Harris

Introduction to CMOS VLSI Design (E158)

Lab 5: Microprocessor Assembly

In this final lab, you will assemble and simulate your entire MIPS microprocessor! You will first put together the top-level schematic and simulate that it executes MIPS instructions properly. You will then assemble the layout and check that it is clean and matches the schematics. Finally, you will generate a pad frame to connect your processor to the external world and verify that your system works in the pad frame.

1. Top-Level Schematic Simulation

Copy your lab4_xx library to lab5_xx for this lab. Look at the top-level mips cell that includes your datapath, alucontrol, and controller icons. The system has the following exported inputs and outputs:

Inputs	Outputs
ph1	adr[7:0]
ph2	writedata[7:0]
reset	memread
memdata[7:0]	memwrite

Table 1: MIPS Processor Inputs & Outputs

Notice that the internal wires and icons are also named to simplify debugging.

To demonstrate basic functionality of your microprocessor, you will simulate the processor running a very simple program:

```
addi  $1, $0, 43   # Register 1 <- 43     instr: 20010043
addi  $2, $0, 1A   # Register 2 <- 1A     instr: 2002001A
or    $3, $1, $2   # Register 3 <- 5B     instr: 00221825
sb    $3, 38($2)   # Mem[52] <- 5B        instr: 60430038
```

Recall that you have developed a multi-cycle processor, so each instruction will require several steps. The table attached at the end of the lab lists the inputs and expected outputs on each cycle. The steps for the first instruction have been completed for you. Please fill out the remainder of the table.

A mips.cmd file has been provided for you with the commands to simulate the first instruction. Add more commands to check the other instructions and verify that the final output driven on the writedata lines is 5B. If they do not, track down the problem; it is likely an error in your `addi` code or your FSM synthesis. You can use the Tools ? Simulation ? Down Hierarchy command to descend into various cells and check that the inputs and outputs match expectation. This is easiest if you give the cells names in your schematics.

*** The simulator seems to hang Electric fairly regularly, so save often. If you can reproduce the way to cause the hang, please report it.

*** Note that there is a bug in this year's lab. On each instruction fetch step, the address is incremented by 4 (one word) rather than 1 (one byte). You may ignore checking the address during instruction fetch to avoid fixing this bug.

2. Top-Level Layout

Create a new layout in the lab5_xx library named mips. In this top-level cell, place your datapath, alucontrol, and controller so that they will be easy to connect. Wire together the modules. Don't forget to connect power and ground with fat wires and arrays of vias to handle the higher levels of current that may flow! You will avoid creating a rats nest of wiring if you systematically reserve metal2 for vertical lines and metal1 for horizontal lines. You may wish to consider placing a large number of long horizontal wires between the datapath and the controller/alucontrol, then dropping vertical lines in a systematic fashion to connect the blocks together.

Export all the inputs and outputs. Label the internal signals. Be sure all your labels and exports agree with the schematic.

Verify that your layout passes DRC (except the controller), ERC, and network compares with the schematic. Fix any problems that might arise. As usual, simulating may catch problems that are difficult to isolate with NCC.

3. Pad Frame Assembly

The tiny transistors on a chip must eventually be attached to the external world with a pad frame. A pad frame consists of metal pads about 100 microns square; these pads are large enough to be attached to the package during manufacturing with thin gold bonding wires. Each pad also contains large transistors to drive the relatively enormous capacitances of the external environment.

Electric provides a handy pad frame generator that automatically assembles a pad frame for you from a library. To use the pad frame generator, you need your library, a pad library, and a pad arrangement file. The pad library is named muddpads11_ami05.elib. Two pad arrangement files, mips_sch.arr and mips_lay.arr are also in the class directory. Look at one of the pad arrangement files. It defines the library in which the pad frame is stored, the name of the cell to generate, and the name of your top-level design. It then contains a list of each pad for the system.

Use the Tools • Generation • Pad Frame command and choose the mips_sch.arr file. This will create a new facet called padframe{sch} with generic unrouted arcs connecting the pads to the mips core. Inspect the padframe to be sure it looks reasonable.

Opening the muddpads11_ami05 library will leave you with two libraries open. Note that the Edit Facet dialog has a drop-down list of open libraries to help you navigate among the libraries. You may also find the File • Change Current Library command to be useful to switch between default libraries.

Now you must connect power and ground for the pads together. Each pad has a VDD and a GND export on both edges that must be connected to the adjacent pad. To do this, we can take advantage of the auto stitching capability to save having to make every connection by hand. Select everything with the Select All command. Then choose Tools • Routing • Autostitch Highlighted Now to automatically connect all the overlapping VDD and GND exports in the pads.

Repeat the process for the layout using mips_lay.arr. Autostitch power and ground again. In a real design you would need to delete the generic unrouted arcs in the layout and replace them with real metal lines, but in this lab you may skip that step.

Do a thorough final check on the top-level layout using the following steps and fix any errors (except DRC errors in the controller):

1. Tools • DRC • DRC Options: Clear valid DRC dates
   Check that alternate contact rules are selected and 3 layer submicron rules are used.
2. Tools • DRC • Hierarchical Check
3. Tools • Network • Network Options:
   Clear valid NCC Dates
   Recursive Through Hierarchy
   Select Ignore Power and Ground and Check Export Names
   Do NCC Now
4. Tools • Electrical Rules • Analyze Wells
5. Simulate your design from the master command file. This command file would also be used to test your chip after it is manufactured.

4. Tapeout

The final step in designing a chip is creating a file containing the geometry needed by the vendor to manufacture masks. Once upon a time these files were written to magnetic tape, so the process is still known as tapeout. The two popular output formats are CIF and GDS; we will use CIF (the Caltech Interchange Format) because it is a human-readable text file and thus easier to inspect for problems than the binary GDS format.

To write a CIF file:

1. File • IO Options • CIF Options:
   Check Output Instantiates Top Level
   Select Show Resolution Errors
   Output resolution 0.5
2. File • Export • CIF

Look at the CIF file in a text editor. You should be able to identify the various cells. Each cell contains boxes (rectangles) for each layer. For example, the CMF layer is first-level metal and the CWN is n-well.

Electric may issue warnings about obscured facet exports; these may be ignored. However, if you receive any resolution errors, try to fix them. These occur if the elements of the design are not all on a 0.5 lambda grid and usually indicates sloppy layout practices such as not using facet centers or not drawing on grid.

Electric will report a MOSIS Cyclical Redundancy Check (CRC) code to ensure your CIF transmits correctly. Record the two numbers indicating checksum and count, respectively.

5. Summary

If you have successfully completed the lab, congratulations! You have designed, assembled, and tested your own microprocessor! You now are familiar with the major aspects of custom CMOS VLSI design:

- Leaf cell design
- Datapath design and assembly
- Hardware specification with Verilog
- Standard cell synthesis and place & route
- Top-level system assembly
- Pad frame generation and routing
- Switch-level simulation and logic debug
- Design Rule Checking
- Electrical Rule Checking
- Network compare and debug of mismatched networks

You will put these skills to use as you proceed with your final project!

6. What to Turn In

Please provide a hard copy of each of the following items:

1. Please indicate how many hours you spent on this lab. This will not affect your grade, but will be helpful for calibrating the workload for the future.

2. What was unclear in this lab writeup? How would you change it to run more smoothly next time?
3. A printout of your mips schematic.
4. Your table with inputs and expected outputs when simulating the processor.
5. Simulation waveforms demonstrating correct operation of the mips processor.
6. A printout of your mips layout.
7. What is the verification status of your mips layout? Does it pass DRC? ERC? NCC?
8. A color printout of your chip layout.
9. What is the verification status of your mips layout? Does it simulate? Pass DRC? ERC? NCC?

Extra Credit

As you are probably aware by now, Electric has plenty of bugs and idiosyncrasies. A major goal of this class is to improve the stability and ease-of-use of Electric. Please email your bug reports directly to Prof. Harris in the format described in Lab Manual 1.

Cycle	Reset	MemData	Adr	WriteData	MemRead	MemWrite	ALUSrcA	ALUSrcB	ALUControl	State
0	1	x	x	x	x	x	x	x	x	x
1	0	20	00	x	1	0	0	01	010	0
2	0	01	01	x	1	0	0	01	010	1
3	0	00	02	x	1	0	0	01	010	2
4	0	43	03	x	1	0	0	01	010	3
5	0	x	x	x	0	0	0	11	010	4
6	0	x	x	x	0	0	1	10	010	13
7	0	x	x	x	0	0	x	x	x	14
8	0	20	04	x	1	0	0	01	010	0
9	0									
10	0									
11	0									
12	0									
13	0									
14	0									
15	0									
16	0									
17	0									
18	0									
19	0									
20	0									
21	0									
22	0									
23	0									
24	0									
25	0									
26	0									
27	0									
28	0									