EE4321 Digital Systems Design using HDL

Final Project

Mark W. Welker

TEXAS STATE

Project: Simplistic Processing Engine

You will be creating a processor that contains a matrix ALU, a math ALU, instruction fetch, execution, and memory interface .

The matrix unit will perform matrix multiplication, scaler multiplication, subtraction, addition, and transposition.

A top module will be provided to tie all the pieces together. You MUST use the TOP Module provided and the interfaces must comply with the TOP module.

A testbench will drive the master clock and reset, a grading module will be added to make certain your outputs properly perform all tasks. Following reset, the CPU will request the first instruction from the ROM and begin execution of the program.

Project Details

The registers mentioned in this document are internal processor registers, you are free to create as many registers as necessary to complete the project. You MUST have a justification for all registers created.

Most Matrices will be 4x4 16 bit deep.

Matrix multiplier will multiply two 4x4 matrix and return an appropriate size matrix. And Multiply a 4x2 * 2 x 4 matrix and return a proper result.

Scalar multiplication is multiplying a matrix by a single number

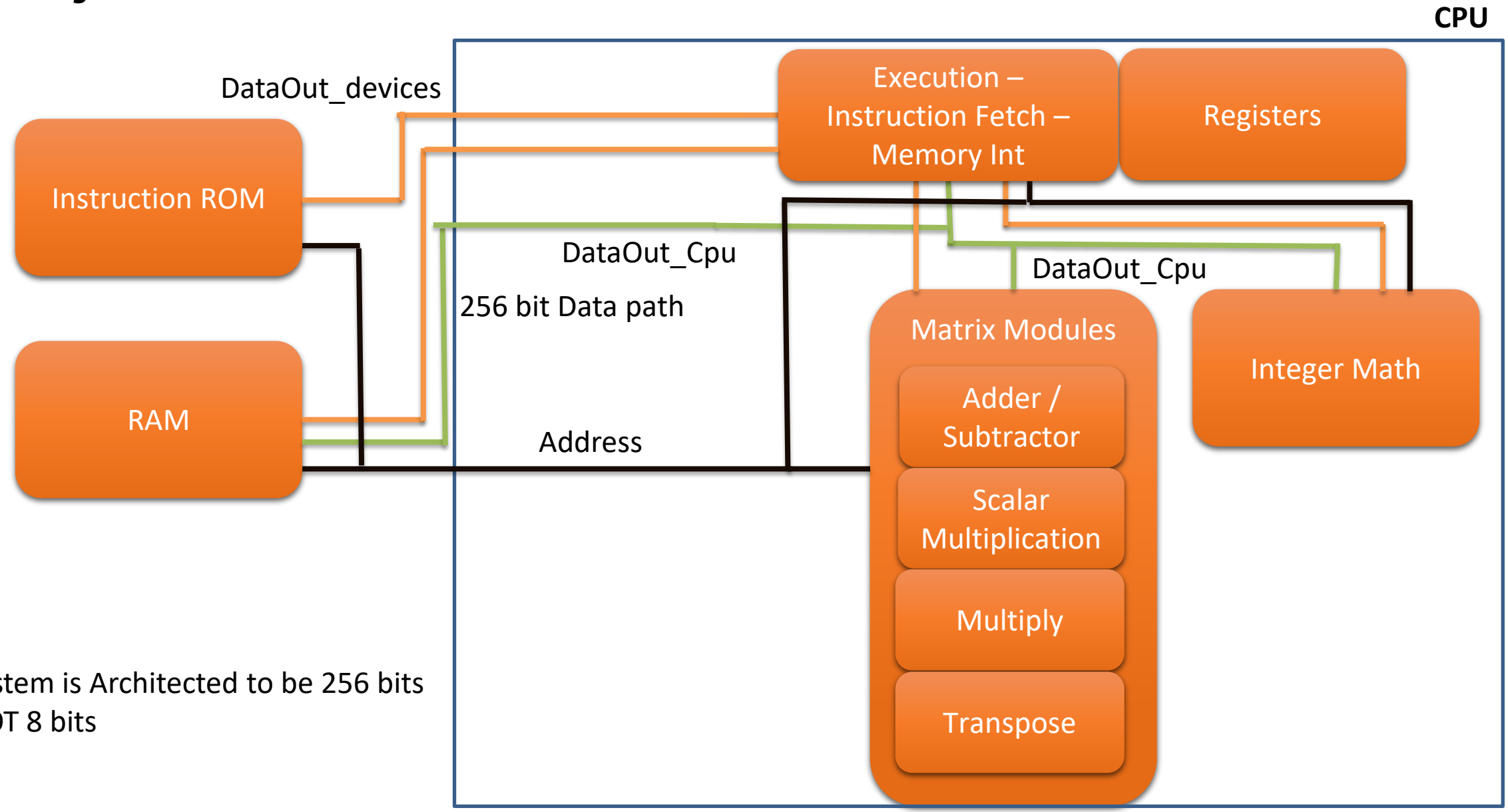
Add and Subtract will add and subtract 2 4x4 matrix

Transpose will flip a matrix along its diagonal.

If you want to brush up an Matrix Math

https://www.mathsisfun.com/algebra/matrix-introduction.html

Project: Architecture



System is Architected to be 256 bits
NOT 8 bits

TEXAS STATE

Memory organization

- Memory is to be 256 Bit width. It is 256bit aligned. The lower 7 bits are NOT on the address bus.
- Instruction memory is any width you want to utilize.
- The table on the right should be utilized for memory mapping the modules
- Address bus is 16 bits

Memory Location	Module
0000h	Main Memory
1000h	Instruction Memory
2000h	Matrix ALU
3000h	Integer ALU
4000h	Internal Register block
5000h	Execution Engine

Module Address				Module Offest											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Main Memory organization

- Memory is to be 256 Bit width. It is 256bit aligned. The lower 7 bits are NOT on the address bus.
  - Matrix 1 is at address 0
  - Matrix 2 is at address 1
- The table on the right should be utilized for memory mapping

Memory Location	Information	Memory Location	Information
0000h	Matrix 1	000Ah	Integer Data1
0001h	Matrix 2	000Bh	Integer Data2
0002h	Result 1	000Ch	Integer results
0003h	Result 2	000Dh	Integer results
0004h	Result 3		
0005h	Result 4		
0006h	Result 5		
0007h	Result 6		
0008h	Result 7		
0009h	Result 8		

Project Details: How it operates

- The test bench will start the system clock and toggle reset
- Execution engine will fetch the first opcode from instruction memory and begin execution.
- The execution engine will direct the transfer of data between the memory, the appropriate matrix modules and memory.
- The execution engine will continue executing programs until is finds a **STOP** opcode.
- Matrix ALU and Integer ALU will **NOT** require overflow.

Project Instruction Set

- The following are instructions and their opcodes that can be used for this project. You are welcome to add more instructions for your project.
- The instructions are 32 bits in length each of the fields; opcode, destination,source1,source2 are 8 bits

Instruction	Opcode	Destination	Source1	Source2
Stop	FFh	00	00	00
MMult1	00h	Reg/mem	Reg/mem	Reg/mem
MMult2	01h	Reg/mem	Reg/mem	Reg/mem
MMult2	02h	Reg/mem	Reg/mem	Reg/mem
Madd	03h	Reg/mem	Reg/mem	Reg/mem
Msub	04h	Reg/mem	Reg/mem	Reg/mem
Mtranspose	05h	Reg/mem	Reg/mem	Reg/mem
MScale	06h	Reg/mem	Reg/mem	Reg/mem
MScaleImm	07h	Reg/mem	Reg/mem	Immediate
IntAdd	10h	Reg/mem	Reg/mem	Reg/mem
IntSub	11h	Reg/mem	Reg/mem	Reg/mem
IntMult	12h	Reg/mem	Reg/mem	Reg/mem
IntDiv	13h	Reg/mem	Reg/mem	Reg/mem

TEXAS STATE

Project: Test Bench operation

Matrix Operations

1. Add the first matrix to the second matrix and store the result in memory.
2. Add the 16-bit numbers at memory location 0x0a to location 0x0b store in a temporary register
3. Subtract the first matrix from the result in step 1 and store the result somewhere else in memory.
4. Transpose the result from step 1 store in memory
5. Scale the result in step 3 by the result from step 2 store in a temporary register
6. Multiply the result from step 4 (4x4) by the result in step 3 (4x4), store in memory.
7. Multiply the result from step 6 (4x2) by the result in Step 3 (2x4). Store in memory.
8. Multiply the result from step 5 (2x4) by the result from step 4 (4x2) Store in memory

Integer operations (Operations at memory location 0 and 1 are assumed to use ONLY the LSW (first 16 bits) of the 256 data located at 0 and 1.

9. Multiply the integer value in memory location 0 to location 1. Store it in memory location 0x0A
10. Subtract the integer value in memory location 01 from memory location 0x0A and store it in a register
11. Divide the result from step 8 by the result from step 9 and store it in location 0x0B

Testing

A top module and testbench will be created to grade your modules.

The top module will be provided during the semester to make certain your interface IO are the same as the testbench. The top module will define the interfaces between the modules defined on slide 4.

As part of the final grading your code will be executed with a testbench that will include the instructions on slide 8 of this presentation. The order of the instruction, MAY change in order to determine if you wrote the code properly, or faked the output with the correct answers at the correct time.

Project Input Data

Matrix 1

8	12	8	6
12	16	13	9
10	9	5	13
12	3	10	6

Matrix 2

3	4	7	8
7	8	14	7
16	9	12	11
12	5	5	6

Memory Location

8 = 06h

9 = 0Dh

TEXAS STATE

What you need to turn in

Documentation showing and explaining your state machine for the execution unit

Output from the final matrix's. transcript if you used $display to output the matrix values.

Waveforms : output waveform from each individual operation.

Use appropriate radix to allow the display to become readable

All the code from your project

Your code must be documented with comments

Your Code MUST Compile

Rubric

Final Project Criteria	Minimal Effort			Mastered the criteria	
Can I determine you will finish	5	10	15	20	
Did you include wave forms	5	10	15	20	
Does it Compile	5	10	15	20	
Is code well written: Proper System Verilog: Well Commented	4	6	8	10	
Does it Perform ALL tasks	5	15	25	30	
	24	51	78	100	totals