**Project Deadlines and Demonstration Procedure**

There are three phases. First phase is the verification of pipelined datapath with forwarding and hazard detection. Second phase is the execution of the vbsme.s on your datapath. Third phase is the completion.

* **Phase 1 Deadline: November 28 at 2pm. Signup for a demo slot Nov 28 or 29**
* **Phase 2 Deadline: December 3, at 2pm. Demo during your lab session**
* **Competition Deadline: December 5, at 2pm. Demo during your lab session**

**Phase 1: (2000 points)**

* **Implement forwarding in ID, EXE, MEM stages**

Incrementally add forwarding units, show that each case is resolved with forwarding

Write a short program for each dependency case and test individually

* **Implement Hazard detection**

Identify all cases that require hazard detection (cases where forwarding cannot resolve the dependency) and implement.

* **Demonstration:**
  + Submit all .v files on D2L “Project-Phase1” folder.
  + Functional verification based on post-routing simulation.
  + Your instruction memory must read from input file named "Instruction\_memory.txt".
    - Inside initial begin block of Instruction\_memory.v, use following initialization method:
      * $readmemh ("Instruction\_memory.txt", <memory register identifiers>)
  + Public test cases: project\_public\_test\_cases.s
    - Note that these public test cases cover only a subset of dependency scenarios.
  + Instruction\_memory.txt for those test cases are included in the “Project” folder.
    - File is generated by MIPS Helper and branch offset bug is fixed in this file
  + Before the demonstration you will be given the instruction memory ("**Instruction\_memory.txt**") with the private test cases.
    - We highly encourage you to work with other teams and share your test cases. Students benefited from this collaboration significantly in the past.
  + You will generate the post-routing simulation using this instruction memory
  + **Waveform must show the following signals:** Reg\_writedata, pcresult, Hireg and Loreg as outputs to top module.
  + TAs will check your waveforms and will let you know about the failing cases.
  + Teams will have only one chance to re-demo with 75% of the lost points to recover for each case.
  + We will use public test-cases only if most of the private test cases fail.
  + Each demo will be conducted for 20 mins on Nov 28 or Nov 29. You will have to select a slot for the demo. Please select one and insert comment on the demo slot which is suitable for all of your team members. Slots are available on first come first serve basis. Put your "d2l GROUP NUMBER" as comment in your slot. We will update sheet regularly.  Avoid selecting already booked slot. Sign up sheet is at: <https://docs.google.com/spreadsheets/d/1fhdYJUleE97OxbHKipPMck0GMNAVKy22rZ5QSEnlXYo/edit?ts=59ff665f#gid=0>
  + All team members are required to attend the demo.
  + During the demo, **TA will log onto your system and download the group submission** from d2l. Then TA will take over the control of the system and will copy the instruction and data memory file from his/her flashdrive into newly created project, and will start the synthesis and implementation.
  + While the implementation is on-going, TA will ask questions about your implementation and details of pipelining to the team members and will take notes.
  + Once post-implementation simulation is ready, TA will check their waveform
  + After checking waveforms TA will explain what kind of forwarding or hazards are not taken care by the teams, if there is any. TA won’t be sharing actual testcases.
* **Offline Testing:**
  + Functionality of individual instructions on the pipelined datapath will be tested offline based on post-routing simulation. Your datapath should still be able to execute all the instructions from Lab9-18.
* **Deliverable:**
  + Submit the following files
    - All verilog files: "\*.v"
  + Include the following notes under comment section during your submission
    - % effort
    - Number of pipeline stages:
      * We accept both four and five stage based pipelined datapaths
      * Indicate the number of pipeline stages in your design.
    - Branch decision and resolution stage:
      * Indicate the stage (DE or EX or MEM) where you are making branch decision.
    - If you don't include any note, we will **assume** that it is a 5-stage pipeline and branches are resolved during ID stage during the offline testing.
* **Penalty Conditions:**
  + private test cases on the pipelined datapath (2000 points)
    - If majority of the private test cases fail, then you can use public test cases for a maximum score of 800 points
  + Percent effort not reported (15% penalty)
  + Late submission (10% per day)
  + Submitting files in a folder or in compressed form (zip/tar). (20% penalty)
  + Changing the file name or extension. (20% penalty)
  + Failing to demonstrate (80% penalty)
  + Design works in behavioral simulation but fails to synthesize (70% penalty)
  + Design works in behavioral simulation, synthesizes with warnings but post-routing simulation fails (60% penalty)
  + Both team members must attend the demonstration
    - Unable to answer questions about your implementation during demo - 50% penalty
    - Missing demonstration – 80% penalty
  + Each unattended lab without notice 50pts penalty.

**Phase 2: (500 points)**

* **Objective:**
  + Execute your vbsme on the pipelined datapath
* **Eligibility**
  + At least 75% of the instructions are functional on the datapath
  + At least 75% of the private test cases for pipelining is passing
* **Deliverable:**
  + Submit the following files under “Project-Phase2” folder
    - All verilog files: "\*.v"
    - vbsme.s
* **Method**
  + A test case will be given at the beginning of the lab for your data memory
  + Prepare your instruction memory before the lab with your SAD routine
  + Prepare data memory with the new test case
  + Execute the program on the FPGA
  + The (X,Y) coordinates of the block of the current minimum SAD should be displayed on the FPGA
  + Display will start with (0,0) and each time a block with smaller SAD is found new coordinates should be displayed
  + Initialization part for the data memory will then be given to you by the **TA or ULA**
  + Set up behavioral simulation showing
    - **PC, Current minimum, X, and Y values**
  + TA will check both the FPGA and simulation
  + TA/ULA will collect your demonstration version of the datapath (.v) files and your vbsme.s
  + Offline validation to be conducted by TAs
* **Deliverable**:
  + Submit the following files
    - All verilog files: "\*.v"
    - Data file used for initializing data and instruction memory (\*.txt)
    - .s (final form of your sad routine)
* **Penalty Conditions:**
  + Percent effort not reported (100 pt penalty)
  + Late submission (10% per day)
  + Submitting files in a folder or in compressed form (zip/tar). (20% penalty)
  + Changing the file name or extension. (20% penalty)
  + Failing to demonstrate (80% penalty)
  + Design works in behavioral simulation but fails to synthesize (70% penalty)
  + Design works in behavioral simulation, synthesizes with warnings but post-routing simulation fails (60% penalty)
  + Design works in post-routing simulation, but FPGA fails to display (25% penalty)
  + Both team members must attend the demonstration
    - Unable to answer questions about your implementation during demo - 50% penalty
    - Missing demonstration – 80% penalty

**Competition**

* Qualification for the competition
  + At least 75% of the instructions are functional on the single cycle datapath
  + Pipelined design passes at least 75% of the private test cases
  + vbsme executes on the pipelined datapath on the FPGA
* No pipelining demo will be taken.
  + Only competition
* Your files datapath (.v) and vbsme (.s) must be on D2L (**“Competition” folder**)
* You will then be given the initialization part for your data memory
  + Test case will be 64x64, 4x4
* Prepare behavioral simulation showing PC, Current minimum, X, and Y values
* Synthesize the design and prepare your post-routing resource utilization
* Program the FPGA
* Call your TA
  + Show that FPGA sequences through the coordinates and finally displays the correct X,Y coordinate
* Show your behavioral simulation waveform
  + TA will collect cycle count data (**A**): number of cycles it takes to run the program
* Show the post-routing data:
  + TA will collect Critical path delay(**B**) , BRAM usage(**C**), LUT usage(**D**), DSP usage(**E**)
* Results for A,B,C,D,E will be tabulated
  + Winners will be announced by the end of the last lab session.

Questions From Previous Years

Q1. If you need to use stack for your SAD routing:

Stack will be implemented in data memory. On a Reset, you can initialize  $sp to the last location in the data memory.

Q2. JAL

JAL implementation and register update

For project, while implementing JAL do not add additional write port to MIPS register file (32 register file). JAL should update register $ra in the write-back stage only.

Q3. Test Cases

Sharing testcases

I was wondering if it would be alright if each group wrote their own test cases and stick them in a huge repository (git or google docs). It would help out in trying to cover the multitude of cases that can be present for this part of the lab.

We highly encourage sharing test cases.

Q4. reading instruction\_memory.txt

I placed my .txt file for instruction memory in the simulation sources folder as suggested in another post, but I still can't seem to get vivado to find it. Anyone else having this issue?

You should put this in to design sources folder. Or try adding it into project in the vivado. You will have to follow similar steps that you use for adding RTL file.

In Vivado:

Add Sources --> Add or Create Simulation Sources ---> Change file type to "All Files" --> Select "Instruction\_memory.txt"

Also, you may have to right click on the text file and change the type of file it is to "Data" before it can be read by your instruction memory. I had that issue even after I loaded the file into the project.

Q5. MIPS Helper

Creating Instruction memory to be read by readmemh

I can't seem to get the mipshelper to output hex instructions without comments. I really don;t want to go through and delete each line, so has anyone figured out how to get it to output just hex similar to "Instruction\_memory.txt" that was given to us on d2l?

You can use following commnad:

1) Run MipsHelper.

./mipsHelper369 -aiohd input.s output.txt

2) On ECE node open the text file in "VI" editor by typing "vi output.txt" on the shell then hit enter. It will open the file in VI editor.

in VI editor just type this ":%s/ ->.\*$//g" and hit enter. This will do the required changes. Save file by typing ":wq!" and then hit enter.

FQ6. FPGA Board

Output to the Board

What exactly will be the required output to the FPGA board per cycle?

For SAD on FPGA, it should be the registers tracking the coordinates (X,y) for the current minimum SAD location during the execution.

Q7. What about when we reach the end of the program? Should we put an infinite loop with a nop inside at the end of our program or something, so that we don't keep fetching more instructions?

Implement an indefinite loop like the following:

here: j here

Q8. Clock Divider

Is a Clock divider module necessary? I saw in the provided xdc file the following line:

create\_clock -period 100.000 -name Clk -waveform {0.000 5.000} [get\_ports Clk]

would changing the values in this line in the xdc essentially function to adjust the clock speed to the required setting? If so, which values would I have to change, and which values do what?

xdc file is not the place to create the clock divider. The xdc file is used for applying design constraint. Above clock command is used by the compiler as follows:

create\_clock : it indicates the signal named as "Clk" in top module is a clock signal.

-period: With this input, compiler will try to synthesize and route design on FPGA with maximum critical path of 100ns. It indicates the maximum operating frequency for your design (1/period)Hz. For competition you will have to play with this parameter to determine the minimum critical path acceptable for your design. Making it too small will give you incorrect post-implementation behavior.

-waveform: It indicates the "Clk" duty cycle. In above case it indicates that if clock starts at 0 ns then positive edge will occur at 0ns and next negedge on clock will occur on 5ns. You can change 5ns to 50ns to indicate 50% duty cycle but it wont affect your critical path. critical path will be dependent on "-period".

You will have to write a RTL module to create a clock divider which will use above clock as a reference clock input.

Q9. Instruction Memory

instruction\_memory.txt wrong

In the instruction\_memory.txt line 18 is supposed to be the - la $s2, asize1 - instruction on line 45 of project\_public\_testcases. We know that the la instruction is the equivalent of lui. However, if you convert the hex instruction on line 18 to binary and compare the opcode to what lui is supposed to be, it is different. The opcode turns out to be the opcode for an ori instruction, which causes unexpected results. Is this an error with the file that was given to us?

In qtspim, we can not access data memory at address location 0 as initial address space is reserved for system specific code so we need to use "la". In your HDL implementation, memory location 0 is accessible to you. So mipsHelper will take the starting memory address as location 0 and will convert "la" to "ori" to load correct memory address.

Q10 could not open $readmem data file "Instruction\_memory.txt"

[Synth 8-4445] could not open $readmem data file 'Instruction\_memory.txt'; please make sure the file is added to project and has read permission, ignoring ["C:/Users/ryanr/Downloads/InstrMem.v":923]

I'm getting this error after implementation, although the Instruction\_memory.txt is in simulation sources and is a memory initilization file. Does anyone know the fix?

It has to be added as a design source as well for it to work for synthesizing and implementation.

EDIT: you also might have to right click on the memory files after they're added to the project and set the type as memory initialization files.

Q11. Data Memory

Phase 1 Data Memory

The project requirements state we only need to read from the .txt file for the instruction memory. What should data memory be initialized to for the set of private test cases we are getting for the phase 1 test? Do we leave the data memory as it is from the public tests, Set it to all zeroes, or does it also need to read from some data\_memory.txt etc. ?

You will be reading data memory from a memory file. Name it as "data\_memory.txt".

In the public testcases .s file given on D2L, the data memory comes from the lines at the top of the file before '.text'. You can use MipsHelper to convert those lines into a datamemory.txt file by running:

./mipsHelper -aiodh ./project\_public\_testcases.s ./public\_out.txt

This will produce two files, 'public\_out.txt' and 'public\_out\_data.txt'. Since the instruction memory is already given to us in the proper format, you can ignore the first. The second will essentially be a Verilog file that you could copy into the data memory module. However, in order to make it readable, (it should be fairly short anyways) you should take each line and replace 'memory[n] = 32'h100' (for example) with the plain hexadecimal number, 00000064.

Q12. initialize Data memory

Does anyone know what value to initialize the data memory in order to get the list to load, mentioned in the public test cases?

.data

asize0: .word 100, 200, 300, 400, 500, 600

asize1: .word 700, 800, 900, 1000, 1100, 1200

.text

If you are looking to create the file for data Memory you can use the mipsHelper with the '-d' command and it will create a data memory file in addition to the instruction memory file, for the demo you will need to read from a file so I recommend that way. If you want to hard code a few values the memory of asize0 starts with position 0, and increments up from there. So it would look like: memory[0] = 100; memory[1] = 200; etc and asize1 starts at position 7.

Q13. Timing Analysis

I am currently trying to run the timing analysis of our datapath and I am following the outline provided in the Verilog folder in D2L and it is showing a Clock in step 8). Is that clock from the Clock Divider in our project? I don't seem to have that in my timing analysis which makes me believe that I currently don't have an external clock running my project.

You can put same timing constraints on both the clock, input clock to your wrapper and output of clock divider (input to your datapath). But only input clock to wrapper should be assigned a pin number on the FPGA.

Q14. Issue with $readmemh and Instruction Memory

We're trying to read Instruction\_memory.txt into our InstructionMemory module, but we keep running into this error, no matter how we add the text file to our project.

"could not open $readmem data file 'Instruction\_memory.txt"; please make sure the file is added to project and has read permission"

We've already added instruction\_memory.txt to the simulation sources, but the same error still shows. Has anyone else run into this problem?

download file> add sources>add design source>(find the file)> all files(dropdown)> finish>(at this point the file should be in a folder in your project called text files)> open that folder> right click file> setfile type> memory initialization file

Q15. Question on $readmemh

The notation given to us was "$readmemh ("Instruction\_memory.txt", <memory register identifiers>)" and some lines of code in the text file are:

3c120000

8e520000

3c130000

8e730004

02538820

Which are hexadecimal values. Does that mean $readmemh ONLY accepts hex values? I've been trying to put my own binary values such as:

00100000000100000000000000000001

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00100000000100010000000000000001

to be read but it isn't processing correctly. Do I simply need to convert these binary values to hex values for it to be read by $readmemh correctly? I would think $readmemh would be able to read the value in the text value no matter what notation it's in.

You can use readmemb.

readmemh: to read hexadecimal

readmemb: to read binary

Also, you can use the h flag in mipshelper to have the output be in hex.

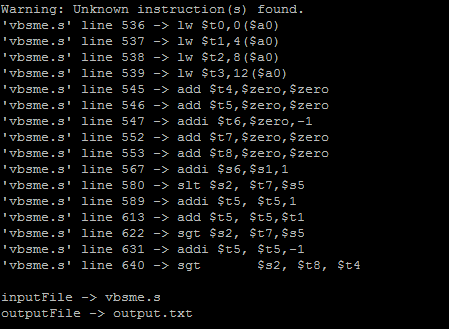
Q16. Project public test case wrong?

When I get to case 3 (write after write) case, I get very different numbers than what are listed in the file. Case 1 and 2 work perfectly, and case 3 instruction 1 and 2 work fine, but the ori messes up. I've come to the conclusion that I've either gone crazy, or something is wrong in the file. The ORI instruction uses 0xaaaa as the immediate field, and I searched for this value in hex, and decimal, and couldn't find it in my entire datapath... it doesn't exist. The instruction is correct in my memory. Anyone else run into this issue?

Andi ori and xori all are zero extended and not sign extended.

Q17. Mips helper giving errors on LW

When trying to convert our VBSME I am getting unknown instructions on known instructions. looking for advice.



you need to insert space in between registers for each instruction.

for example instead of : lw $t0,0($a0) it should be: lw $t0, 0($a0). pay attention to space between $t0 and 0($a0),