COL215 – Digital Logic and System Design

Department of Computer Science & Engineering, IIT Delhi

Semester I, 2025-26

Lab Assignment 4(Friday)

Multiply-Accumulate (MAC) unit

Developed By :

Bharmal Bhakar 2024CS10403
Ritik Raj 2024CS10086

Introduction

In this assignment, we have designed a Multiply-Accumulate (MAC) unit that performs the function:

a = a + (b × c)

b and c are multiplied, and the result is accumulated into a. The accumulation operation is performed for a sequence of different pairs of b and c values. In the process, we get familiar with:

1. Resetting a design to a known initial state.

2. The concept of debouncing. When a button is pushed, there might be some brief oscillation due to imperfect contact; the design needs to anticipate this.

3. The design goals included capturing inputs reliably from slider switches, debouncing a mechanical push button, detecting overflow conditions in the accumulator, and providing informative visual indicators on the Basys3 board.

Design Decisions

We have taken reference from our lab 3 assignment, which was about displaying digits on the 7-segment display. We implemented our module by using the following approaches:

Design Decisions

1. Variables

Pin	Function
BTNC	Reset pin
SW12	Enable for MAC to accumulate the product
SW11	SW7–SW0 data is sampled as MAC input c
SW10	SW7–SW0 data is sampled as MAC input b
SW7–SW0	Keys to input 8-bit value
LED15–LED0	Accumulated value

Anodes – ano – an1,
Cathodes – a – g,
Clock – clk

Defined variables

```
reg [7:0] store_B, store_C;
reg [15:0] store_A;
reg [1:0] active_anode = 0;
reg [16:0] counter = 0;
parameter WIDTH = 20;
reg [WIDTH-1:0] counter2 = 0;
reg sw_sync0, sw_sync1;
reg sw12_delayed;
wire sw12_rising;
reg overflow;
```

Active anode = which anode is activated
Counter = clock divider to slow down active anode cycling
Parameter = bit width of debounce counters
Counter2 = debounce counter for sw12
Sw_sync0, sw_sync1 = two stage synchronizers
Sw12-delayed = holds the previous debounced value of sw12
Sw12_rising = becomes 1 for 1 clock cycle whenever sw12 goes from 0 to 1;
Overflow = temp overflow flag

2. Taking input and storing it :

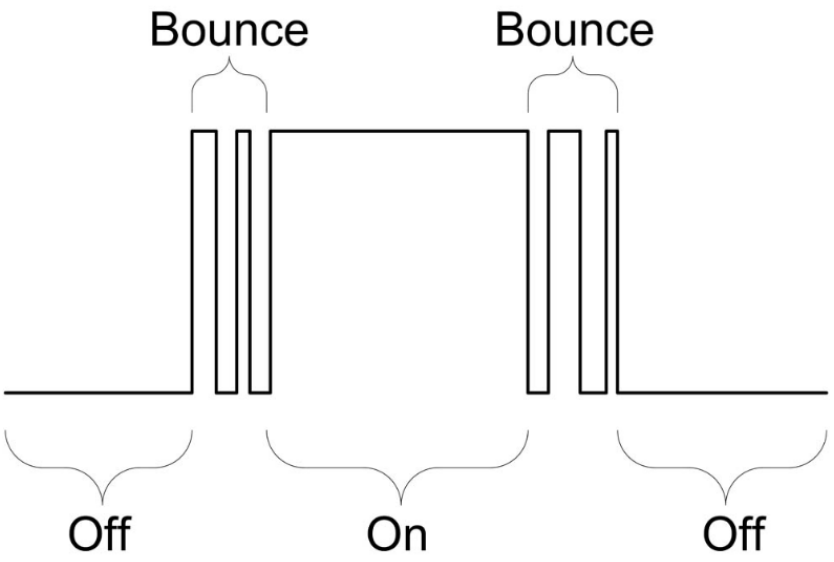
```
always @(posedge clk) begin
    if (sw10) store_B <= sw;
    else if(sw11) store_C <= sw;
end
```

Inputs b and c are stored in registers (store_B, store_C) when sw10 or sw11 are high. This ensures stable inputs for multiplication, even if the switches change, because we have already stored them in our registers.

3. Debouncing:

```
always @(posedge clk) begin
    if(sw_sync1 == sw12_debounced) counter2 <= 0;
    else begin
        counter2 <= counter2 + 1;
        if(counter2 == {WIDTH{1'b1}}) begin
            sw12_debounced <= sw_sync1;
            counter2 <= 0;
        end
    end
end
```

The push button on the Basys 3 board may lead to bouncing. We need to anticipate this, and correct it by a debouncing logic. When a button is pressed, the associated signal ideally changes from LOW to HIGH, but due to metallic contacts, the signal transition may contain glitches, as shown in Figure



Glitched when operating push button

A counter-based debouncing circuit with two-stage synchronizers is used for both SW12 and BTNC to prevent glitches caused by mechanical bouncing.

4. Accumulator & Overflow:

```
always @(posedge clk) begin
    if(rst_debounced) begin
        store_A <= 16'b0;
        overflow <= 1'b0;
    end
    else if(sw12_rising) begin
        if(sum > 16'hFFFF) begin
            store_A <= 16'b0;
            overflow <= 1'b1;
        end else begin
            store_A <= sum[15:0];
            overflow <= 1'b0;
        end
    end
end
```

We have named accumulator as store_A. The accumulator is 16-bit. Whenever sw12 is turned on the stored value of b and c is multiplied and accumulated in store_A. while addition is widened to 17 bits to detect overflow. On overflow “OFLO” is shown on the 7-segment display.

5. Reset Handling:

BTNC is the reset button. The reset button clears the accumulator and shows “-rSt” for 5 seconds on the 7-segment display, confirming reset to a known state. The accumulator is reseted to 0.

6. **Output Display:**

Accumulator value is mapped directly to 16 LEDs for real-time observation. It is displayed in the binary form.

7-segment displays are reserved for showing only special events (reset or overflow), with clock division used for multiplexing. Overflow is showed till until the value of accumulator is not reset.

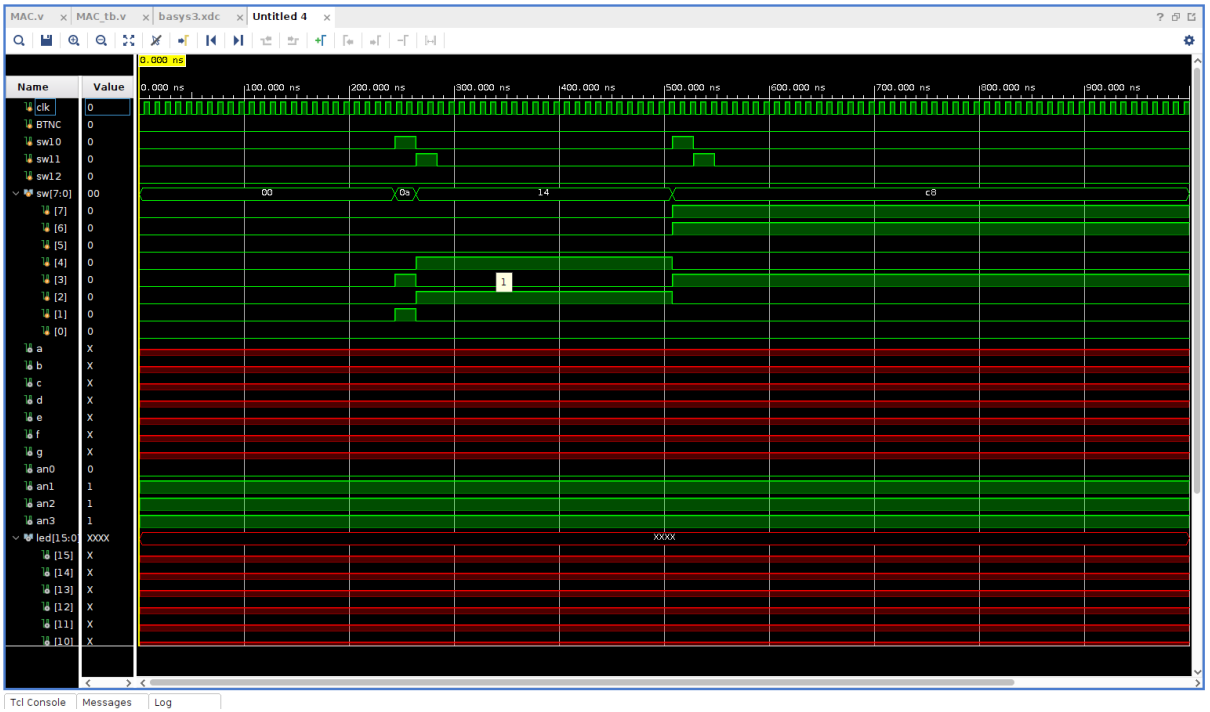
7. **Multiplier Implementation:**

The Behavioral * operator is used for 8-bit multiplication, letting the FPGA infer efficient DSP hardware.

8. **Rising Edge Detection:**

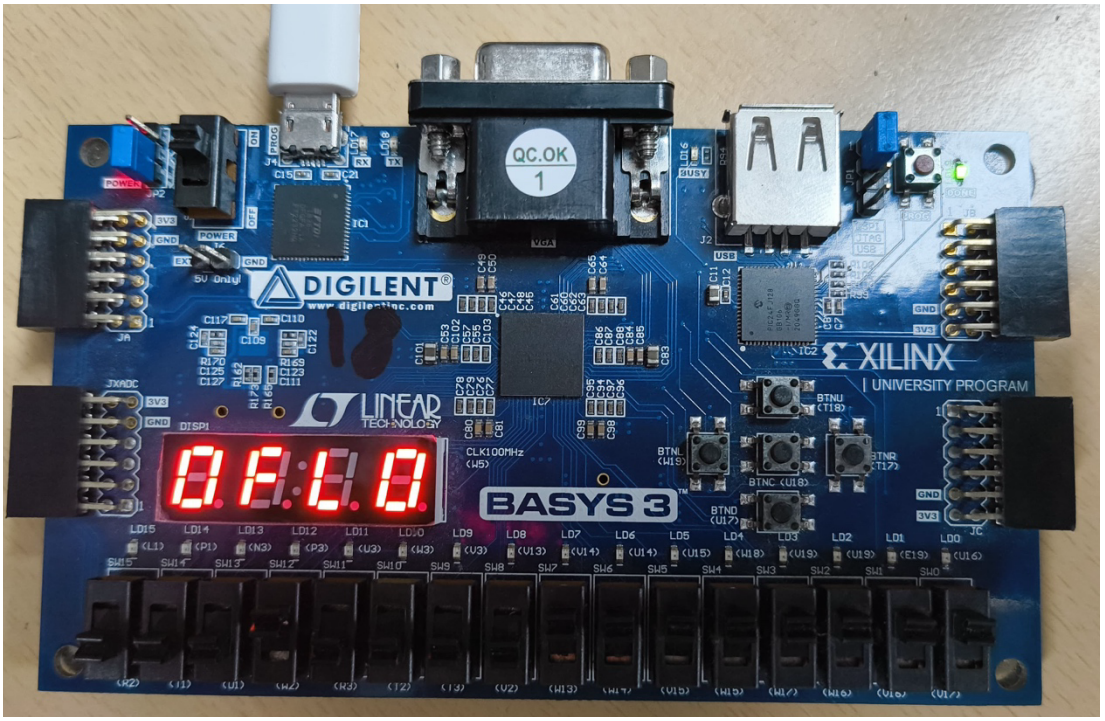
Accumulation is triggered only on the rising edge of sw12, ensuring a single MAC operation per toggle.

Simulation snapshots

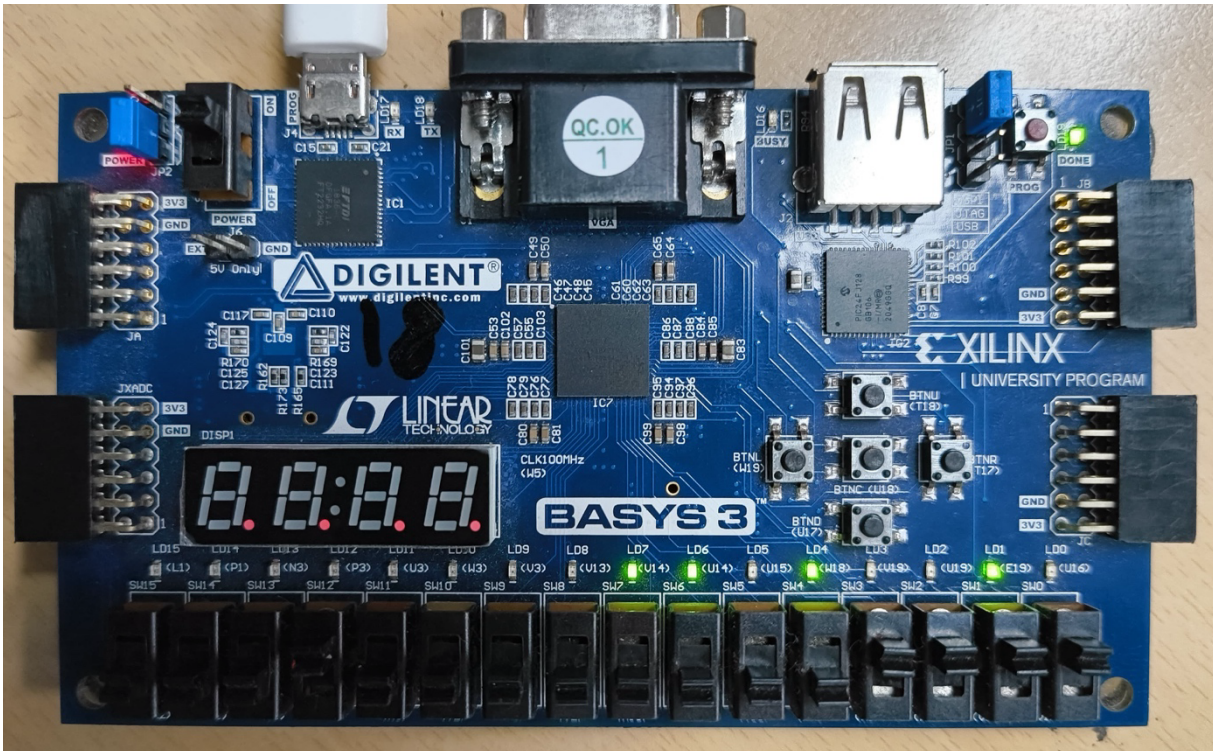


It shows the clock (clk), input switches (sw), and accumulated outputs (leds). The waveform confirms that when inputs b and c are loaded, their product is added to the accumulator correctly, updating the result at each clock cycle.

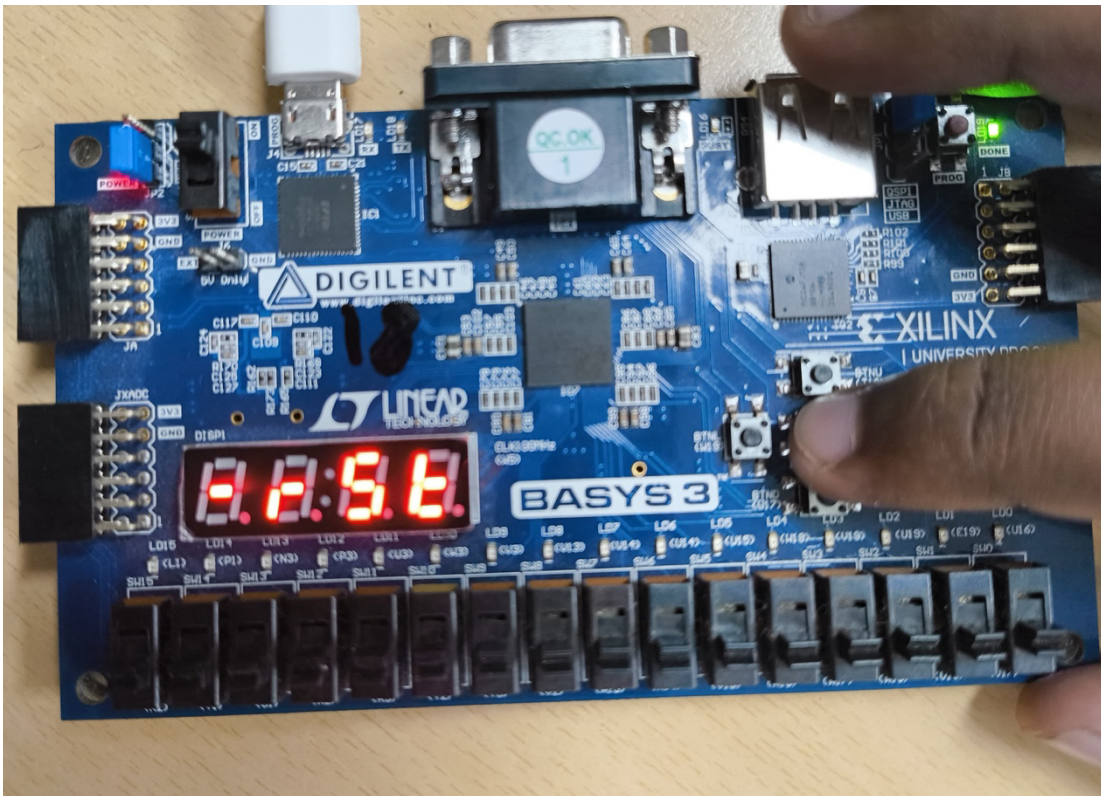
On board implementation



The picture in which display is showing overflow as OFLO

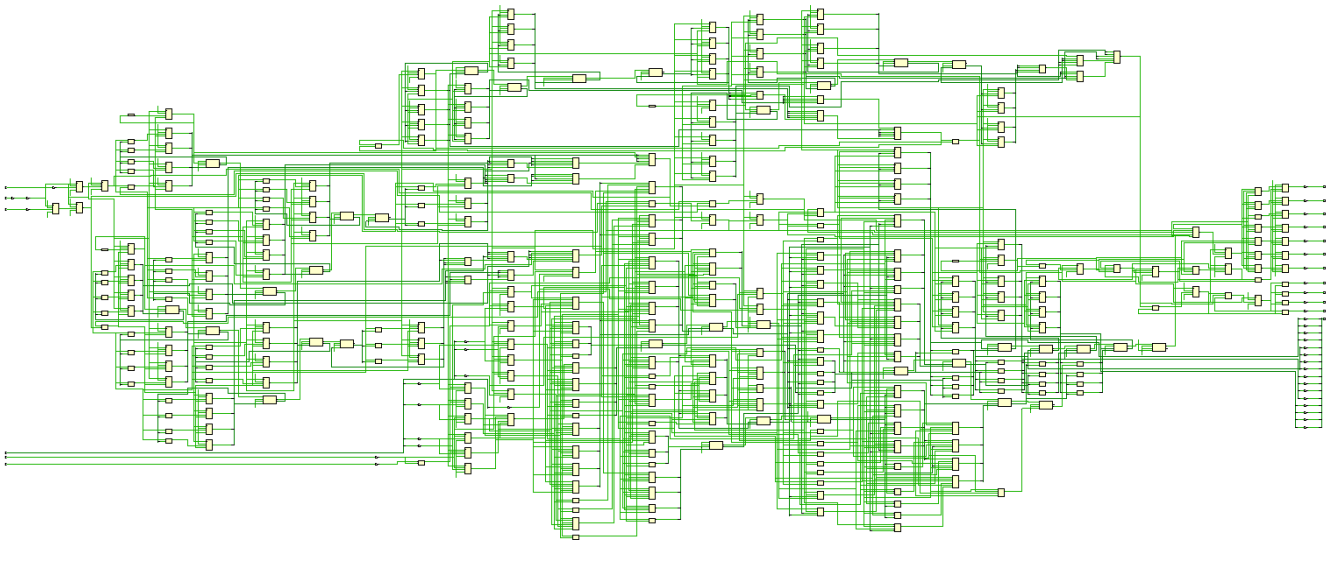


Leds showing a in binary form



Display showing reset

Generated schematics



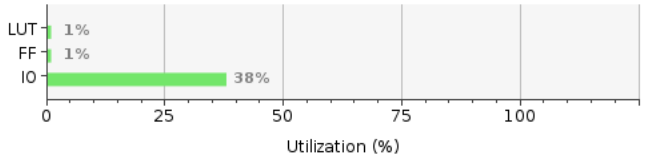
Synthesis report

Utilization | Tcl Console | Messages | Log | Reports | Design Runs | DRC | Methodology | Power | Timing

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start
synth_1	constrs_1	synth_design Complete!												140	99	0	0	0	8/22/25, 4:17 PM
impl_1	constrs_1	route_design Complete!	0.572	0.000	0.246	0.000		0.000	0.082	0	39 Warn			143	99	0	0	0	8/22/25, 4:18 PM

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	90	0.00

Resource	Utilization	Available	Utilization %
LUT	153	20800	0.74
FF	127	41600	0.31
IO	40	106	37.74



Conclusion

In this assignment, we successfully designed and implemented a Multiply-Accumulate (MAC) unit on the Basys3 FPGA board. The design incorporated input capture, debouncing, rising edge detection, and overflow handling to ensure reliable operation. The accumulator output was displayed on LEDs, while reset and overflow conditions were clearly indicated on the 7-segment display. Through this exercise, we gained practical experience in combining arithmetic operations with input conditioning and user feedback mechanisms in digital system design.