COL215
Digital Logic & System Design

Assignment 4

Submitted by:

Dhairya Kuchhal	(2024CS50396)
Siddhant Agrawal	(2024CS50469)

September 11, 2025

1 Task

The objective of this lab was to design, simulate, and implement Verilog files for a Dot Product Unit re-using the Multiply Accumulator Unit from last Lab. We take in input using SW[0:7], write it onto two 4-dimensional vectors A and B through SW[12:13] choosing index through SW[8:9]. Individually turning on SW[14:15] enables reading any index's value from the 7-segment display board. The accumulated value (even in cases of overflow) is displayed on the LEDs.

2 Design Decisions

2.1 System Architecture and Modularity

A modular design approach was adopted from the outset to enhance clarity, simplify debugging, and promote code reusability. The system was partitioned into several specialized auxiliary modules, each with a single, well-defined responsibility. These modules are instantiated and interconnected within the top-level `Dot_product_top` module. This strategy allowed for each component to be developed and tested independently before being integrated into the final design.

Module Name	**Function**
`debouncer`	Stabilises the physical reset button signal to prevent spurious resets.
`capture_digits`	Manages writing data from the switches into the vector registers.
`en_detector`	Converts a sustained switch signal into a single-cycle start pulse.
`mac_unit`	Performs the core combinational multiply-accumulate operation.
`dot_product_controller`	An FSM that orchestrates the sequential dot product calculation.
`display`	Manages the multiplexed 7-segment display for all UI feedback.

Table 1: Auxiliary Modules and Their Functions

2.2 Core Logic and Data Handling

2.2.1 Vector Input and Storage

The mechanism for user input was designed to directly map to the physical switches on the FPGA board, providing an intuitive interface. The `capture_digits` module implements this logic:

1. **Data Input:** The lower eight switches, `SW[7:0]`, are used to specify the 8-bit magnitude of a vector element.
2. **Index Selection:** Switches `SW[9:8]` act as a 2-bit binary selector, allowing the user to target any of the four element positions (0 to 3) within a vector.
3. **Vector Selection:** The state of switches `SW[12]` and `SW[13]` serve as write-enable signals for Vector A and Vector B, respectively. The module updates the appropriate internal register only when one of these switches is active.

2.2.2 Overflow Detection

To reliably detect when the dot product exceeds the 16-bit display capacity, a 17-bit register was used for the accumulator within the `dot_product_controller`. The multiplication of two 8-bit numbers can result in a 16-bit product. Summing four such products can lead to a value requiring more than 16 bits. By using a 17-bit accumulator, the most significant bit (MSB) naturally functions as an overflow flag. When the final calculation is complete, this 17-bit value is split: the MSB is assigned to the 1-bit `overflow` signal, and the lower 16 bits are assigned to the `result_out` signal. This provides a simple and hardware-efficient method for overflow checking.

2.3 Input Conditioning and User Interface

2.3.1 Debouncer for Reset Signal

Physical buttons are susceptible to mechanical bouncing, which can cause multiple rapid transitions and trigger a reset multiple times. The `debouncer` module was implemented to filter this noise. It uses an internal counter that starts when the button is pressed. The module waits for the input signal from `BTNC` to remain stable for a predetermined number of clock cycles (approximately 10ms). Only after this period of stability is the clean, internal `rst` signal asserted, ensuring the system undergoes a single, reliable reset sequence.

2.3.2 Rising Edge Detector for Calculation Trigger

To ensure the dot product calculation runs exactly once per user action, the `en_detector` module was implemented. Rather than having the calculation run continuously while the trigger switches (`SW[14]` and `SW[15]`) are held high, this module detects only the moment they transition from low to high (a rising edge). It produces a single-cycle `start` pulse, which is sufficient to initiate the FSM in the `dot_product_controller`. This prevents unwanted re-calculations and ensures predictable system behavior.

2.3.3 Display Logic

The `display` module handles all user-facing visual feedback. Its most important feature is the timed display for the reset message. When the `rst` signal is active, a dedicated 29-bit counter, `rst_timer`, is enabled. The duration for this counter was calculated based on the Basys 3 board's 100MHz internal clock frequency (5 seconds $\times$ $100,000,000$ Hz $=$ $500,000,000$ cycles). This ensures the "-rSt" message is displayed for precisely five seconds. The underlying 7-segment display structure, which multiplexes the four digits at a high frequency, was reused from previous lab assignments to provide a stable, flicker-free visual output.

2.4 Simulation on Testbench, and Snapshots

We wrote the testbench module (tb_main.v) for simulation of all three situstions: reset, Dot Product under normal conditions, and Dot product leading to overflow

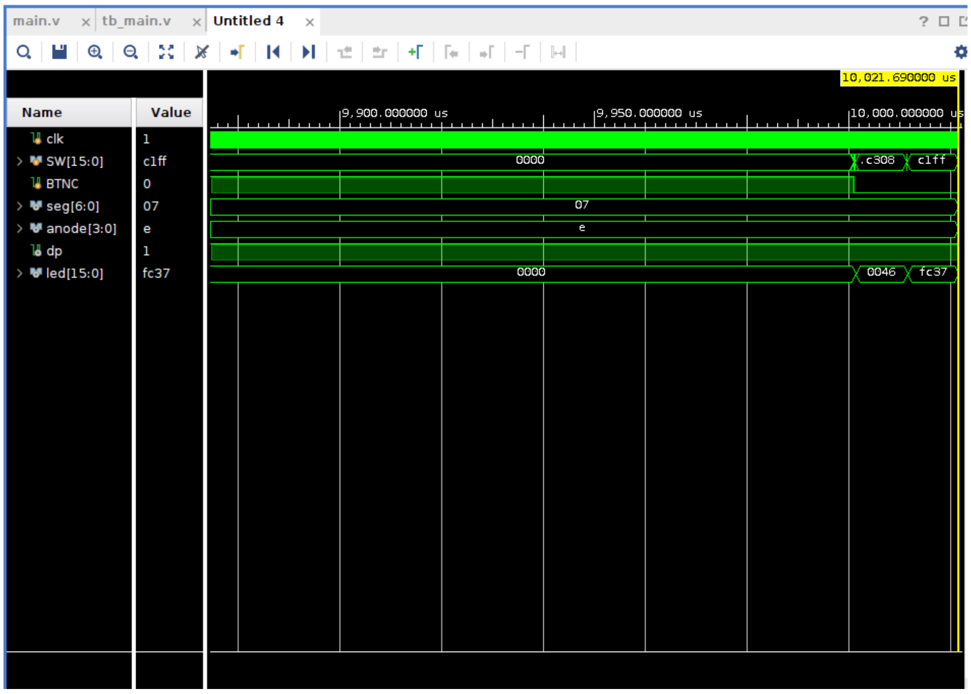


Figure 1: Complete Simulation

1. this image shows the overall simulation. the leftmost extended section demostrates 5 seconds of reset. then we have a shorter section displaying multiplication of Vectors A and B. then we have the third ending sections displaying dot products of vectors with maxed out values leading to overflow.

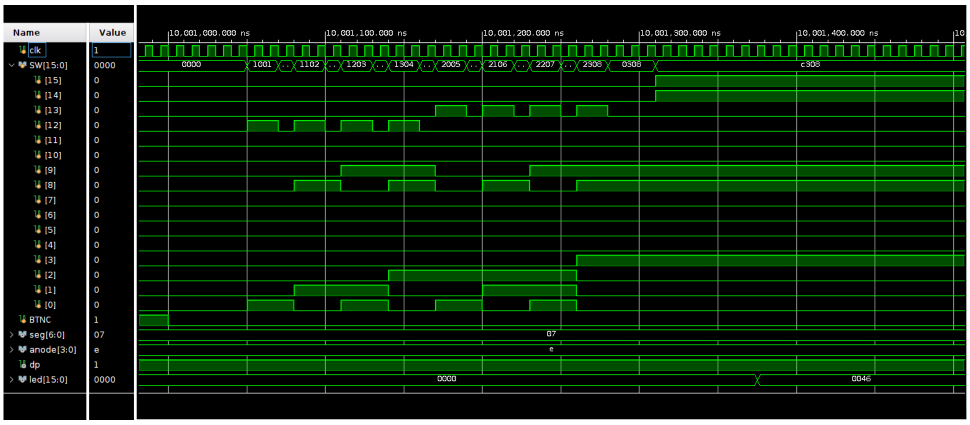


Figure 2: Calculation

2. This is the the detailed image of part 2 (normal caclulation) where we set Vector A = [1,2,3,4] and Vector B = [5,6,7,8]. Dot Product yields 70, which is then visible in the LED row as 0046 (hexadecimal)

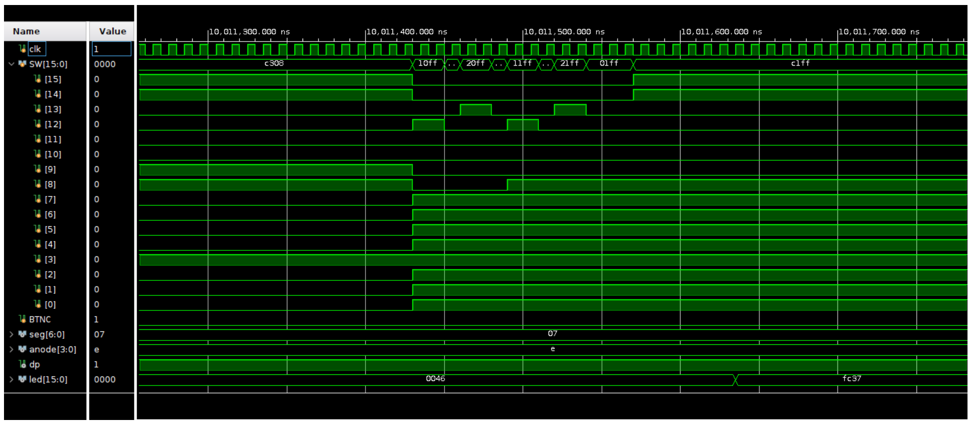
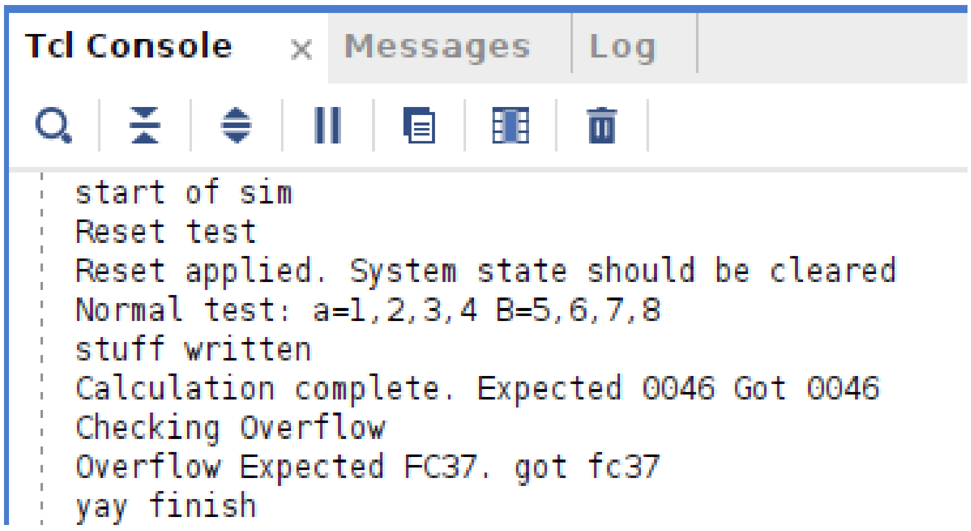


Figure 3: Overflow

3. we max out the value to be uploaded (to 2$\hat{8}$ - 1) and assign it to the zeroeth and first index of A and B. The Dot product overflows, and last 16 digits (fc37) are displayed on LED



```
start of sim
Reset test
Reset applied. System state should be cleared
Normal test: a=1,2,3,4 B=5,6,7,8
stuff written
Calculation complete. Expected 0046 Got 0046
Checking Overflow
Overflow Expected FC37. got fc37
yay finish
```

Figure 4: Complete Simulation

4. shows the printed output displayed values during simulation

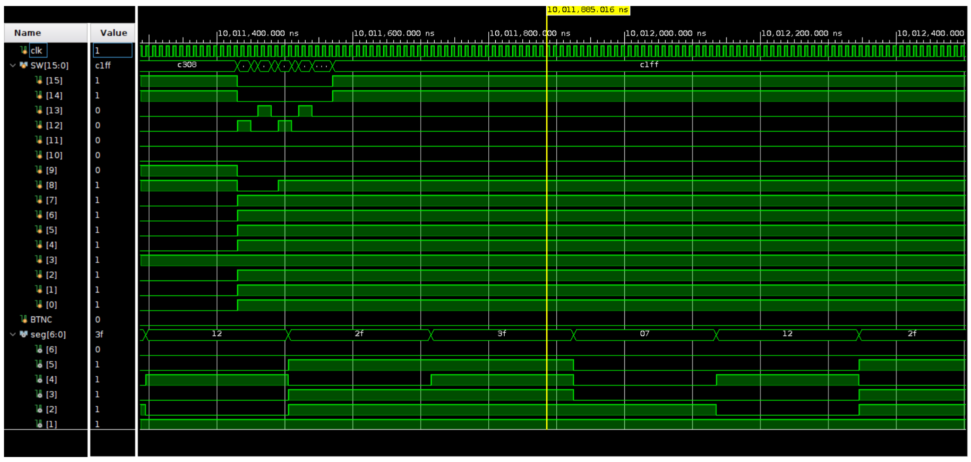
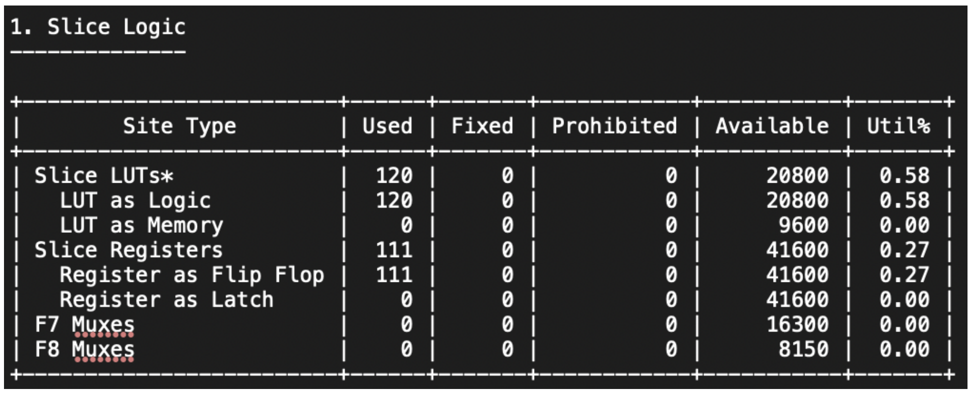


Figure 5: Cycling display

5. We note that variations in seg and anode were not visible in the above simulations. this is because our clock is of 1ns duration, while the simulation in order to represent 5s long delays has a timestep of 1ms which overshadows 1ns changes. by parametricising the "refresh rate" in main module and setting it to 1ms we can view the variation.

3 Synthesis Report

The following tables show the main resource counts. Other details are present in the synthesis report.



```
1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |  120 |     0 |          0 |     20800 |  0.58 |
|   LUT as Logic          |  120 |     0 |          0 |     20800 |  0.58 |
|   LUT as Memory         |    0 |     0 |          0 |      9600 |  0.00 |
| Slice Registers         |  111 |     0 |          0 |     41600 |  0.27 |
|   Register as Flip Flop |  111 |     0 |          0 |     41600 |  0.27 |
|   Register as Latch     |    0 |     0 |          0 |     41600 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |      8150 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
```

Figure 6: Flip-Flops and LUT

4 Generated Schematics

This makes us see beyond the code and into the actual fpga boards wiring between the various combinational gates and ports

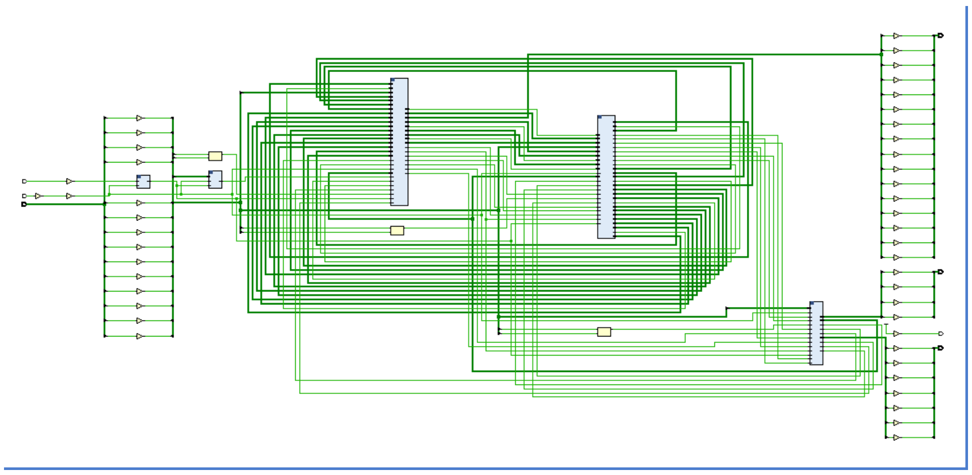


Figure 7: Generated Schematic design