Midterm design report requirements

1 Turning In Your Report

This report is due by *Friday 11:59pm* in form of .pdf file. There is also a LaTex Template provided on the course web. Feel free to use it or not as long as you submitting a .pdf file. To submit your report you must use the command at where your report file is: *submit -p midtermreport*

2 Overview

The purpose of the midterm report is to provide an overview of what you have implemented for the single cycle and pipeline design. The report should be written so that a technically competent reader can read it quickly and understand what you accomplished. Detail is expected but at a level that can be digested with a single read.

The final report, which will be due closer to the end of the semester, will be very similar in format to this report and will include some of what you are going to do for this report. Therefore doing a good job on this report will save you some time on your final report.

3 Formatting

- The overall midterm report should be 9 or 12 pages in length including cover page, executive summary.
- Reports longer than 12 pages or shorter than 9 pages will be penalized. (appendixes do not count as part of the report).

- Three pages total from 12 pages are allocated for both the single cycle(2 pages) and pipeline (1 page) block diagrams (hand drawn and scanned will not be accepted, nor will copies of the diagram in the book, you can use software tool like visio or powerpoint to draw your diagram, but fully software generated diagram will not be acceptted.)
- Do not include source code in the text except possibly for short code fragments to explain parts of your design. If longer code segments are required include them in an appendix.
- Text should be 12 point font.
- Line spacing should be 1.5 lines.

4 Report Sections

4.1 Cover Page

Title
Course number and name
Teaching assistant name
Due date
Author

4.2 Executive Overview

This section should be about two paragraphs in length on a page by itself.

Give a short overview of what you have implemented. Mention what was or was not successful. Mention any insights into processor design you gained from designing the processors. Finally, give a brief outline of what is in the rest of the report.

4.3 Processor Design

In this section you will discuss your processors and the design decisions you made while implementing your processors. Also discuss the differences between the single cycle and the pipeline processors that influenced your designs. There could be performance reasons for design choices as well as ease of implementation. Block diagrams should be included for the pipeline and

the singlecycle. Two block diagrams should be included for the singlecycle one for each person in the group. When discussing the singlecycle each person should discuss this section individually, thus there will be two singlecycle discussions and one pipeline discussion.

4.4 Processor Debugging

In order to demonstrate the techniques you used to debug your processor assume the code and the resulting memout.hex output in Table 1. Assume you are debugging a pipelined processor, and that the processor is the only component that is in question, the memory works perfectly. First identify what is wrong with the output. Second identify two possible hardware design errors (as specific as possible) that each alone would result in the memout.hex file. Third for each error, describe how the error propagates into the memout.hex file (which components does the error pass through). Fourth describe what you would look for in your waveforms to distinguish the design errors. It is not enough to just say the instruction does not work, you should instead give a description of what is implemented incorrectly that causes the instruction to fail, and then say how that failure results in the provided memout.hex file. Finally you must describe what tests you could perform to distinguish which of the two bugs that you identified is actually causing the faulty memout.hex file.

4.5 Results

This is where you will compare the designs you implemented. Prepare a table for each processor summarizing: (you can use the programs from lab as your benchmarks for calculation)

- The estimated frequency of your design.
- The average instructions per clock cycle.
- The average latency of one instruction(how long one instruction takes to complete).
- The performance of the design in MIPS.
- FPGA resources required for your design.

Table 1: Debug files.

	asm file		memout.hex
	org	0x0000	:0400000034010100C6
start:			:0400010034020001C4
	ori	$1, $0, 256	:0400020014020001E3
	ori	$2, $0, 0x01	:04000300FFFFFFFFFD
	bne	$2, $0, foobar2000	:040004003C04BEEF0B
	halt		:040005003885DEADAF
foobar2000:			:04000600AC25002CF9
	lui	$4, 0xBEEF	:04000700FFFFFFFFF9
	xori	$5, $4, 0xDEAD	:00000001FF
	sw	$5, 44($1)	
	halt		

Include a short explanation of your designs before each table. Discuss the meaning of the numbers in the table and how you derived them. Check compile.log to get the FPGA resources for your design. The FPGA resources should be reported in area (not percentage), and number of registers. Area is given in terms of FPGA logic blocks. A description of the critical path for each of your processors should be discussed. Also say were the critical path begins and ends, as well as what portion of the processor contains all or most of the critical path. Discuss if the critical path was where you expected it to be, and explain, only saying yes it was, or no it was not is not acceptable.

Compare the performance of your processor designs. Run the asm files (search,fib,mult) on the synthesized version of your processors. Graph the total execution time of each program for each of the following time periods of 100ns/50ns/20ns/10ns/5ns/2ns. Discuss the trends you observe and what would the execution time be if the clock frequency could increase to infinity.

You should have the MIPS number for each processor, use it to discuss and compare the performance of your designs. Did the designs perform as you expected, discuss any interesting insights you discover in comparing the designs. Which design performed best and why?

If any of your processor designs did not work well enough to get experimental data, please estimate the results you could not obtain using assumed but realistic data. You will need to determine the number of cycles required

to complete each time of instruction (latency). For each test case (fib, search, mult) estimate the number of cycles that would be required to complete each program, put this in a table. Your CPI should be greater than 1 and your IPC must be less than 1. Compare these computed throughput to your processors throughput.

4.6 Conclusions

This section should be two paragraphs. If your designs worked, summarize your results and accomplishments. If your designs did not work, say what you could have done to get them to work and what you did accomplish. Finally discuss what you have learned about the advantages and disadvantages of each design.

4.7 Bonus

Read the C to MIPS GCC Cross Compilter Tutorial on the course website. Try to code a standard C Program for sort. It can be any sort algorithm, bubble sort, insertion sort, quick sort, etc. But try to keep your code simple. You can try to run your C program by compiling it under standard GCC(not the one we provide), which should be available in most ECN machines. If you have less knowledge on how to use GCC, you can just do "gcc yourcode.c" and then it will produce a file called a.out, you will just have to run "./a.out" in terminal to see the result of C program execution. After you confirm that your C program is correct, then you can follow our tutorial to use the cross GCC compiler(we provide, not the standard one) to compile your C program to MIPS assembly. You will find that you need to make some manual changes on produced assembly to make it work in our asm and sim because some of the instructions that compiler produced will not be supported by our simulators. However, it should not be too much to modify if you keep your original C program simple.

The sample data set to sort is also provided on the course web. and you should be able to access the cross compiler via "mips-cc" in your terminal just like asm and sim. The usage of the cross compiler is "mips-cc yourcode.c", then at the same location a file called "yourcode.asm" will be generated by the compiler. The file may also say which line you have syntax error or instruction are not supported by our sim. You just have to manually fix those since the processor we designed in this course is not fully implemented

as commerical product. After all, the purpose of using this compiler is to make your life easier if you want to code a meaningful assembly program like you will do in lab 10. By help from compiler, you can now easily generate assembly code structure or port algorithm to assembly.

The Bonus points for this section is 5 points for correctness of the C program execution. 10 points for producing a sort assembly that is fully supported by our instruction set in the lab. And 5 points for correctness of the assembly execution using asm and sim, plus 5 points for correctness of the assembly execution in your hardware with your pipeline processor. You can demo the execution to your TA during Midterm Report Week(any lab section or office hours). And include both your C program and final assembly in your report paper. The total points for this bonus section is 25.

5 Grading

0 pts	**Cover page (5 point penalty if it does not meet specification)**
5 pts	**Executive overview**
15 pts	**Processor design**
15 pts	**Processor debug**
	12 pts for describing errors (6 pts each)
	3 pts for distinguishing
20 pts	**Results**
	6 pts Graphs and explanations
	2 pts critical path
	2 pts processor frequencies
	3 pts throughput
	2 pts resources
	5 pts discussions
10 pts	**Conclusions**
	4 pts summary
	3 pts lessons learned
	3 pts uses for processors
0 pts	**Length of paper (5 point penalty for violating length specification)**
10 pts	**TA subjective evaluation for overall report quality**
75 pts	**Total Report Grade**
25 pts	**Report Bonus, to be recorded seperately**