
ECE 436 Laboratory 3

INTERFACING WITH MEMORY

Description:

This laboratory will expand on your design from the prior week's assignment by introducing you to the memory interface. Remember that in last week's assignment, you completed a register file implementation and used a control block to interface the register file to the ALU, designed in the first week. This week, you will take your existing design and incorporate a memory interface simulator that we have provided.

As described in the Jackal ISA reference document, the memory interface consists of the address bus, data input/output, clock, and control signals (rd, wr, rst, and done). In addition, you must implement a program counter (capable of being incremented – branching and jumping is not yet necessary) and an instruction register. (Note: you may also incorporate other components as you see fit, such as an MAR or MDR.) The control unit must drive the control lines so that your design is capable of fetching instructions and executing instruction opcodes '0000' through '0111' (i.e. the LD/ST instruction in addition to the arithmetic/logical instructions you have already been implementing).

At this point, you may wish to use StateCAD for your control unit, if you haven't already done so. The state machine editor will greatly simplify your ability to encode states and define a proper control sequence.

Process:

For this lab, you will use a simple SDRAM controller/memory simulator component (sdramsimu.vhd is available on the Toolkit webpage) that has the same interface as the real SDRAM controller. Take a close look at the VHDL code for the simulator component. Note that a 32 byte (16 addresses X 2 bytes each) memory array is defined, which will actually be synthesized and placed-and-routed onto the FPGA slices (as opposed to the regular off-chip memory you will use in the final design). This simulator allows you to pre-load the memory with different instructions and data by editing the VHDL code (see the table array). You should insert different instructions and data into this memory block to fully test your design.

Notes:

- Note that the simulator component is designed so that the read/write is completed after two clock cycles. However, remember that the actual memory will not be deterministic. Instead of waiting a defined number of cycles for the read/write to complete, you must wait instead for the 'done' signal.
- Remember that reset occurs on logic low.
- Each of the components may be implemented using any of the design entry utilities that the tools offer: schematic editor, state machine editor, and VHDL editor.
- Be mindful of timing issues related to interfacing with memory.

Deliverables:

We will be expecting the following items in the post-laboratory report that you need to submit electronically before the due date. Please note that you do not have to print all of your VHDL or schematics for this write-up; however, any diagrams, schematics, VHDL, etc., that you think are useful in describing your implementation are welcome (i.e. don't waste space, but clearly explain the key points).

- A description of your program counter implementation.
- A description of your control unit implementation.

- A description of the dataflow.
- A description of your verification plan and the simulation output for verification of your design.
- A writeup of your synthesis results (i.e. optimizations, area values, delay values, etc.).
- A summary of any problems encountered or critical lessons learned from the assignment.