**ECEGR 2220: Microprocessor Design**

**Spring 2018**

**LAB 5 REPORT**

**Name: Don Le**

**Performed by:**

**Don Le - Thanh Nguyen – Lauren Molina**

**Date of Report: 05/27/2018**

**SEATTLE UNIVERSITY**

**Department of Electrical and Computer Engineering**

**Part 1: RAM**

In this part, we design a RAM module which has following entity definition:

Reset - causes all the data in the RAM to be cleared to 0 when Reset = 1

Clock - is the input clock of the system

OE - Output Enable signal, data is read from RAM if OE = 0, Z otherwise

WE - Write Enable signal, data is written to RAM if WE = 1

Address: address of 32bit word to read/write from/to RAM

DataIn - 32bit data to write to RAM at Address if WE = 1

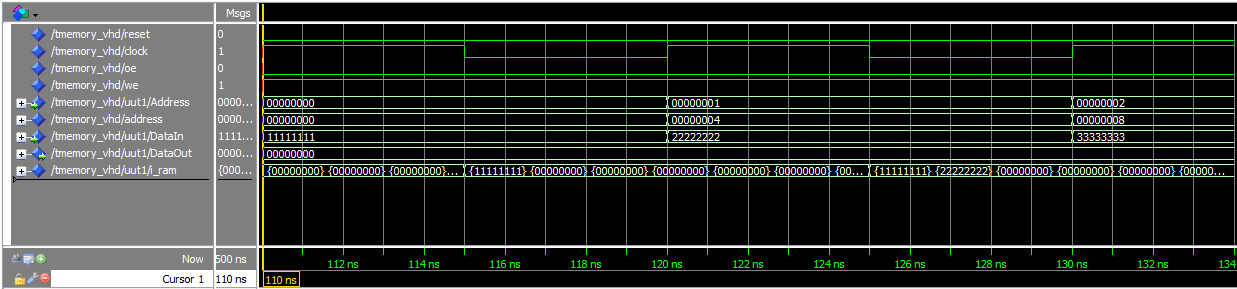
DataOut - 32bit data output (read from Address if OE = 0)

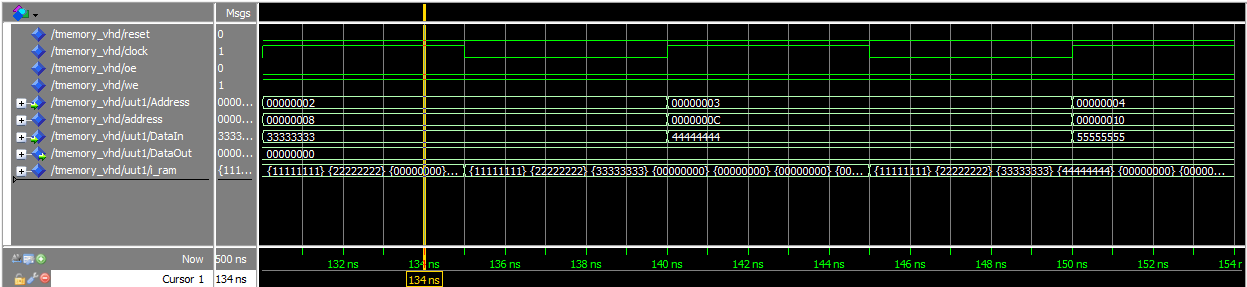
When WE = ‘1’ on the falling edge of clock, the DataIn will be latched into the Ram module.

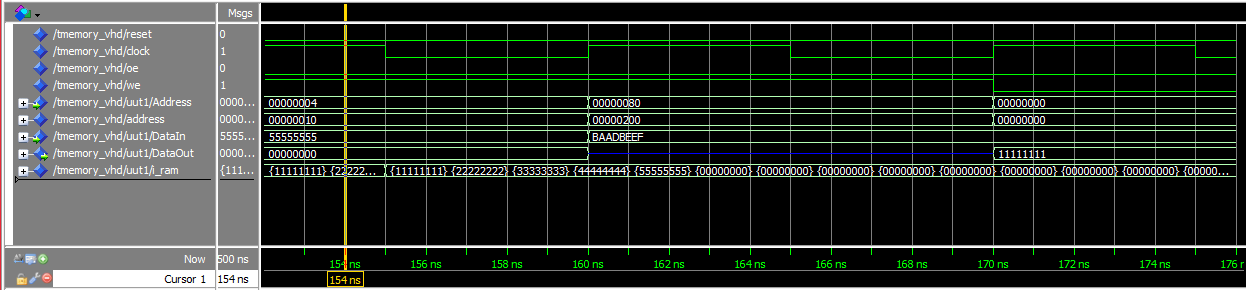
When OE = ‘0’ on the rising edge of clock, the DataOut will be driven out from the data in RAM immediately.

* **TESTING:**

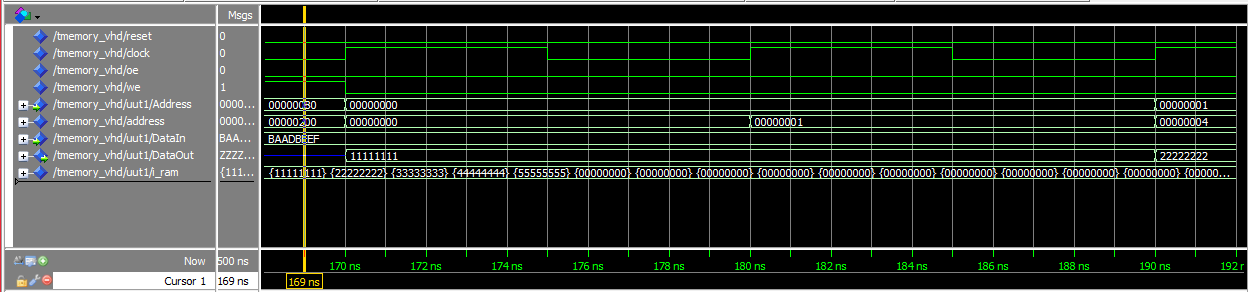
As we can see, when WE = ‘1’, data from datain is written to i\_ram, at falling edges of clock.

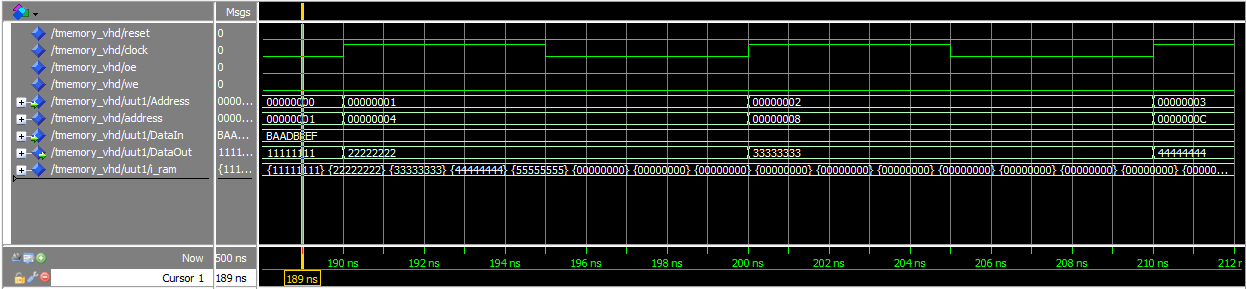


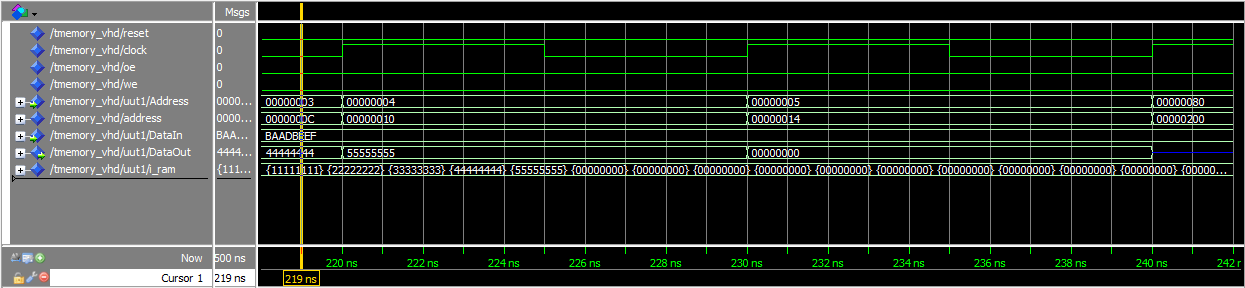


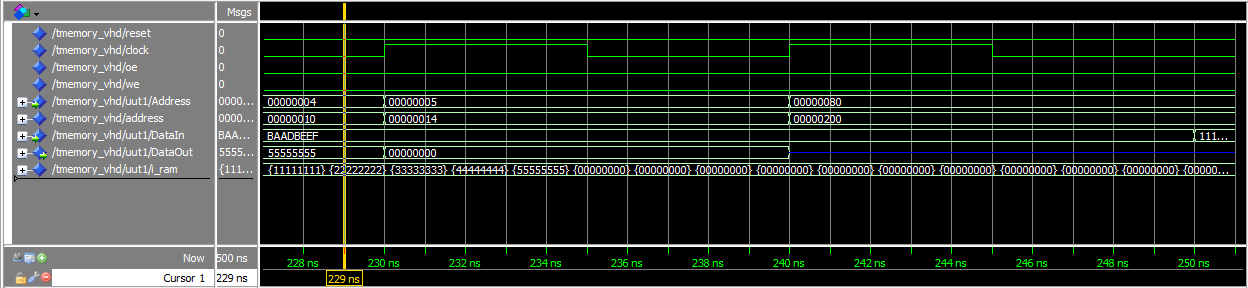


When WE = ‘0’, OE= ’0’, DataOut starts taking value from i\_ram, at rising edges of clock









From the last picture, as the address has the value of 0x00000200, which is out of range (bigger than 0x000001FC), DataOut gets the value of 0xZZZZZZZZ

* Explanation of why the][][][][][][ Address port for the RAM module is only 30 bits wide and not 32 bits:

Since an address of RAM module is already shift 2 bits, we don’t need to care the last 2 bits, the address port for the RAM module only needs to be 30 bits wide, instead of 32 bits.

**Part 2: Register Bank**

In this part, we implemented the register bank of the RISC-V processor using the register32 component to be the basis of our registers and has the following entity definitions:

ReadReg1 – Index of the register to read and output to ReadData1

ReadReg2 – Index of the register to read and output to ReadData2

WriteReg – Index of the register to write to with the value of WriteData

Write Cmd – 1 = write data to WriteReg; 0 = no op

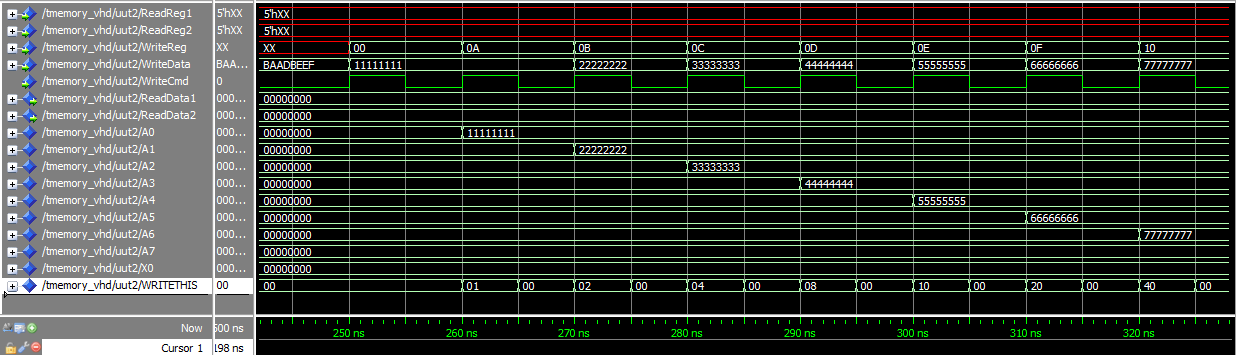
ReadData1 – Value read from register indexed by ReadReg1

ReadData2 – Value read from register indexed by ReadReg2

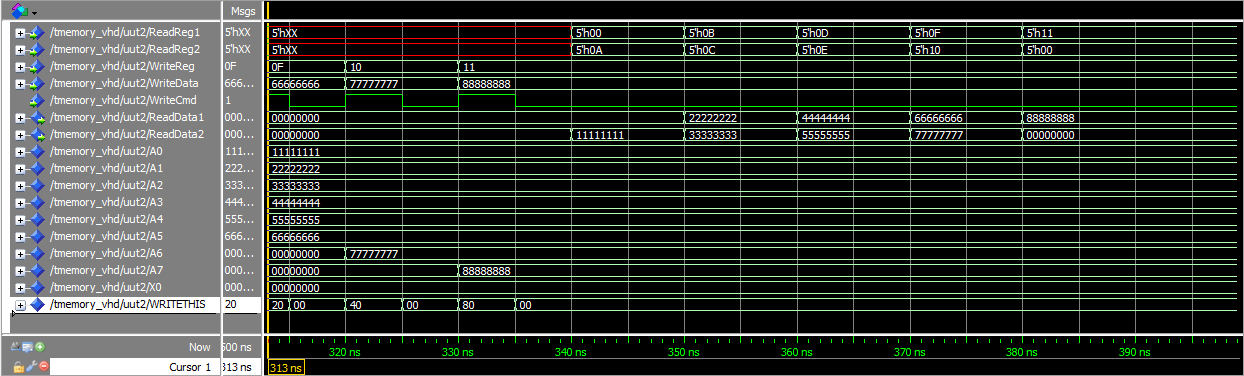
The code supports the registers A0 to A7 and the X0 register in the register Bank.

**Testing:**

From the following images you can see register A0 getting “11111111”, A1 getting “22222222”, A2 getting “22222222”, all the way through A7 getting “88888888”. The first attempt to was to assign “11111111” to X0 but X0 remains to be zero which is the main purpose of X0.



This second image shows the index of the register to read and output to ReadData1 and ReadData2.



* Link for our github:
* Link to our master branch:

<https://github.com/SU-ECEGR-2220/AVOCADOS/tree/master/Lab%205>

* Link to individual branches:

Thanh: <https://github.com/SU-ECEGR-2220/AVOCADOS/tree/Thanh/Lab%205>

Don: <https://github.com/SU-ECEGR-2220/AVOCADOS/tree/Don/Lab%205>

Lauren: <https://github.com/SU-ECEGR-2220/AVOCADOS/tree/Lauren/Lab%205>

* Note:
* Each individual built the program then saved and test on his/her own branch first. Then we came up with the best version of the program and pulled it to the master branch. You can find the contribution of each member towards the project in the individual branches. Thank you!