COL215 – Digital Logic and System Design

Department of Computer Science & Engineering, IIT Delhi

Semester I, 2025–26

Lab Assignment 6 (Friday)

Memory Read and Write

Developed By:
Bharmal Bhakar (2024CS10403)
Ritik Raj (2024CS10086)

1 Introduction

- In this assignment, we will be working with three memory blocks: a Read-Only Memory (ROM) and two Random-Access Memories (RAMs). The core task is to create a control unit that handles reading from and writing to these memory blocks based on user inputs.
  Our task is to design a module that performs vector addition of two given input vectors, defined as follows.
  C[i] = A[i] + B[i], $0 \leq i \leq 1023$
  Where,
  A: Input vector of 1024 4-bit elements stored as ROM
  B: Input vector of 1024 4-bit elements stored as RAM0
  C: Output vector of 1024 5-bit elements stored as RAM1

  Along with addition, an increment operation should also be supported over input vector elements. Also give a proper indication on the four 7-segment display in case of reset condition, and display the content when read.

Design Decisions

The design follows a hierarchical structure. The top module acts as the main controller, instantiating three memory IP cores (vector_A_rom, vector_B_ram0, vector_C_ram1) and the seven_seg_display module. This modularity separates the core control logic from specialized functions like memory storage and display driving, making the design easier to manage and debug.

2.1. Top-Level Module (`top.v`)

The `top` module serves as the central controller. It interfaces with the physical I/O of the Basys3 board, including the clock, switches (`sw`), and the center button (`BTNC`).

Key Features :

- The core of the controller is a sequential always @(posedge clk) block that manages the system's state based on the mode input from switches sw[15:14]:
  00: Inactive display. **01**: Read mode. The values from ROM (A[i]), RAM0 (B[i]), and

RAM1 (C[i]) at the address specified by `sw[13:4]` are displayed. **10**: Write mode. The value on `sw[3:0]` is written to RAM0 at the specified address. Simultaneously, the sum is calculated and written to RAM1. **11**: Increment mode The value in RAM at the specified address is incremented by one, and the corresponding value in RAM1 is updated.

- `sw[13:4]` provides the 10-bit address `i` for the memory blocks, and `sw[3:0]` provides the 4-bit data input for write operations.
- The BTNC button is used for a system-wide reset. A debouncing circuit with a 20-bit counter(debounce_counter). The debounced_reset signal only changes state after the BTNC input has been stable for approximately 10ms, providing a clean, glitch-free internal reset signal.
- A separate 30-bit counter (counter) is triggered by the debounced_reset. This holds the system in a visible reset state for a longer duration (approximately 5 seconds), allowing the "-rSt" message to be clearly displayed on the 7-segment display as required.
- write and increment operations happen only once when the mode switches are changed, not continuously on every clock cycle. This was achieved by creating a simple edge-detection mechanism. A register mode_prev stores the mode from the previous clock cycle. A write or increment is triggered only if the current mode is 'write' or 'increment' AND the mode_prev was different. This ensures the action is a single-shot event, executing only on the transition into that mode.

2.2. Memory Modules

1. vector_A_rom (ROM): A 1024 x 4-bit ROM to store vector A. It was initialized using the rom_vector.coe file**.**
2. vector_B_ram0 (RAM): A 1024 x 4-bit single-port RAM to store vector B. It was initialized using the ram_vector.coe file.
3. vector_C_ram1 (RAM): A 1024 x 5-bit single-port RAM to store the output vector C.

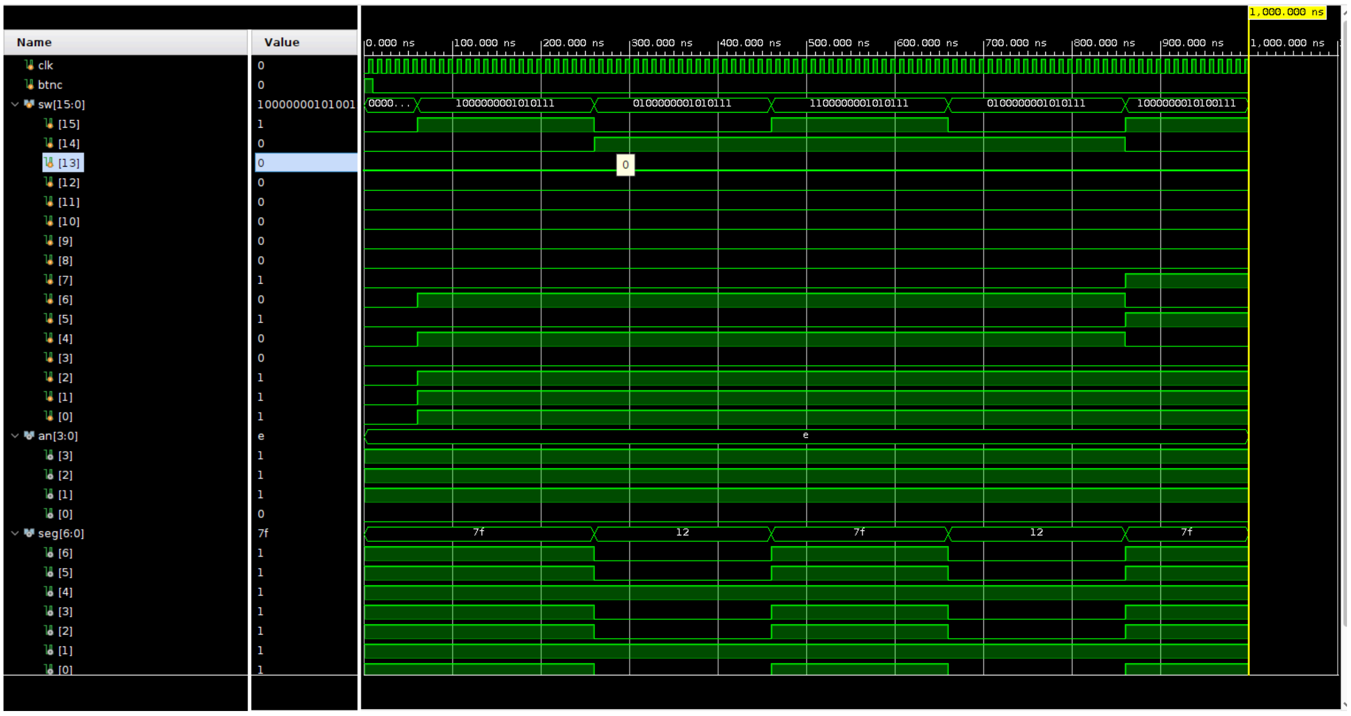
These IP cores were instantiated within the top.v module

2.3. Seven-Segment Display Module (`ss_display.v`)

The seven_seg_display module from previous labs was reused. It takes four 7-bit segment patterns as input and drives the four anodes of the 7-segment display. An 18-bit counter is used to cycle through the anodes at a high frequency (~190 Hz), creating the illusion that all four digits are lit simultaneously without any perceptible flicker.

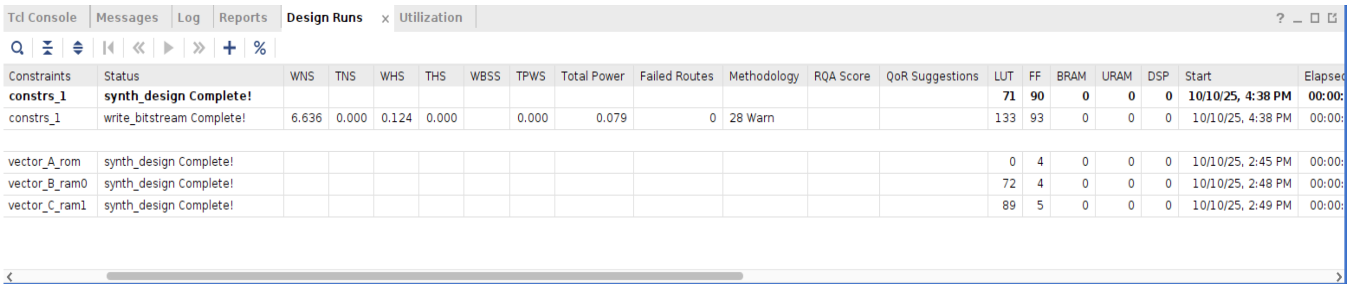
3. Simulation

The simulation confirmed the correct behavior of the design.
Reset: Upon reset, the system correctly displayed the "-rSt" message.
Write: When writing 7 to address 5, the simulation showed ram0_we and ram1_we being asserted for a single clock cycle.
Read: In read mode, the outputs correctly displayed the values from the ROM, the newly written value in RAM0, and the calculated sum in RAM1.
Increment: The increment operation correctly updated the value in RAM0 from 7 to 8, and the value in RAM1 was updated accordingly.



4. Resource utilization data

The following table summarizes the utilization for the complete design and the individual memory IP cores.



Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elapsed
constrs_1	**synth_design Complete!**												**71**	**90**	**0**	**0**	**0**	**10/10/25, 4:38 PM**	**00:00:**
constrs_1	write_bitstream Complete!	6.636	0.000	0.124	0.000		0.000	0.079	0	28 Warn			133	93	0	0	0	10/10/25, 4:38 PM	00:00:
vector_A_rom	synth_design Complete!												0	4	0	0	0	10/10/25, 2:45 PM	00:00:
vector_B_ram0	synth_design Complete!												72	4	0	0	0	10/10/25, 2:48 PM	00:00:
vector_C_ram1	synth_design Complete!												89	5	0	0	0	10/10/25, 2:49 PM	00:00:

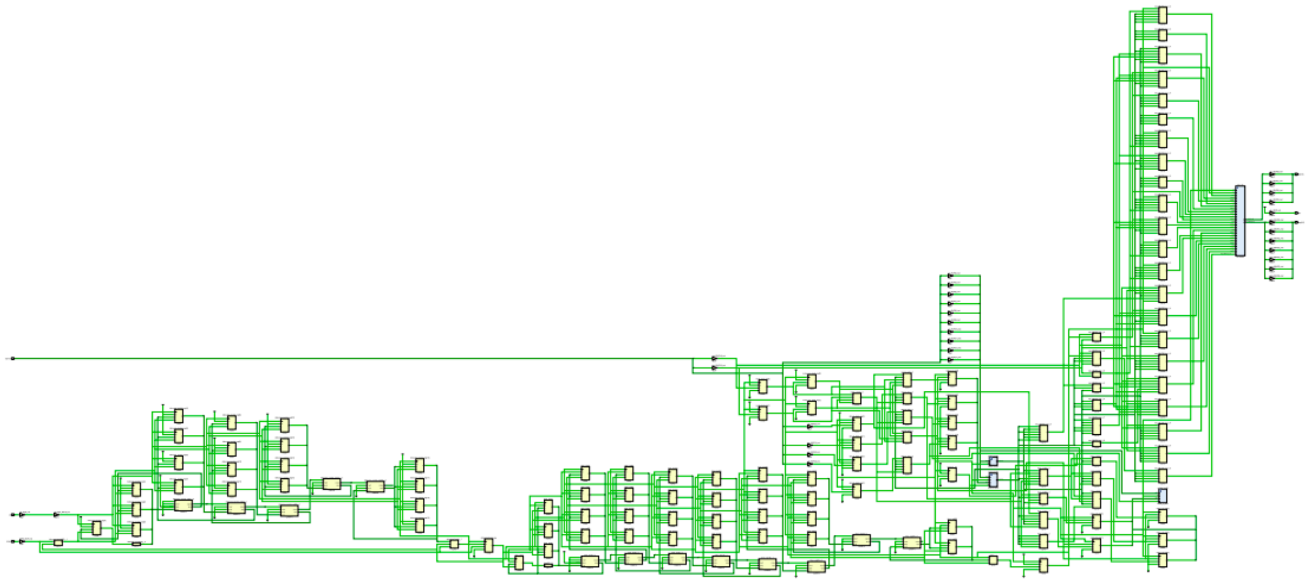
- **LUTs:** The final design uses **133 LUTs**. A significant portion of these is used to implement the two RAM blocks as "Distributed RAM," meaning the memory is built from general-purpose LUTs instead of dedicated memory blocks. The control logic, reset debouncer, 7-segment decoders, and the adder also contribute to this count. The LUT count increased from 71 after synthesis to 133 after implementation, which is expected as the place-and-route tool performs optimizations and logic replication to meet timing constraints.
- **Flip-Flops (FFs):** These are used for all sequential logic (registers). The design uses **93 FFs**. These are primarily for the counters (debounce_counter, timer, and the counter in ss_display), state registers (mode_prev, reset), and importantly, the registered outputs of the three memory blocks. vector_A_rom and vector_B_ram0 each have 4 FFs for their 4-bit outputs, and vector_C_ram1 has 5 FFs for its 5-bit output, as specified in the IP generator settings.
- **Block RAM (BRAM):** The design uses **0 BRAM** blocks. This is a critical design observation. Although we implemented memory, the "Distributed Memory Generator" IP was chosen. This instructs Vivado to use LUTs to create the memory, which is suitable for smaller memory blocks. If we had chosen "Block Memory Generator," Vivado would have used the dedicated, more efficient BRAM resources

on the FPGA. For this lab's memory size, distributed RAM is an acceptable implementation choice.

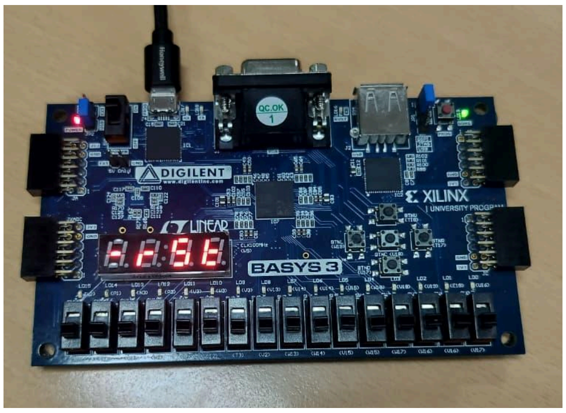
- **DSP Slices:** The design uses **0 DSP** slices. The addition operation (C[i] = A[i] + B[i]) is simple enough to be implemented efficiently using LUTs. Dedicated DSP slices are typically reserved for more complex arithmetic operations like multiplication.

5. Generated Schematics

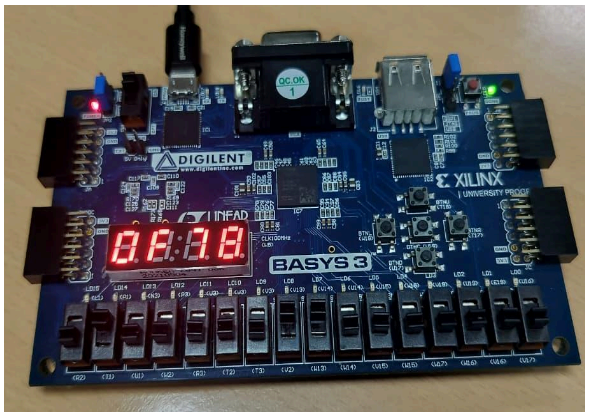
The RTL schematic provides a visual representation of the synthesized hardware. The inputs (sw, BTNC, clk) are shown on the left, feeding into the central control and memory logic. The three instantiated memory blocks are visible as regular structures. The logic on the right side of the schematic handles the multiplexing and decoding for the 7-segment display outputs (seg and an). The schematic confirms the dataflow described in the Verilog, with the shared address bus and the path for the adder output feeding into the third RAM.



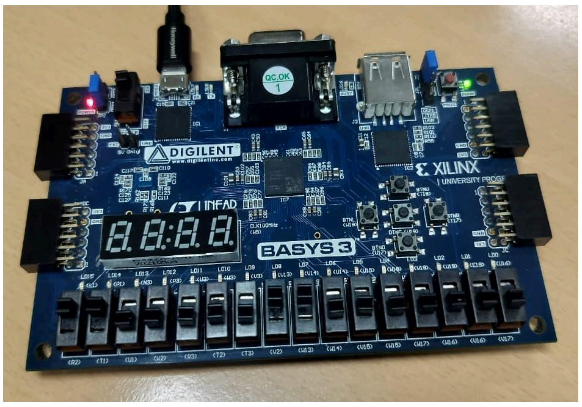
6. On-Board Implementation

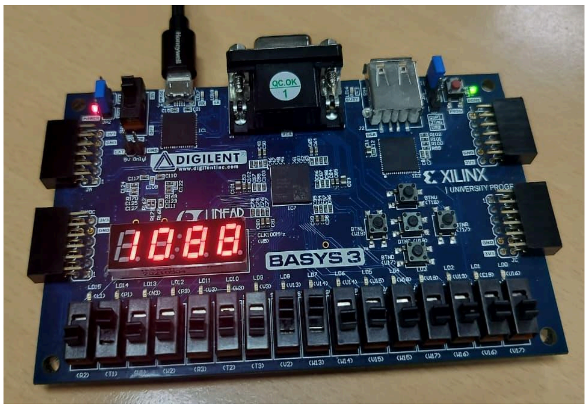


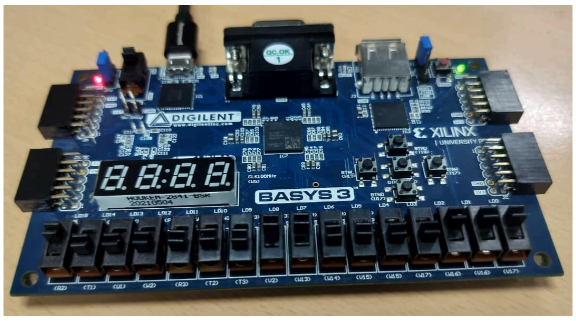
“-rst” is displayed when reset button is pressed

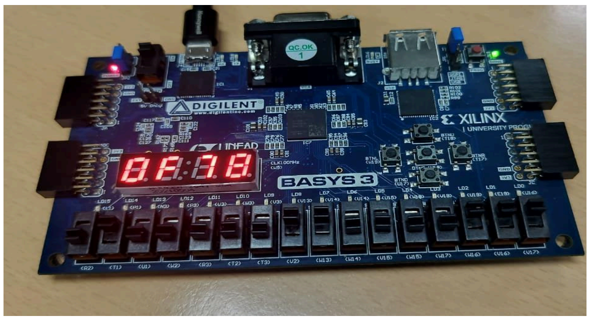


when sw[15:14] = 01(read) then rom(an0) ram0(an1), ram1(an2 & an3) is shown


When sw[15:14] = 11(increment) Value of ram0 at index sw[13:4]will increased by 1 and so the sum


incremented value is shown, as at rom0(an1) 7 changed to 8 and so the sum changes from 0F to 10


When sw[15:14] = 10(write), value on sw[0:3] will be written in ram0 at address Given by sw[4:13]


The updated value is benign shown

7. Conclusion

This lab successfully demonstrated the design and implementation of a memory-based vector processing system on an FPGA. Key Verilog techniques like edge-detection for single-cycle operations and multi-stage reset conditioning were crucial for a robust implementation. The functionality was thoroughly verified through simulation, and analysis of the hardware utilization and schematic confirmed the design was synthesized as expected.