FAET630004: AI-Core and RISC Architecture **(Due: 4/2/20)**

Homework Assignment #1

Instructor: Chixiao Chen *Name:* XinRu Jia , *FudanID:* 20212020041

Problem 1: Implement a simple RISC Core

(2+5+8=15 points)

On class, we define an extremely simple RISC ISA and its hardware organization, which are both shown below. For this homework assignment, we need to design/implement/simulate this core.

极简指令集（RISC）Load/Store 架构			
opcode	目标 Reg	源寄存器/立即数	说明
Load	rd	rs+imm(5b)	//在 rst imm 地址→rd
Store	/	rs(地址)/rs(DATA)	//rs(DAI)→Mem index =rs(地址)
MOV	rd	imm(9b)	//赋值→rd
MAC	rd rd	rs1,rs2,funct=0 rs1,/,funct=1	//乘加 //初始赋值清除

Figure 1: RISC Instruction Set

(a) Which HDL you want to use ?

(b) For Verilog-players, please write an top-level structural verilog file to decribe the system.
For C-players, please complete a header filer and .cc file , compatible to Vivado HLS, to describe the system.
(Hint: Please submit your script as well)

(c) For Verilog-players, please write a testbench verilog file to simulate the system. It should complete a 4-MAC (neuron) computing.
For C-players, please complete a _test.cc file , compatible to Vivado HLS, to simulate the system. It should complete a 4-MAC (neuron) computing.
(Hint: Please submit your script and simulated waveform, highlight the final results and compare it with the theoretical value.)

(Solution) Implement a simple RISC Core
(a) System Verilog.

(b) In the attachment I submitted, The tb-top.v is my testbench, and top.v is the top file of the whole core mudule, including 5 subfiles for circuit sub-module. You can run testbench.v file to get the simulated waveform. I also submitted the waveform file, whose screencut at the bottom of the document.

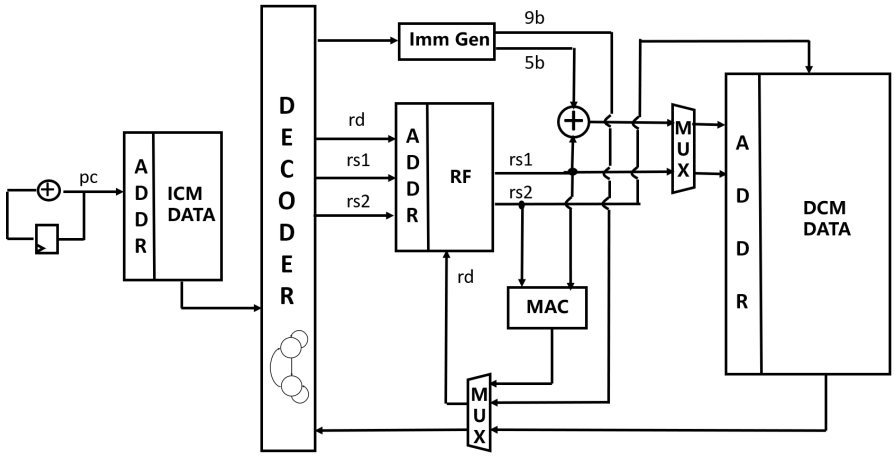

Figure 2: Architecture for scalar mac

(c)

$$\begin{bmatrix}-1 & 2 & -3 & 4\end{bmatrix} * \begin{bmatrix}-1 & -1 & 0 & 2\\ 2 & 2 & 3 & 1\\ 3 & -3 & 1 & 4\\ 4 & 4 & 2 & 5\end{bmatrix} = \begin{bmatrix}10 & 30 & 11 & 8\end{bmatrix}$$

In the testbench called tb-top.v, I designed a 4-mac Matrix to achieve. At first I stored the calculating data(int16) in dmem, and the instructions in imem at the at the initial of simulation. And part of addresses were stored in register file.
In each cycle, I loaded the data from dmem and calculated them. We also need an extra instruction to reset the psum register. At the end of cycle, the sum was restored in dmem, and the address was just we prestore in the rf(rs in assemble codes).
So we need such clk/instructions below assemble codes to complete the whole cycle. Lack for branch instruction, so we have to repeat similar instructions to calculate 4 cycles.

```
load  rd1,  rs,  1
load  rd2,  rs,  2
load  rd3,  rs,  3
load  rd4,  rs,  4
load  rd5,  rs,  5
load  rd6,  rs,  6
load  rd7,  rs,  7
load  rd8,  rs,  8
mac   rd ,  0 ,  /,  1
mac   rd ,  rd1, rd2,0
mac   rd ,  rd3, rd4,0
mac   rd ,  rd5, rd6,0
mac   rd ,  rd7, rd8,0
store /  ,  rs , rd
```

As showed in picture waveform, testdata was stored in Dmem at the beginning , after 4 cycles calculation, dmem was put in 10, 30, 11, 8 one by one (Display in hexadecimal). But We have reached the expected functional goal.

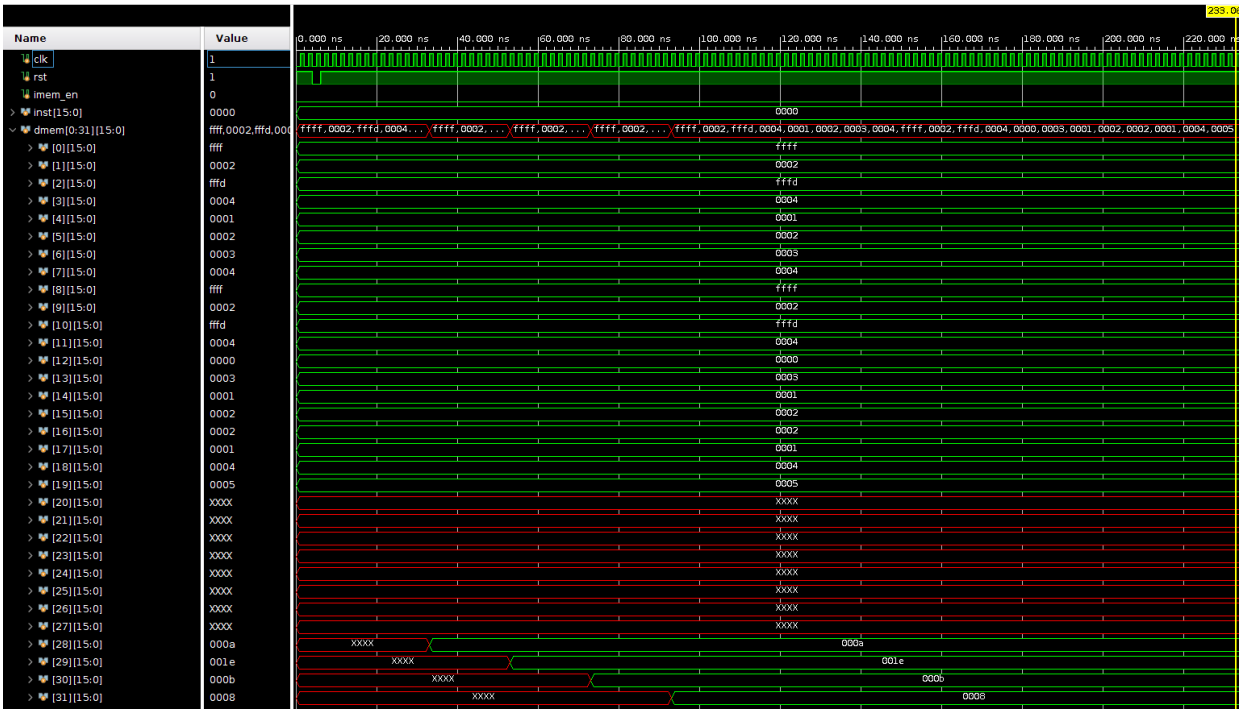

Figure 3: Waveform for Dmem