Indian Institute of Technology Kharagpur

AUTUMN Semester, 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Computer Organization and Architecture Laboratory

Addendum: Final FPGA Guidelines for Memory Operations

October 14, 2025

Instructions: *This document provides specific guidelines for the FPGA demonstration of the memory subsystem, focusing on the* `LD` *(Load) and* `ST` *(Store) instructions from the main assignment,* ***"Experiment: Design and Synthesize a 32-bit Processor using Verilog"****.*

Context and Objective

The main assignment requires the design of a complete 32-bit processor, which includes both instruction and data memory. However, for this specific FPGA demonstration, the focus is narrowed to the data path's interaction with the memory system. You will implement the **data memory** and demonstrate its functionality. The instruction itself will be manually provided via the board's switches, bypassing the need for an instruction memory in this particular task.

FPGA Task: Demonstrating the Memory System

For this demonstration, you will implement your 32-bit processor's datapath along with a data memory system. The goal is to prove that your processor can correctly read from and write to this memory using the specified base-addressing mode, with instructions controlled by the switches.

Key Implementation Requirements:

- **Data Memory Implementation (BRAM Only):** You must implement a 32-bit, byte-addressable **data memory** using the FPGA's Block RAM (BRAM) resources. This is a firm requirement. Using a register array to synthesize memory is not permitted, as BRAM is far more area-efficient and represents the standard industry practice for implementing on-chip memory.
- **Pre-loaded State:** To simplify testing, both the data memory and the register file must be **pre-loaded with non-trivial initial values** directly within your Verilog code. These values should persist until a reset signal is applied.
- **Switch-based Instruction Control:** The 16 switches on the FPGA will be used to physically represent the instruction word. You will manually set the switches to define the opcode and operands for the `LD` or `ST` instruction to be executed.
- **Two-Button Control System:** You will use two separate general-purpose push-buttons for control.
  - **Reset Button:** One push-button (e.g., `BTNU`) must be assigned to act as a **synchronous reset**. When pressed, it should restore your processor, register file, and data memory to their initial, pre-loaded state.
  - **Execute Button:** A second push-button (e.g., `BTNC`) will serve as the **single-step execute** signal. When pressed, it should cause the processor to execute exactly one instruction based on the current switch settings.

- **Multiplexed Output for Verification:** To view 32-bit results on the 16 available LEDs, you must implement a multiplexed display. This is a common Design-for-Testability (DFT) technique used to observe wide internal data buses when the number of physical output pins is limited. A dedicated switch will select whether the **lower half (bits [15:0])** or the **upper half (bits [31:16])** of a register's content is displayed.

Example Switch Configuration

The following table provides a suggested mapping for the 16 switches.

Table 1: Suggested Mapping of Switches for Memory Operations

Switch(es)	Bits	Function
`SW[15]`	1	**DFT Display Select**: `0` for lower 16 bits, `1` for upper 16 bits.
`SW[14]`	1	**Operation Select**: `0` for `LD`, `1` for `ST`.
`SW[13:10]`	4	**Dst/Src Register ('Rd'/'Rs') Address**.
`SW[9:6]`	4	**Base Register ('Rb') Address**.
`SW[5:0]`	6	**Signed Immediate Offset** (Range: -32 to +31).

Suggested Demonstration Flow

During your demonstration, you will first use your designated reset button to initialize the processor, then use the execute button to step through operations.

Part 1: Demonstrating the `LD` Instruction

1. Press your designated **`reset` button** (e.g., `BTNU`) to ensure the register file and data memory are in their known, pre-loaded state.
2. Explain the initial values you have loaded, for example: 'Mem[0x84] = 0xCAFEBABE' and 'R3 = 0x80'.
3. Configure the switches for an `LD` instruction to fetch this value (e.g., 'LD R5, 4(R3)').
4. Press the **`execute` button** once. This performs the operation 'R5 = Mem[R3 + 4]'.
5. Use the display select switch (`SW[15]`) to show the TA that the upper (`0xCAFE`) and lower (`0xBABE`) halves of register `R5` now hold the correct data.

Part 2: Demonstrating the `ST` Instruction

1. Explain the initial values for this test, for example: 'R2 = 0x12345678' and 'R4 = 0x200'.
2. Configure the switches for an `ST` instruction (e.g., 'ST R2, -8(R4)').
3. Press the **`execute` button** once to perform the store.
4. **Verification Step:** Reconfigure the switches for an `LD` instruction to read back from the *same memory location* into a different register (e.g., 'LD R9, -8(R4)').
5. Press the **`execute` button** again.
6. Use the display to show that register `R9` now holds the value `0x12345678`, proving the store was successful.