COS/ELE 375 Assignment 3

Fall 2015, Princeton University
Due date: 12/17/15

Submission guidelines:

Your course enrollment (COS/ELE 375) and date of submission should be listed with your name on the front page of your submission. Please submit your assignment in class or in the box outside of CS-241.
LATE SUBMISSIONS: Record the date and time of your submission.

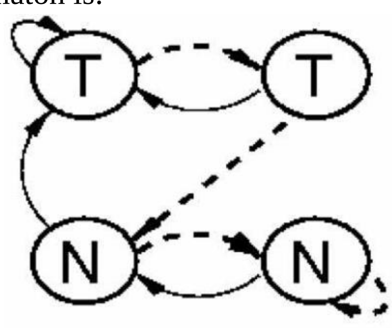
Collaboration Policy:

You can collaborate on your project team basis. Each team only needs to submit one homework with all the 4 or 5 names on it!

Problems:

1. Build a Verilog RTL module for a 2-level Gag branch predictor with a 1-bit global history register and a global table of A4 automata. For your submission, please print out the code. Handwritten code has no points.

   The state machine for the A4 automaton is:



Automaton A4

   The shell for the Verilog you will want to write is:

```
module branch_predictor(clk, reset, enable, taken, prediction,
overall_state_debug);
      input clk, reset, enable, taken;
      output prediction, [4:0] overall_state_debug;

      // Meaning of states in A4 automaton
      // 00 - Strong Not Taken
      // 01 - Weak Not Taken
      // 10 - Weak Taken
```

```
    // 11 - Strong Taken
endmodule
```

The module defines four inputs and two outputs. The `clk` input is a clock signal. The `reset` signal tells your module when it should reset its state. The `enable` signal tells your module when it should look at the `taken` input and use it to compute the next state (both automaton and history register). Update the automaton corresponding to the current state of the history register, then update the history register. The `prediction` output, is the prediction that would be made given the current state. The `overall_state_debug` output is for debugging and is just a dump of the current states of history register and two A4 automata states in the following order:
bit [4:3] for automaton 1;
bit [2:1] for automaton 0;
bit [0] for history register;

The module has simple semantics. Whenever there is a positive edge on the `clk` signal, the module should look at the `reset` signal. If it is high, then it should reset its state. On reset, the history register is 0, and the automata are in the Weak Taken state. Otherwise, if `reset` is low, it should look at the `enable` signal. If the `enable` signal is high, it should use the value of history register to find the corresponding automaton and use the `taken` signal to change both the state in that automation and the history register as well. Your prediction should be high when you would predict a branch to be taken and low otherwise.

Note: each team should install Xilinx on at least one laptop from your teammates'. Follow the tutorials on course website for installing Xilinx, and synthesizing and simulating Verilog files via Xilinx ISE. Make sure your Verilog code can pass synthesis step and simulate correctly. You can access a test bench at */u/ee375/public/share/HW3Q5/test_bench.v*.
One thing to note, please change the "bp_state" variable to 5 bits instead of 2 bits. The file you downloaded directly is wrong at this variable.
Look at the test bench code to see what it is doing. Simulate your Verilog file together with the test bench to check the functionality.

2. You are given an empty 16K 2-way set-associative LRU-replacement cache with 64 byte blocks on a machine with 4 byte words and 32-bit addresses. Describe a memory read address sequence which yields the following Hit/Miss patterns. If such a sequence is impossible, state why.
   - Miss, Hit, Hit, Miss
   - Miss, (Hit)*
   - (Hit)*
   - (Miss)*
   - (Miss, Hit)*

3. For the cache of the prior question:
   a) How many sets does it have?
   b) Give the address bit ranges for the index, block offset, word offset, and tag bits (0 is LSB).
   c) Consider a reference to address 19423. Which set does this map to in the cache?
   d) Including both tag and data bits (and 1 valid bit per line) how many total SRAM bits are needed to implement this cache?

4. A byte addressable machine generates 32-bit virtual address and 27-bit physical addresses. To achieve high performance, bits used to index a physical cache set must NOT go through TLB before they can be used. We have also decided, after preliminary performance study, that the cache block size should be equal to 32 bytes.
   a) If we insist that the page size is fixed at 2^{14} bytes and that we have a 4-way set associative physical cache, what is the maximal physical cache size allowed? Show the break down (the number of bits for *tag, index, byte select*) of the address used to access the cache.
   b) If we insist that the page size is fixed at 2^{14} bytes and we would like to have a 2^{17}-byte physical cache, what is the minimal set associativity of the cache? Show the break down (the number of bits for *tag, index, byte select*) of the address used to access the cache.
   c) If we insist that we have a 2^{15}-byte direct mapped physical cache, what is the minimal page size? Show the break down (the number of bits for *tag, index, byte select*) of the address used to access the cache.
   d) Compute the ratio of the tag store in 4(c) over that of a virtual cache of the same block size and set associativity. Explain why virtual cache can actually be slower than its physical counterpart.

5. This problem is about the TLB performance for an 8-bit microprocessor. Assume that the processor uses a one-level mapping algorithm and the page size is 64 bytes. We have the following virtual address sequence generated by the microprocessor:

   (11100101, 01111110, 01100001, 00011010, 11101100, 11110101, 10101111, 10100111, 11100000, 10111011, 00110111).

   Assume that a 2-entry TLB is used. Assume that before the sequence is performed, the TLB contains PTEs (page table entries) for pages 11 and 00.

   Derive the hit ratios for the access sequence if we use a 2-way set associative TLB with LRU replacement algorithm. Start with initial state. Show the TLB tag store contents and hit/miss status for each access.