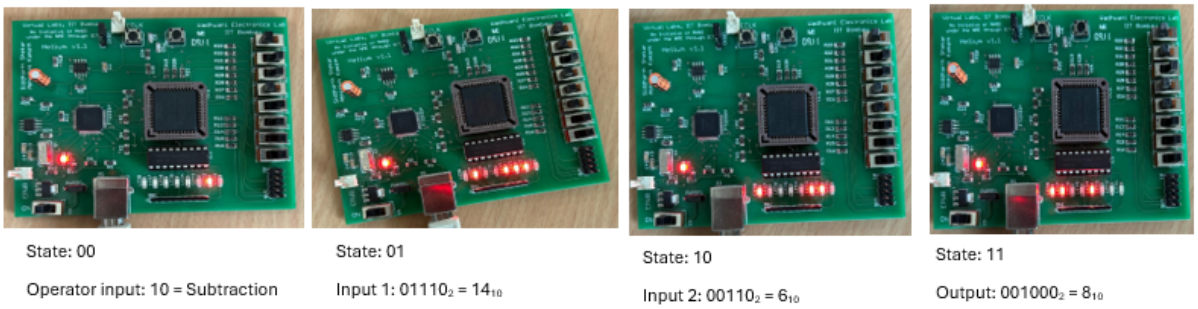

Figure 10: Subtraction: 14-6=8

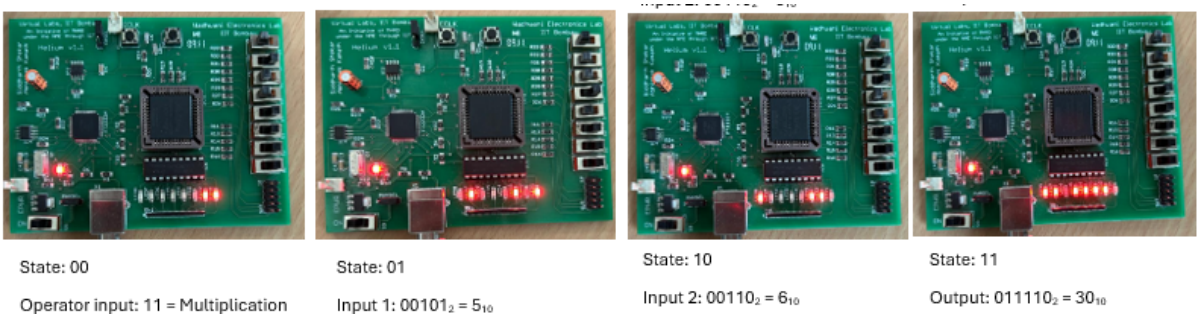

Figure 11: Multiplication: 5*6=30

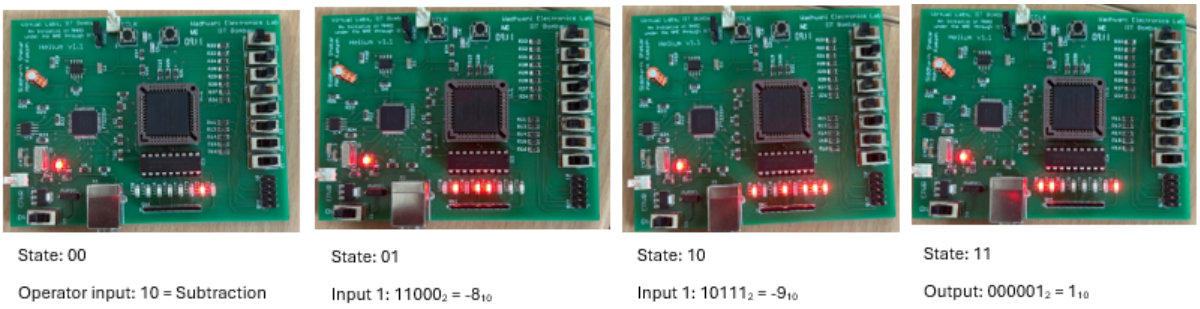

Figure 12: Subtraction: -8-(-9)=1

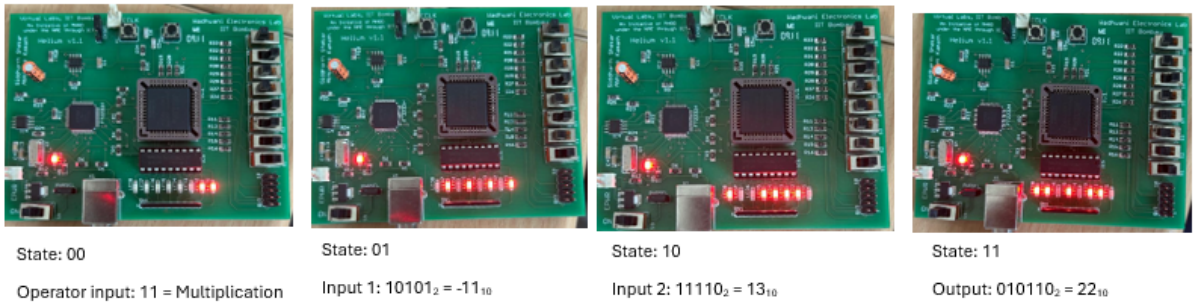

Figure 13: Multiplication: -11*(-2)=22

5 Conclusion

We have successfully designed and implemented a fully functional calculator (adder, subtractor, and multiplier) using a CPLD board programmed in Verilog. By utilizing 63 out of the 64 available macrocells in the Quartus software, we maximized the CPLD's logic capacity, demonstrating near-optimal resource efficiency. The design was rigorously validated through manual verification and testbench waveform analysis, confirming correct functionality under all test cases. This project highlights the potential of CPLD-based embedded systems for arithmetic applications while operating within constrained hardware limits.