COL215 – Digital Logic and System Design
Department of Computer Science & Engineering, IIT Delhi
Semester I, 2025-26
Lab Assignment 4

MAC Unit

1 Introduction

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

$$a = a + (b \times c)$$

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 will get familiar with:

- Resetting a design to a known *initial state*.
- 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.
- Overflow detection in arithmetic operations.

2 Problem Description

2.1 Capturing the MAC inputs

The MAC unit has two 8-bit inputs b and c, provided through the slider switches. Slider switch SW10/SW11 is used to indicate that the 8-bit data SW7-SW0 represents b/c. See table below. The BTNC pin indicates a *reset* of the MAC unit.

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

Reuse your knowledge about capturing input data into registers, based on some event, from Assignment 3.

2.2 MAC Unit

The MAC unit comprises an 8-bit multiplier and a 16-bit accumulator.

2.2.1 Multiplier

A multiplier is a combinational circuit that performs multiplication of the operands. For this assignment, you can use the behavioral implementation for an 8-bit multiplier.

2.2.2 16-bit Accumulator

The MAC unit includes a 16-bit accumulator. When the slider switch SW12 toggles, the accumulator gets enabled and performs one MAC operation. The initial value stored in the accumulator should be parameterized with the default parameter value as zero. This parameter should be passed through the testbench.

The accumulated value should be indicated over the 16 LEDs.

Notice that the slider switch, once turned ON, will remain ON for multiple clock cycles. In order to avoid accumulator accumulating the same value multiple times, the logic should ensure that only one toggling of the slider switch is used for accumulation. One possible way to do this is by detecting the rising edge of the slider switch and performing the MAC when the rising edge is detected. A sample procedural block to do this is shown below:

```
always @(posedge clk) begin
        sw_12_del <= sw_12;    // delaying the sw_12 by a clock cycle
end

// Detecting rising edge of sw_12
assign sw_12_on_1_pulse = sw_12 & ~sw_12_del;
```

2.2.3 Reset to Initial Values

The system should run on the internal Basys3 clock.

The accumulator needs to be *reset* to a known initial value. Use the BTNC pin as the reset signal.

In Verilog, resetting a signal can be done by assigning it within a block conditionally, based on the reset signal. An example with an *active high reset (reset when HIGH)* is shown below:

```
always @(posedge clk) begin
  if (rst) begin
    data_reg <= 0;            // rst clears the data_reg
  end else begin
    data_reg <= data_in;      // Normal operation
  end
end
```

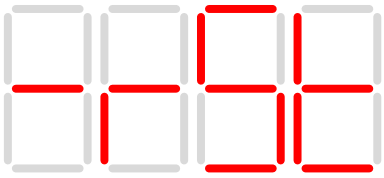

Figure 1: Display in case of reset event

We need to have a way to indicate that a reset event has occurred. To do this, you can use the four 7-segment display on the Basys3 board. When the BTNC pin is pressed, indicate '-rSt' on the display, as shown in Figure 1. The indication should be visible for 5 seconds only.

2.2.4 Debouncing

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 2.

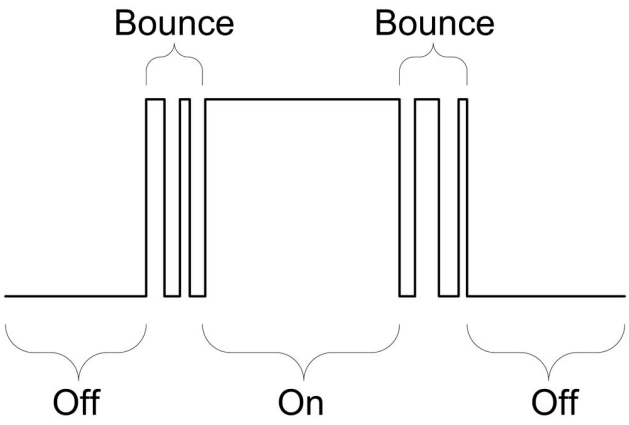

Figure 2: Glitched when operating push button

Your task is to design a module to remove the glitches and output only the final stable state. One way to remove the glitches is through a counter that could be used to wait for a small duration; once the counter overflows, then we assign the input signal to the output port.

The pin bouncing and de-bouncing logic should work on the expected lines in the simulation. The testbench should model the pin bouncing with varying scenarios. The following conditions should be covered. (See Figure 2. The figure indicates 2 bounces each when the push button transitions to ON and OFF.)

- 1 bounce on switching ON and 2 bounces on switching OFF. Push button ON duration: 30ns. ON duration of bouncing glitches: 5ns. OFF duration of bouncing glitches: 2ns.
- 2 bounces on switching ON and 2 bounces on switching OFF. Push button ON duration: 30ns. ON duration of bouncing glitches: 2ns. OFF duration of bouncing glitches: 5ns.
- 2 bounces on switching ON and 2 bounces on switching OFF. Push button ON duration: 40ns. ON duration of bouncing glitches: 5ns. OFF duration of bouncing glitches: 5ns.
- 3 bounces on switching ON and 3 bounces on switching OFF. Push button ON duration: 40ns. ON duration of bouncing glitches: 2ns. OFF duration of bouncing glitches: 2ns.

2.2.5 Accumulator overflow

It might happen that the accumulator overflows due to a sequence of MAC operations. We would like to detect this condition and indicate it on the Basys3 board via the 7-segment display.

In your design, detect the accumulator overflow condition and indicate 'OFLO' when detected, as shown in Figure 3. 'OFLO' should remain displayed and get cleared once the BTNC pin is pressed. Clear the accumulator (reset to zero) upon overflow. In case of no overflow, there should be no indication on the display and it should remain inactive.

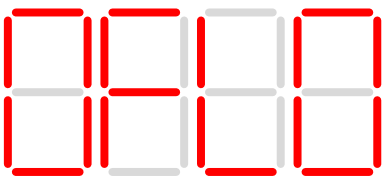

Figure 3: Display in case of accumulator overflow

3 Submission and Demo Instructions

1. Demo should be given in the assigned lab slot itself.
2. You are required to submit the following on Gradescope:
   - 20 points: Verilog files for all the designed modules.
   - 20 points: Verilog files for the testbench and the simulation of reset, overflow and different pin-bouncing conditions.

- 10 points: Warning-clean constraint file (.xdc), Bit file.
- 30 points: Questionnaire corresponding to the assignment.
- 20 points: A short report (2-3 pages) outlining simulation snapshots, synthesis report, and generated schematics. Explain your design decisions. Having only snippets in the report (without explanation) will result in penalty.

We advise you to be ready with your design before the lab session, and during the session, perform validation by downloading it into the FPGA board.

4 Resources references

- IEEE document: `https://ieeexplore.ieee.org/document/1620780`
- Basys 3 board reference manual: `https://digilent.com/reference/_media/basys3:basys3_rm.pdf`
- Online Verilog simulator: `https://www.edaplayground.com`