COL215
Digital Logic & System Design

Assignment 4

Submitted by:
Dhairya Kuchhal (2024CS50396)
Siddhant Agrawal (2024CS50469)

September 25, 2025

1 Task

The objective of this lab was to design Verilog files for a Dot Product machine using existing memory blocks (RAM And ROM). We also integrated functionality for reset, increment and display. ROM is initialised to random 4-bit values from given .coe file and represent Vector A. RAM0 represents the second vector B, with initial random 4-bit values which can be updated. Their sum is stored in RAM1, representing Vector C of 5-bit values.

2 Design Decisions

2.1 System Architecture

A modular design approach was adopted from the outset to enhance clarity, simplify debugging, and promote code reusability. The followign auxilliary modules are instantiated and interconnected within the top-level `vector_adder_top` module. This strategy allowed for each component to be developed and tested independently.

Module Name	Function
`memory_controller`	Manages calculations and reading writing data to RAM and ROM modules
`debouncer`	Stabilises the physical reset button signal to prevent spurious resets.
`rising_edge`	Converts a sustained switch signal into a single-cycle start pulse.
`display`	Manages the 7-segment display for all User feedback and verification.
`RAM`	Random Access Memory from the IP Catalog. Supports Read/ Write
`ROM`	Read Only Memory Imported from IP Catalog. Supports Read only

Table 1: Auxiliary Modules and Their Functions

2.2 Input and Functionality from FPGA Board

The following table maps the functionality required in our task to corresponding switches on the Basys3 FPGA board.

Switch	Function
`SW3 - SW0`	Used to provide four bit input to B[i]
`SW13-SW4`	Index of vector B where value is entered/read
`SW15-SW14`	2-bit index to select element to be displayed. 00: Inactive 01: Displays hexadecimal representation of vectors in order <RAM1><RAM0><ROM> 10: Write on RAM0 11: Increment B[i] by 1
`BTNC`	reset values

Table 2: Input Description

2.3 Simulation on Testbench, and Snapshots

Our testbench module (tb_main.v) demonstrated functionality through 4 sequential tests on 1. Reset Functionality 2. Initial Calculations 3. Writing and Updating Values and 4. Incrementing
On account of finished disk storage space, we have taken photos instead of screenshots. All details are clearly visible.

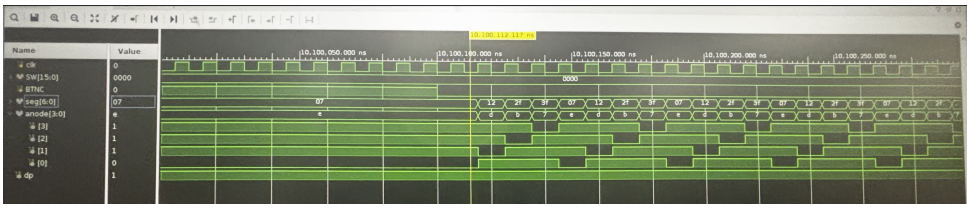

Figure 1: TEST 1: Reset

1. The left part of this image shows us pressing the BTNC button. Shortly after leaving BTNC and letting the signal stabilise, the screen phases to display "-rst"

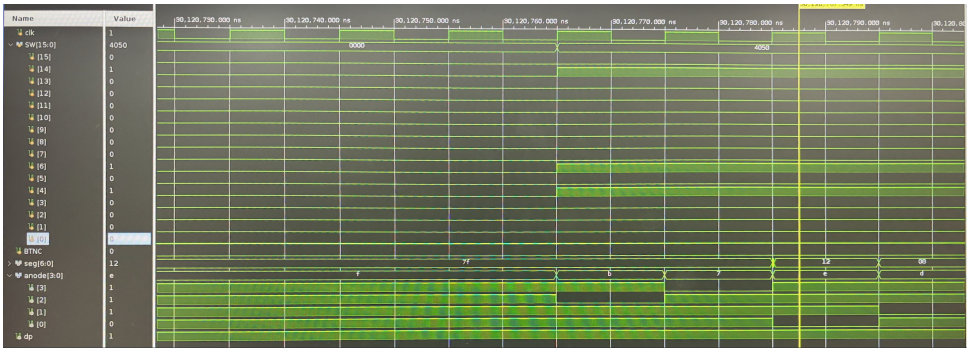

Figure 2: TEST 2: Reading Value at Index 5

2. This image shows us set the SW configuration to Read Mode, Index 5. The displayed value (seg, anode) oscillates to show all 3 values (A,B,C)

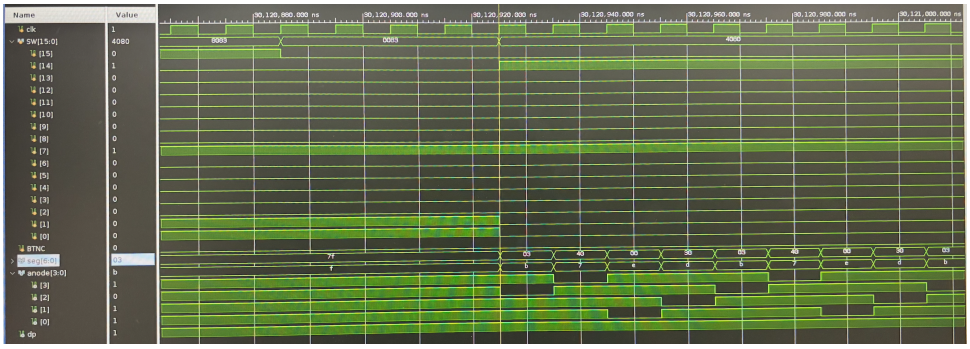

Figure 3: TEST 3: Writing at RAM0, then Reading

3. In the leftmost section, we set SW configuration to Write, 8th index, value 3. then we transition to inactive state and then read the values at index 8. We find a changed value in RAM0 and accordingly calculated RAM1 in seg

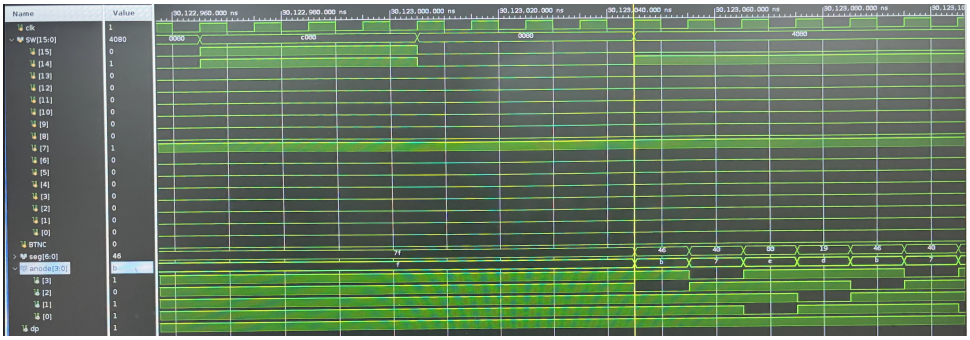

Figure 4: TEST4: Increment Operation

4. We set SW configuration to Increment (11), Index 8. We then read the values.

3 Utilisation Data and Generated Schematics

The following tables show the required utilisation data (BRAMs, DSPs, LUTs and FFs) followed by an analysis of the recorded values.

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	89	0	0	20800	0.43
LUT as Logic	9	0	0	20800	0.04
LUT as Memory	80	0	0	9600	0.83
LUT as Distributed RAM	80	0			
LUT as Shift Register	0	0			
Slice Registers	5	0	0	41600	0.01
Register as Flip Flop	5	0	0	41600	0.01
Register as Latch	0	0	0	41600	0.00
F7 Muxes	40	0	0	16300	0.25
F8 Muxes	20	0	0	8150	0.25

Figure 5: Flip-Flops and LUT

1. LUT's (Look Up tables) are FPGA's core logic elements. They Handle Both Memory (80 LUT's used) and the control and addition logic (9 LUT's)

2. Flip-FLops or FFs are the basic storage elements used to hold values across clock cycles. Since much of the logic was combinational, only 5 FF's were sufficient for this assignment

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	50	0.00
RAMB36/FIFO*	0	0	0	50	0.00
RAMB18	0	0	0	100	0.00

Figure 6: Block RAM: Not utilised

BRAM's are on-chip memory blocks for efficient storage of large arrays. Our distributed memory generator mapped ROM/RAM into LUTs given their small size: hence we didn't use BRAM.

3. DSP

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

Figure 7: Generated Schematic design from Synthesis section

DSP's are meant for high speed multiplication and accumulation. Given that we only do addition of 4-bit integers, they are not called upon.

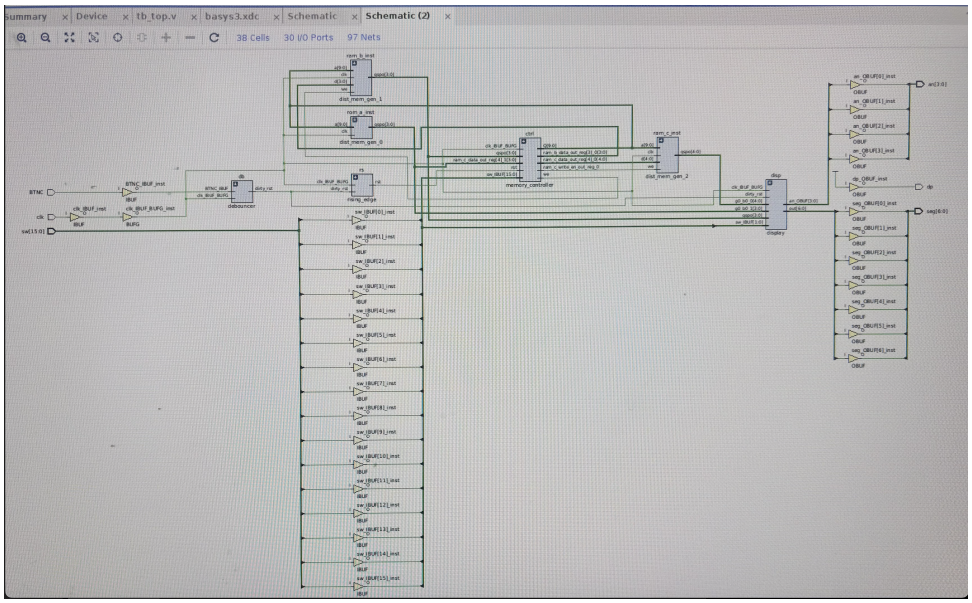

Figure 8: Generated Schematic design from Synthesis section

The above schematic shows the on-board implementation of out design