CMOS VLSI Design
Lab 4: Full Chip Assembly

In this final lab, you will assemble and simulate your entire MIPS microprocessor! You will first put together the top-level layout, check that it is clean, and simulate against the RTL. Then, you will generate a pad frame to connect your processor to the external world and verify that your system works in the pad frame.

I. Top-level Schematic

Copy your lab3_xx library to lab4_xx for this lab. Look at the top-level mips cell that includes your datapath and controller icons. The system has the exported inputs and outputs given in Table 1. Notice that the internal wires and icons are also named to simplify debugging. Buses are exported as individual signals for Electric's automated pad frame generator, which cannot understand buses. Study the schematic until you are sure it is correct.

Inputs	Outputs
ph1	adr[7:0]
ph2	writedata[7:0]
Reset	Memread
memdata[7:0]	Memwrite

Table 1: MIPS Processor Inputs & Outputs

II. Top-Level Layout

Create a new layout in the lab4_xx library named mips. In this top-level cell, you will just have to place datapath and controller, align the exports, and use auto-stitch to connect the two. ph1, ph2, and reset exports on the left side of the cells need to be connected by hand. Connect the metal3-metal2 vias using vertical metal2 to the datapath's inputs. Don't forget to connect the fat wires for power and ground. They use arrays of vias to handle the higher levels of current that may flow! Make sure to export both pairs and add power/ground connected at higher-level annotations so that it passes NCC.

Export all signals that need to be exported. Export • Re-Export Everything can speed this up! Label the internal signals. Be sure all your labels and exports agree with the schematic. The Export • Show Exports command is helpful for checking that the correct signals are exported.

Verify that your layout passes DRC, NCC, ERC. Electric caches DRC results to improve the speed of DRC, but this can sometimes lead to missed errors. Go to File • Preferences • Tools • DRC and click "Clear valid DRC dates." Recheck DRC. You should get into the habit of doing this during the final verifications of a project. Fix any problems that might arise. Simulate your mips layout with the RTL. As usual, simulating may catch problems that are difficult to isolate with NCC.

III. 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 muddpads13_ami05.jelib. The pad arrangement file is mips8.arr. Both are in the lab 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 (called the core), and the ports that should be used to line the pads up together. It then contains a list of each pad for the system.

Open the muddpads13_ami05 library. Look at the schematic for pad_io (an input/output pad). Figure out how it works. Pad_in and pad_out are similar, but the enable is hardwired to 0 or 1.

Then look at the pad_io layout. You will see a big square pad consisting of a stack of metal1, metal2, and metal3, along with a region marked with circular stipples called the passivation layer. The chip is normally covered with a layer of silicon dioxide called the overglass or passivation to protect the circuitry. This overglass is etched away over the pad so that wires can be bonded to the metal stack. The "passivation" layer in Electric indicates where the etching should occur.

Use the Tool • Generation • Pad Frame Generator command and choose the mips8.arr file. This will create three two new cells called chip{lay}, chip{ic} and chip{sch}. chip{lay} will have generic unrouted arcs connecting the pads to the mips core. Inspect the padframe to be sure it looks reasonable. You'll need to move the core inside of the pad frame. Connect both pairs of the large power and ground rails to the power and ground pads at the right and export them. Check the layout with DRC, NCC, and ERC.

You could route the wires from the padframe to the core in the layout, but this is tedious. Electric includes an auto router called Sea-of-Gates router. It will automatically route generic unrouted arcs if you select them or will route all of them if you have none selected. Click the background of the layout cell to deselect any objects and use Tool •

Routing • Sea-of-Gates Route. It may take a few minutes to complete, and you should check DRC after it finishes and fix any errors. Your completed chip should resemble Figure 1.61 from CMOS VLSI Design, except with wires routed to the pads and the large ground and power rails.

Save your library and do a thorough final check on the top-level chip layout using the following steps:

1. File • Check Libraries • Check
   Makes sure library is stable before verification proceeds.
2. File • Preferences • Tools • DRC • Clear valid DRC dates
   Do a fresh DRC on the entire chip, even if the cells passed once before (this shouldn't be necessary, but experienced chip designers are paranoid)
3. Tool • DRC • Check Hierarchically
   There should be no DRC errors
4. File • Preferences • Tools • NCC
   Make sure that Hierarchical Comparison and Check transistor sizes are selected. Verify "Don't recheck cells that have passed in this Electric run" is unchecked.
5. Tool • NCC • Schematic and Layout Views of Cell in Current Window
6. Tool • ERC • Check Wells (this may take a while)
   If the power and ground pads were hooked up, this should give no errors.
7. Simulate the layout with Verilog using the testbench used to validate the RTL.

IV. 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, and the process is still known as tapeout. Before taping out, run the checks mentioned above.

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, choose File • Export • CIF and save the file as mips8.cif. You will get a large number of resolution errors because Electric tends to create some arcs off of the ½ λ grid that CIF prefers. You can ignore these.

Look at the CIF file in a text editor. You should be able to identify the various cells. Each cell contains boxes (B) (rectangles) for each layer (L). For example, the CMF layer is first-level metal and the CWN is n-well. The C statement instantiates a cell whose number has already been defined in the file.

This completes the lab. You now know how to create layouts and schematics. You know how to draw leaf cells, then build up datapaths and random control logic blocks. You know how to verify the design using DRC, ERC, NCC, and simulation. And you know

how to put the design into a padframe and run final checks before manufacturing. May you build many interesting chips!

V. 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. A printout of your mips{lay}.
3. Does mips{lay} pass DRC? NCC? ERC?
4. A printout of your chip{lay} and chip{sch}.
5. Does chip{lay} pass DRC? NCC? ERC?
6. Does your chip{lay} simulate correctly against the RTL?