VLSI Architectures (Sem-2: AY 2024-25)

Lab-9

Objectives: To develop and test the function of the MEM stage in a 5-stage pipelined implementation of RISC-V RV32I instruction set using a behavioral Verilog model

For this purpose a first cut Verilog module named “ rv32i\_mem\_stage” is provided to you. This module attempts to behaviorally model the functioning of the MEM stage logic, but has not yet been compiled and tested.

Understand the Verilog code (vis-à-vis the requirements of the load and store instructions of RV32I. Document any logical or syntactic errors and then correct them.

Develop a test bench to test and validate the functional correctness of the module for each variant of the load and store instructions (e.g. LB, LH, LW, LBU, LHU, SB, SH, SW) by providing through the test bench appropriate values of the inputs required by the module to execute the memory access functions of those instructions.

In fact, this module is a programmable logic block that performs all the memory access related functions for the “load” and “store” instructions (in the same way as the EXE stage logic performs the data operations and address computation functions for all the instructions).

Check out the functional correctness of the module for mis-aligned “load” and “store” of a byte, a half-word and a word also (besides aligned “load” and “store”.

Your test bench can apply a new set of inputs corresponding to a new “load” / “store” instruction every 50ns.

You may also add an initial statement in the provided module “rv32i\_mem\_stage” in order to initialize the data memory as per the requirements of your test bench.

\*\*Please note : While writing your test bench, your memory address (32-bit low-end byte address for a memory access of size byte/half-word/word) should lie in the range 0 to 1020 (even though the address has been modeled as a 32-bit number as per RV32I addressing range). Upper 22 bits of all your addresses should be zeros.

The data memory size in the module has been declared with the above restriction in mind so that the module can work on any machine without making a huge demand on the computer’s memory.