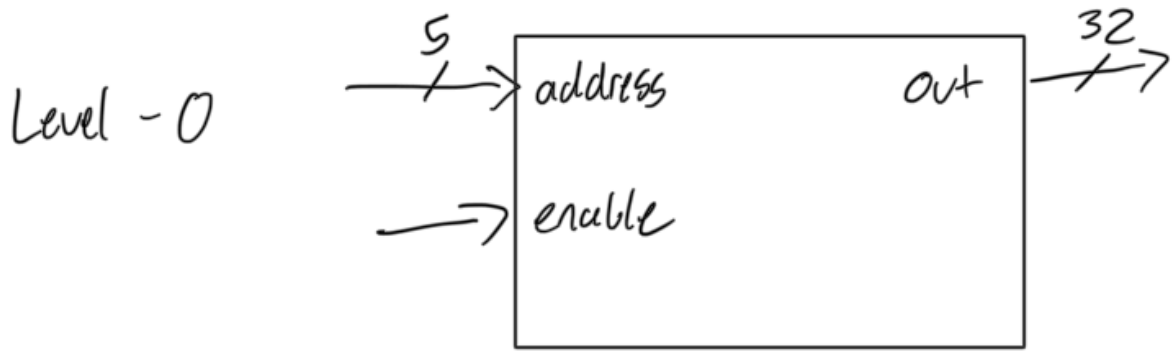
Lab 6 — ROM (RISC-V)

Objectives
For this lab, I am creating a ROM module for a processor which uses RISC-V ISA. This ROM will hold 19 specified instructions, each being 32 bits wide.

Introduction
ROM is read only memory. Some ROMs are erasable or electrically erasable and programmable. The ROM I am designing to hold the 19 specified instructions for this lab will not be erasable or programmable. It will only store the instructions which are specified in the Verilog code.

Methodology
The following is a level -0 diagram of my ROM. It has an 5 bit address and 1 bit enable input, and a 32 bit output. When the enable input is high, the output will be whichever instruction corresponds to the 5 bit address input.

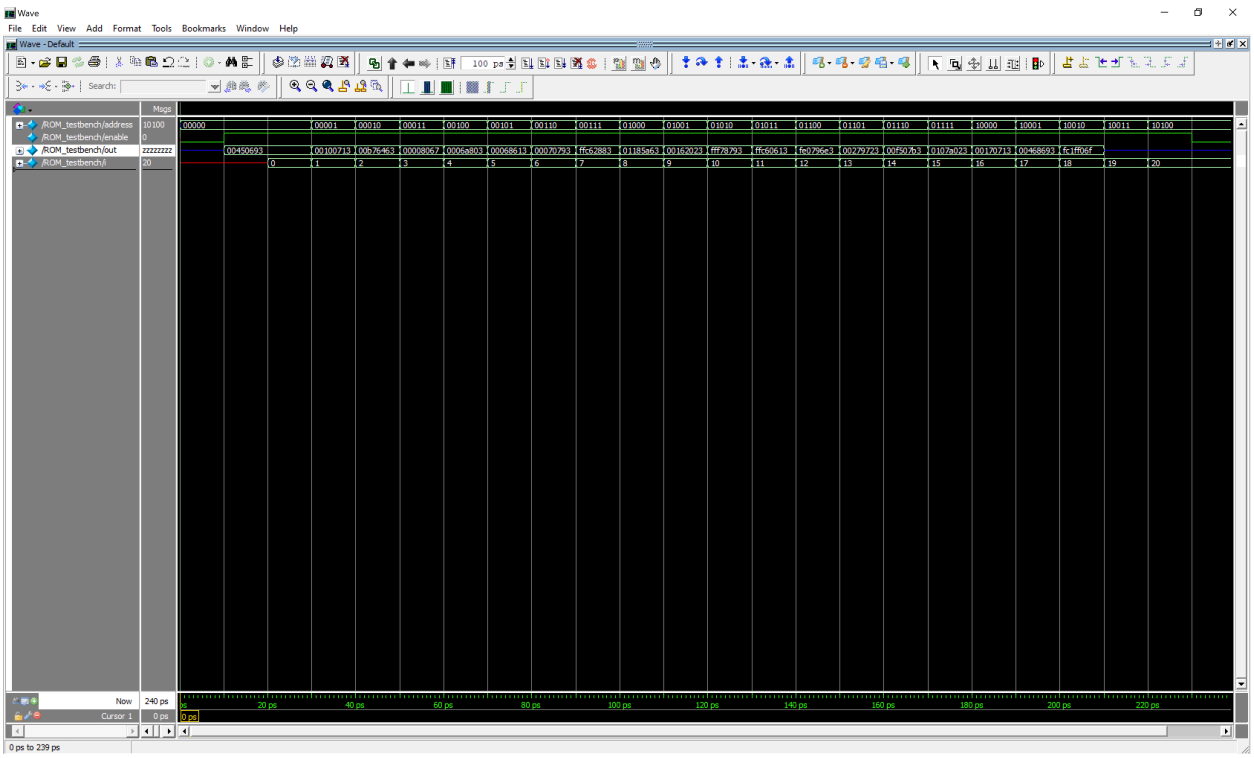


My ROM implements a case statement to set the output equal to an instruction set to correspond to a certain address. When enable is low, none of this happens. If enable is low or the address input to the module has no associated instructions, the output is set to hi-z.

Result and analysis
When tested in ModelSim, each instruction can be reached by inputting the correct address.

All code, diagrams, and simulation results can be found as files on GitHub: https://github.com/Eth7an/Lab-5.git

A video explanation of the top level design and ModelSim simulation can be found here: https://youtu.be/7hg_9xPprTg



Discussion and Conclusion

To conclude, I have built a fully working and tested RAM module. The functionality of all inputs and outputs has been tested and verified, and this module is complete and ready to be instantiated in higher level projects.