11/23/2020

I am not sure of what I will use the rom module for at the moment as I will not be making a proper os for this computer. Maybe I could use the rom as a code save-point, something where I can transfer code into the computer after writing it in a high level language. This would allow me to write more code as I will be more accustomed with the interface, and additionally the code would be saved. However, one problem I had with the 8 bit computer was that every time I removed its eeproms to re-write their contents, the pins would be damaged during removal and insertion into the breadboard.  
  
Possible solutions:  
  
-better chip interface – use rom or eeproms programmer  
-use wires to connect a raspberry pi to module, no removal needed   
 - needs pi to be programmed -> extra work  
 - can be tailored to needs better  
 -coding experience

The ram module should have 65536 addresses to work with. Since I intend on keeping the address bus 16 bits long, this means the rom module will waste some of the ram addresses. I am thinking about doing the first 10 bits for the ram module, meaning that 1024 addresses will be persistent and read only in the computer environment. This is not reflected in the rom modules chip design, I will only be using the first 10 bits of the rom module.

-in order to do this, I will need to have two rom modules connected together and only allow the first two bits of the second module to be used.

- I am starting to think that I may have more time and motivation to complete this project if I use an abbreviated form of multisim design.

* Have a single line representing connections, and the range of bits they represent:  
  i.e input 0-7

11/24/2020  
1:37 pm

Separate wires will be connected to the rom chip’s input, vcc, ground and enable pins for raspi connections to be made. When programming the rom chip, the computer should be off so that signal interference wont occur.