Project 2: Single Cycle Computer (SCC)

CSEE 4290 Fall 2022

Objectives:

1) Design a SCC using the concepts of 5 main parts of the processor: IF, ID, EXE, MEM, WB
2) Based on the CIA
3) Create and use a test bench to validate the design
4) Use class created sample code and memory model
5) Bonus: Implement a prefetch for must take branches and normal next instruction count.
    a. This makes the SCC into a hybrid two stage pipe
    b. You will NOP incorrect next instructions
6) Bonus: Implement a simple emulator
    a. Given opcodes, provide expected output of SCC

Groups:

The project is done in groups and graded as a group. Grading will be done by the groups as well.

The Project Overview:

Using the CIA (Class Instruction Architecture) as your target, design a single cycle computer in Verilog to support the Opcodes in the CIA. A single cycle computer runs one instruction at a time. There is no pipeline. Instruction two does not start until instruction one completes. The assumption for a SCC is that the time required to execute any and all instructions can be completed in one clock cycle.

You will reference the CIA document. It contains explanation and examples for each supported instruction. You may also use the ARM reference documentation as well. If the instruction is listed as (optional) in the CIA document, you do not need to implement in Verilog for this project.

You should support all of the instructions in the CIA document.

Even though it is a SCC design, it is still static. It is clocked. Registers/flops, should only change when they are required to change based on the instruction. Assume the inputs into the SCC come from a clocked source. The outputs of the SCC go to flops prior to output. Registers are flops; they change of rising clock edges.

The memory model is also clocked.

There is one global clock driven from test bench.

Assumptions for the Project:

1) You only need to support 32bit data widths. Of course all instructions are 32bits wide since it is ARM-like RISC architecture. Any 64bit wide registers or instructions are not supported.
2) If the instruction sets a flag (carry flag, for example), support the flag.
3) You do not need to use optimized architectural design for parts of the ALU. For example, you can just add two registers using the "+". You do not need to instantiate carry look ahead adders as we did on the first project.
4) This is your code. Comment it. Do not cut and paste other code.
5) Try to use meaningful names. Use variable names. Use parameters where appropriate.

Wrapper

All designs must use the same I/O wrapper so that we have consistency on the I/O names for debug/test.

Test Bench

The testbench must use the memory model(s) created by the class.

Final functional testing will use the same instruction opcode stream for all groups.

Functional Test Code

You will use the class designed program(s) and others to stress and validate the function of the design.

Here are ideas for additional test programs:

1) A number sorting algorithm given a starting and ending address range for the sort
    a. Assume base 16 numbers 2 bytes wide (a word)
    b. Data is 4 bytes wide so a 4 byes of data has two words (A double word).
    c. Sort at the word level
2) A pattern search algorithm given a 1-4 byte pattern, a starting and ending address.
3) A Hex to decimal convertor for 4byte data supports 2's complement format for hex.
4) A countdown timer. Set starting value. Timer counts down to zero. Each count is a length of time (number of clocks) which is changeable.

Verilog Structure

1) Hierarchical: A system level which includes the SCC top level and memory models. A top level SCC wrapper for I/O only. Inside the SCC: a) One Verilog file for each of the five main stages. b) One file for the registers. c) dataflow and address muxing. There should be no logic in the top level wrapper.
2) Do not mix dataflow (memory, registers) and random logic.
3) A testbench file template
4) Multiple testbenches that test the full regression suite, subsets of tests, a debug environment, etc…

Documentation

1) All Verilog should be commented
2) Description in Word of:
    a. Testing plan: How to validate/verify the design
    b. Description of each of the test programs (with assembly included) and how they map to the test plan AND to the function of the CIA
3) Bug map and count. Tracks bugs, where found, how found, and number found per (day, or week, or test run)
4) Changes recommended for CIA
5) Group member roles and responsibilities

GROUP

Any design must be validated with no prior bias. Therefore, a design team is typically made up of independent design and verification teams. Each of these teams need a project manager and division of labor.

Groups to fill:

Design of SCC: Five sections, Regs, top level, documentation, design level test and debug

Testbench: Class program, additional programs to stress, mapping to CIA and plan, documentation, system level test and debug

Grading

1) Completion of design compared to CIA and test plan
2) Documentation
3) Schedule (due end of October (10/31).
4) Self-feedback and group feedback
5) Professionalism of work product and effort
6) Design team will assess Test team's work
7) Test team will assess design team's work
8) If the group implements a bonus, bonus amount tbd based on quality/results of effort.