# Part 1

1.

* The purpose of firmware sub-circuit is to provide 8-bits of data at a 10-bit address (up to 1024 addresses).
* The oneKram circuit takes an input address, data to be written, clock, and read/write flag. When the clock is on: it will either write the input data to the input address or output the data at the input address.
* The 10-bit adder controls the address generation.

2.

* Read/write (read\_write) control signals are floating. They are utilized to provide default a default for accessing the memory in the oneKram circuit.
* These components implement the floating input detection:

A diagram of a circuit

AI-generated content may be incorrect.

When read\_write is disconnected, default behavior is provided – that is, writing and output to the corresponding ram circuit is set to 0.

* This relates to the concepts in the study guide because it utilizes a buffer and pull resistor. The oneKram circuit takes advantage of a buffer and pull resistor to provide a default state for when read\_write is disconnected.

3.

* When sweep\_enable input is activated, the write/enable for the register (at the top of main) is activated when the clock is on, the output of that register (memory address) is fed into the 10-bit adder, incremented by one and passed to the address tunnel and re-input to the register. The address is then passed onto the firmware circuit which hands off data to the oneKram circuit. This process continues so long is halt is not active.
* Through rw\_sel. If rw\_sel is 0, data at the input address to oneKram is read and output to rd. If rw\_sel is 1, the write data (wd on the oneKram circuit) at the input address is written.
* To select the various operations to perform on the firmware’s data at a given address with respect to the oneKram circuit.

## Part 3.1 – Testing

Read the captions and look at the red boxes in each screenshot.

* Memory protection for addresses below 0x0FF: A diagram of a computer

  AI-generated content may be incorrect.

When trying to write with memory protection on, and at an address lower than 0x0FF. Take note that the mux to select rw\_sel is disconnected when trying to write at an address lower than 0x0FF.

A diagram of a machine

AI-generated content may be incorrect.

Writing to memory with memory protection on when above an address higher than 0x0FF. Take note of the mux selecting rw\_sel.

* Error detection for invalid switch configurations:

A diagram of a computer

AI-generated content may be incorrect.

Trying to read and write to memory at the same time. Take note that halt is set to 1 in this case.

A diagram of a software system

AI-generated content may be incorrect.

Selecting either to write or read will not halt.

A diagram of a software

AI-generated content may be incorrect.

Selecting either to write or read will not halt.

* Display correctly shows address and data values: A computer screen shot of a computer

  AI-generated content may be incorrect.

Display of address and value (when read is enabled). Note, In binary, 0100110001 is 0x131.

A computer screen shot of a computer

AI-generated content may be incorrect.

The corresponding graphic to the image above. Note that the oneKram circuit matches the display above.

## Part 3.2

1. Memory Protection: I used a comparator, multiplexer, constant and AND gates for this functionality. The multiplexer is wired up to write\_mode (top option) and the > (lower option) of the comparator for inputs. The select for the multiplexer is wired up to mem\_protect. The ouput for the multiplexer is wired up to the multiplexer used to select rw\_sel. When mem\_protect is 1, it will select the > output of the comparator which will only be activated when the address is greater than 0xFF (meaning write\_mode has no bearing on rw\_sel until address > 0xFF). Otherwise, if mem\_protect is 0, write\_mode can affect rw\_sel freely. The mem\_fault bulb is connected to a NOT gate and a 3-input AND gate which is wired up to mem\_protect, write\_mode, and the < of the comparator. When, all three inputs for the AND gate are true, the bulb is turned off (colored red). In other words, a memory fault has been commited (because address < 0xFF, mem\_fault is set to 1 and write\_mode is set to 1) – dun dun dun. Otherwise, the bulb remains green.A diagram of a circuit

   AI-generated content may be incorrect.
2. Configuration Error Detection: I used an AND, OR and NOT gate to implement this functionality. read\_mode and write\_mode are wired up to the AND gate. The output of the AND gate is wired up to the OR gate. The OR gate’s other input is the = of the comparator that determines when to halt. The OR gate’s output is wired to halt. When read\_mode and write\_mode are 1, the AND gate is 1 and the output of the OR gate is 1, causing a halt to occur. Part of the line from the output of the AND gate to the OR gate is wired up to a NOT gate which goes to a bulb. The bulb is green when read\_mode and write\_mode are not 1 at the same time. Otherwise, it is red. A computer diagram of a program

   AI-generated content may be incorrect.
3. Hex Display: For this I created a circuit involving several splitters, one 10-bit input and 3 outputs. I used this circuit in main. To explain how this circuit works I will explain by example. Take the 10-bit address : 0010111001. First, I used a splitter to separate the 10 bits into a group of 4 bits and 6 bits. In the case of the example, 0010111001 (red is lower 4 bits, blue is upper 6 bits). The lower 4 bits were then wired up to an output of 4 bits. The upper 6 bits were wired up to another splitter. This splitter then split the 6 bits into another 4 bits and 2 bits. In the case of the example:

After first split: 001011 1001

After second split: 00 1011 1001

The bits in the orange, were wired up to another 4-bit output. And you guessed, the upper 2 bits or the purple bits were wired up to another splitter. This splitter has two constants and the last 2 bits. This splitter was then wired up to the final 4-bit output. It’s best to look at the implementation of the circuit: A diagram of a computer code

AI-generated content may be incorrect. A computer diagram of a computer

AI-generated content may be incorrect.