ECS 154A Final Project
Cache Building
Total Points: 215

For you final project in the class you will be implementing a cache in Logisim.

Cache Specifications

Parameter	Value
Number of bits for primary memory	10 bits
Cache Size	24 Bytes
Number of Sets	2
Number of Ways (lines per set)	3
Line Size	4 Bytes
Write Policy	Write Back
Eviction Policy	Least Recently Used

Input Pin Specifications

Input Pin	Width	Explanation
CpuAddress	10 bits	The address that should be read from / written to
CpuRead	1 bit	1 if the CPU is trying to read the given address and 0 otherwise.
CpuWrite	1 bit	1 if the CPU is trying to write to the given address and 0 otherwise.
CpuWriteValue	8 bits	The value the CPU is trying to write to memory if CpuWrite is 1.
LineFromMem	32 bits	The line you requested to read from memory.

All of the inputs will only be valid for a **SINGLE** clock cycle except for LineFromMem. LineFromMem will be valid for as long as you choose to keep reading from memory

Output Pin Specifications

Output Pin	Width	Explanation
Ready	1 bit	The cache is ready to process a new request/has completed the current request.
DidContain	1 bit	1 if the cache contained the element at CpuAddress and 0 otherwise.
ByteRead	8 bits	The value of the byte at CpuAddress when the CPU makes a read request.
MemAddress	8 bits	The address to memory that your cache would like to read from or write to.
MemRead	1 bit	1 if your cache would like to read from memory at MemAddress and 0 otherwise
MemWrite	1 bit	1 if your cache would like to write memory at MemAddress and 0 otherwise
Line2Mem	32 bits	The line your cache would like to write to memory at MemAddress

The reason that MemAddress is only 8 bits as opposed to 10 is because our memory is going to be line addressable. This means that each line, as opposed to each byte, has an address. This makes things easier for us as we can write an entire line to memory in a single clock cycle.

Other Specifications

1. Your circuit must complete each read or write operation within 10 clock cycles. Mine can finish within 7 on the longest path but I've given you a little extra just in case.
2. You may use any of the builtin in Logic Components.

Testing

To test your program do the following.

1. Open Cache.circ
2. Open the Subcircuit called Inputs
    1. Right click the ROM and select Load Image.
    2. Select the test you want
    3. Example: seq_read1.txt
3. Open the Subcircuit called Checker
    1. Right click the ROM and select Load Image.
    2. Select the matching solution for the test chosen in Step 2.
        1. Example: seq_read1_seq_mem_sol.txt
4. Go back to main and right click the RAM and select Load Image
    1. Select the matching memory file for the test chosen in Step 2.
        1. This has to be done every time you reset the circuit :(
    2. At the top of each solution file it will tell you the matching test file and memory file
    3. At the bottom of each solution file it will tell you the number of correct points in that test as well as the correct hit count.

I recommend testing in the following order: seq_read1_seq_mem_sol.txt, seq_write1_seq_mem_sol.txt, set1_targeted_read_seq_mem_sol.txt, set0_targeted_write_seq_mem_sol.txt, random0_seq_mem_sol.txt, and final_test_rand_mem_mem_sol.txt.

Your circuit will be tested against final_test_rand_mem_mem_sol.txt or a similar such file to determine your grade. So final_test_rand_mem_mem_sol.txt is the test you are aiming to pass.

The sum of hit counter and miss counter should be equal to the the test number – 1 once the test is completed.

Your grade will be calculated as follows: correct counter + hit counter + miss counter.

Hints

1. Components I used: Basic logic gates, mux, dmux, register, splitter, comparator, bit finder.
2. In case you didn't know you can copy the logisim tables from/to Excel or another spreadsheet program. This is a good idea because you will probably need to make some edits to your state machine.
3. Before you start drawing in logisim, make sure you really lay down the step by step process of how a cache works. This will help you figure out what needs to be inside the cache to make it work.
4. You should approach this problem in a top down manner and construct the data path first
    1. First start with the cache. A cache is made up of sets and sets are made up of lines.
        1. You should build them in that order using a subcircuit for each one. Figure out what each should input and each should output and then go fill in the internals. I did bottom up and wasted a lot of time because I kept realizing I would need more inputs or outputs for a particular subcircuit and so would have to keep redrawing the connections in the higher level circuits.
        2. You are really, really, really going to want to use subcircuits for this assignment.

5. After you complete the data path figure out the control signals. My controlling FSM had 7 states.
6. You can greatly simplify your logic by just thinking that you will get your value from a set or a line. Obviously you want to use the right one, but you can easily select among them by using muxes and make sure that right one gets the correct input by using dmuxes. This will save your from having to enumerate all the possibilities of where the element you are looking for could be.
7. Your circuit does not have to start out being ready. I needed an initialize state so that I could set the ages of the lines and as such I wasn't ready to begin until after I had passed through that state.
8. Here are some of the mistakes I made that I made solving the problem. Hopefully you can learn from them
    1. Forgot to rename some labels
    2. Needed to both add and remove states from my FSM
    3. Messed up on the aging of lines. I assumed that they would all start with unique values but of course they don't.
        1. The fix was to add an initialization state where I would set the ages of the lines to 0, 1, and 2.
    4. Used the wrong Tag bits to when addressing memory on writes.
        1. Fix. When writing to memory you should use the Tag bits found in the Line that you will be writing. When reading from memory you should use the Tag bits from the CpuAddress.