**Lab 5: 32-bit Single-cycle Datapath and Controller Design**

**Overview:** In a series of laboratory exercises you will incrementally implement a 32-bit processor that supports a basic set of operations.

**Lab 5 schedule overview**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Score** | **Start date** | **Due date** |
| Task 0 | 125 points | Week of  Oct 22 – Oct 26 | The first 60 minutes of your lab session during the week of Oct 29 – Nov 2, 2018 |
| Pre-lab of Task 1 |  | Week of  Oct 29 – Nov 2 | The first 30 minutes of your lab session during the week of Nov 5- Nov 9, 2018 |
| Task 1 | 325 points | As soon as you finish Task 0 | By 4.45 pm on Friday Nov 16th, 2018 |
| Task 2 | 150 points | As soon as you finish Task 1 | By 4.45 pm on Thursday Nov 29th, 2018 |
| **Task 3** | **100 points** | **As soon as you finish Task 2** | **By 11.45 am on Wednesday May 2nd, 2018** |
| **Total** | **700 points** |  |  |

**Lab 5-Task 3 (100 pts):**

**Demo:** on the FPGA board.

**Due date:** see the table above

**References:**

* Lecture Notes: Verilog ppt slides and Verilog chapter on Zybook
* “single\_cycle\_datapth\_Task1” on D2L under Lab 5
* “single\_cycle\_datapth\_Task2” on D2L under Lab 5

**Objectives:**

* Use the single-cycle processor to execute a specified application.
* Implement the processor on the FPGA board

**Assignment for Lab 5 Task 3**

We will now use the 32-bit processor to execute a matrix multiplication code for multiplying the square matrices of size 4x4.

1. Get the zipped folder “task3\_matrix\_multiplication\_files” under Lab 5 from D2L.
2. Copy the contents of “matrix\_mult\_**instruction**\_memory\_init\_hex.txt” into your initialization (initial begin … end) block in the **instruction** memory.
3. Copy the contents of “matrix\_mult\_**data**\_memory\_init\_hex.txt” should be copied into your initialization block in the **data** memory.
4. **Modify the code in DataMemory.v file as follows:**

reg [31:0] memory [0:1023]; reg [31:0] memory [0:**16383**];

always @(posedge Clk) begin

if (MemWrite == 1'b1) begin

memory[Address[**15**:2]] <= WriteData;

end

end

always @(\*) begin

if (MemRead == 1'b1) begin

ReadData <= memory[Address[**15**:2]];

end

else

ReadData <= 32'h0;

end

always @(posedge Clk) begin

if (MemWrite == 1'b1) begin

memory[Address[**11**:2]] <= WriteData;

end

end

always @(\*) begin

if (MemRead == 1'b1) begin

ReadData <= memory[Address[**11**:2]];

end

else

ReadData <= 32'h0;

end

5) **Simulation** and **Download to FPGA board for 4x4 matrix multiplication**

a) (20 points) Verification will be based on the post-synthesis simulation. Display the contents of the register number 23 from the register file on the waveform. This register RegFile[23] will display the final value in each cell of the product matrix during the multiplication process. See Functional Verification section below.

Note: 50% point deduction for this part if only work in behavioral simulation.

b) (80 points) After part a) is successful, display the contents of RegFile[23] on the FPGA board. You have to figure out a way to send the value of RegFile[23] to the 7-segment display on FPGA board whenever RegFile[23] is updated!

Hints:

* TwoDigitDisplay used in previous labs should be modified to display eight digits on the board. Reminder: your 7-segment code is used in TwoDigitDisplay
* ClkDiv can be used to ensure that you can see RegFile[23] when it is updated.

**Functional Verification:**

* **First line** in the “matrix\_mult\_data\_memory\_init\_hex.txt” indicates the **dimension of the matrix**. Currently it is set to 4 indicting that two matrices (A and B) of 4x4 will be multiplied.
* In the folder named “matrix\_multiplication\_files” 4x4 inputs are provided in 2D format (A\_input\_4x4, B\_input\_4x4) for your convenience to test your code with the expected result in “C\_output\_4x4.txt file”. To verify the correct functionality you need to confirm that the updates to the Register[23] are matching with the “C\_output\_4x4.txt”.
* Part of waveform for 4x4 matrix multiplication: A\*B = C is shown on the next page.

In the folder named “matrix\_multiplication\_files, you will see the given matrices A, B and the results C are given below.

A

**243 175 214 192**

142 35 191 248

37 166 205 32

90 118 12 250

C

**94099** 113445 47839 100162

96626 71435 30846 83513

48913 89588 23297 51067

66556 39596 17358 45178

B

**28** 81 107 146

**27** 208 38 26

**175** 251 62 185

**235** 19 10 107

=

\*

C(1,1) = **94099** = 243\*28 + 175\*27 + 214\*175 + 192\*235

= 6804 + 4725 + 37450 + 45120

Observe that RegFile[19] has elements of A, RegFile[20] has elements of B,

RegFile[21] has a product, RegFile[22] has accumulated sum and RegFile[23] has element of C

**When demo to TA (to get full score for post-synthesis simulation), your simulation (RegFile[23]) MUST show every element of matrix C. The last element, C(4,4), is 45178.**

