# COL215 – Digital Logic and System Design
# Department of Computer Science & Engineering, IIT Delhi
# Semester I, 2025-26
# Lab Assignment 5

Vector Dot Product

## 1 Introduction

In this assignment, we will implement the dot product of vectors reusing the Multiply-Accumulate (MAC) unit from the last assignment.

$$\text{Dot Product} = \sum_{i=0}^{n} A[i] \times B[i].$$

where:
$A[i]$: $i^{\text{th}}$ element of vector $A$
$B[i]$: $i^{\text{th}}$ element of vector $B$

Design a Dot Product Controller reusing the MAC designed from assignment 4 to compute the Dot Product and print the results on the 7-segment display and the LEDs.

## 2 Problem Description

### 2.1 Switch and pin functions

For the dot product computation we have two input vectors, $A$ and $B$, provided through the slider switches, SW7–SW0. Both vectors consist of four positive elements (8-bit integers). Slider switches SW8 and SW9 are used to select the element. SW12–SW15 are used to select the read and write functionalities. See table below. The BTNC pin indicates a *reset* of the system.

| **Pin** | **Function** |
|---|---|
| SW7–SW0 | Will be used to input numbers. |
| SW9–SW8 | 2-bit index to indicate the position of the element to be entered. <br>• 00: Element at index 0 <br>• 01: Element at index 1 <br>• 10: Element at index 2 <br>• 11: Element at index 3 |
| SW12 | Write enable for vector A. |
| SW13 | Write enable for vector B. |
| SW14 | Read enable for vector A. |
| SW15 | Read enable for vector B. |
| BTNC | Resets everything and displays '-rSt' for 5s. |

## Notes

- When read enable is turned ON, the value for that element should be visible on the 7-segment displays 0 and 1. The other two should remain inactive. See Figure 1.
- When both write enables are ON, the same data should get recorded for both the vectors' elements.
- When both read enables are ON, the result of the dot product of the two vectors should be visible on the displays.
- If the resulting dot product value cannot fit within 16 bits, the display should indicate 'OFLO'.

- The LEDs should display the 16 LSBs of the resulting dot product (irrespective of read enables).

For example, a setting with BTNC not pushed and SW15–SW0 as 'OFF, OFF, ON, ON, OFF, OFF, ON, ON, OFF, OFF, ON, ON, ON, ON, ON, ON' should sample 3F as the $4^{th}$ element of both the vectors, i.e., index 3 where indices start from 0.
Switching SW15 as ON should result in 3F being displayed on the four 7-segment display as in Figure 1.

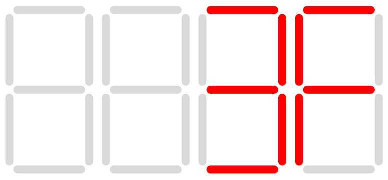
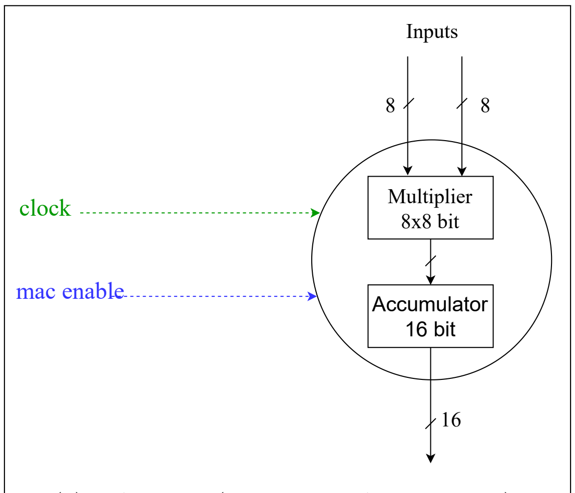


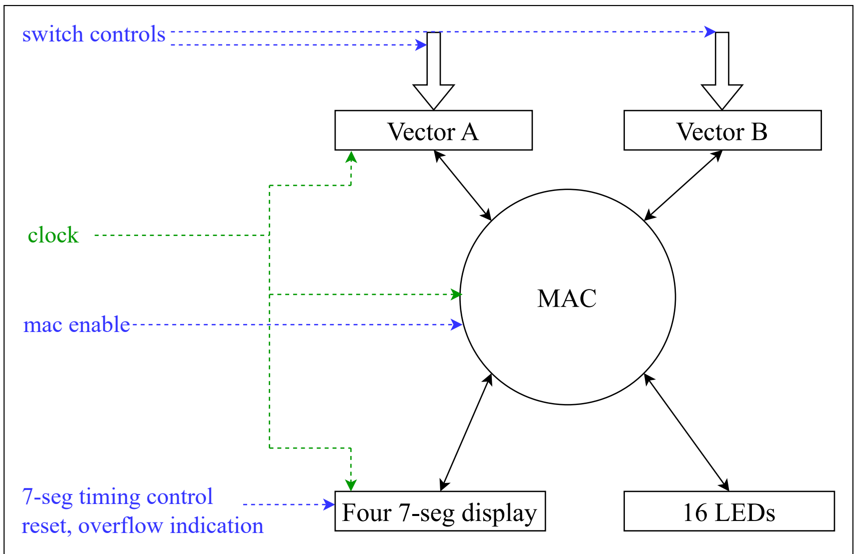
Figure 1: Example display

Further switching SW14 as ON should display the final dot product on the four 7-segment display. The LEDs keep displaying the 16 LSB of the result.

## 2.2 Controller Unit for Dot Product System



(a) MAC Unit (Designed in Assignment 4)



(b) Controller Unit (To be designed in Assignment 5)

Figure 2: (a) Multiplier-Accumulate Unit and (b) Dot Product Controller

We will design a *controller* to implement the dot product, see Figure 2b. The controller will steer the dot computation, using the MAC unit depicted in Figure 2a. **You must instantiate the MAC unit from**

**Assignment 4.**

One approach to building the controller could be:

- Detect writes on the elements of the vectors A and B.
- Once all the elements are written, initiate a counter (e.g., an up-counter from 0 to 'number of elements+1') – 0 to 4 in this case.
- Based on the counter value, keep updating the MAC input elements and accumulating the result.

You can use the switch values to decide whether an element is written or not. An example to do so is listed in the example below:

```
always @(posedge clk) begin
  if (sw12 & sw9 & sw8) begin
    A_written[3] <= 1;            // rst clears the data_reg
  end
end
```

Once all the bits of A and B are written, you can start the dot product computation.
The final mac enable signal can be:

```
assign mac_enable = initiate_mac & ~dot_product_computed;
```

where dot_product_computed is an indication which indicates the completion of dot product. The inputs to the MAC can be expressed as:

```
assign mac_input_b = counter>3 ? 8'b0 : A[counter];
assign mac_input_c = counter>3 ? 8'b0 : B[counter];
```

where counter varies from $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. When the counter reaches 4, the signal dot_product_computed goes high. The counter should be an up-counter which starts from 0 and ends at 4.

```
    always @(posedge clk) begin
        if (initiate_mac & counter < 4) begin
            counter <= counter + 1;
        end
    end
assign dot_product_computed = (counter > 3);
```

Once after dot_product_computed goes high, print the result on the 7-segment display and the LEDs.

## 2.3 Four 7-segment display

Use your knowledge about the 7-segment display from previoius assignments for this assignment. The seven-segment display can be used to display numbers in hexadecimal format as in Figure 3

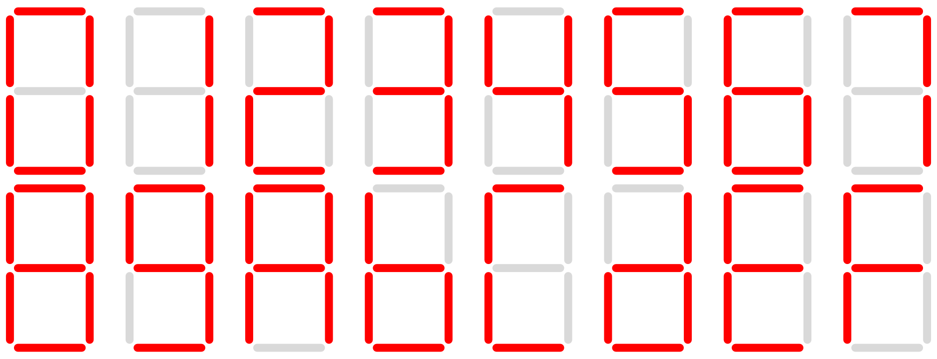


Figure 3: Displaying hexadecimal numbers on 7-segment display

## 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:
   - 30 points: Verilog files for all the designed modules. All the blocks within always @(posedge clk) block should get reset when BTNC gets pushed.
   - 10 points: Verilog file of the testbench and simulation waveforms indicating reset, overflow and normal operations.
   - 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 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`