



## 1. Overview

The final project is a chance for you to apply your new skills in VLSI design to a moderate sized problem of your choosing as part of a two-person team. You should begin thinking about a project and teammate right away. Your project has the following milestones with dates defined in the syllabus and grade percentages listed:

- Project Proposal 10%
- Verilog Checkoff 15%
- Schematic Checkoff 10%
- Block Layout 5%
- Final Project Checkoff 30%
- Report Due 20%
- Project Presentations 10%

## 2. Project Requirements

Your project must fit on a 1.5 x 1.5 mm 40-pin MOSIS “TinyChip” fabricated in a 0.6  $\mu\text{m}$  process. That means your project cannot exceed 5000 x 5000  $\lambda$  including I/O pads.

Therefore, the core of your project must fit in a 3400 x 3400  $\lambda$  box and have exactly 40 pins. Three pins should be dedicated to VDD and three to GND, so only 34 are available as I/Os. Exceptions may be made for project proposals that need to exceed this area or pin count but are simple enough to be feasible in the time allotted; such projects will not be placed in a pad frame. Unless negotiated in the proposal, there will be a grade penalty for exceeding the area available.

Your project should contain at least one custom datapath or array and at least one synthesized block. It should also involve the layout of at least one new leaf cell.

Be creative when selecting your project. Your project should be bigger than a weekly lab assignment, but small enough to be doable. If in doubt, err on the side of smaller; you will receive a much better grade for a simple project that is completed and convincingly verified than a large project that is incomplete. Examples of suitable projects are listed below, but do not let the list limit your imagination!

- Digital signal processing unit
- Processor with new instructions or on-chip memory
- High-speed adder
- Games (tic-tac-toe, checkers, Simon, etc.)
- SRT divider
- Tiny FPGA
- Encryption unit
- Alarm clock
- Clinic-related circuits
- Cache memory
- Translation lookaside buffer
- Analog / Digital Converter
- CORDIC function generator
- Phase-locked loop
- Metastability characterization circuit
- Process characterization circuits

### **3. Design Budgeting**

One of the challenges of chip design is to learn to budget your time and area. Experience is crucial to doing this well. One of the elements of the project will be to track this data so that you can learn to budget in the future.

In your proposal, you will submit a floorplan with area estimates. At the conclusion of the project, you will submit a comparison between the initial estimates and the actual results, along with an explanation of discrepancies.

Even more importantly, track the time you spend on the project. Keep a notebook and update it each day you work on the project. Note how much time you spent on each cell. Include the time spent designing the schematic, symbol, and layout as well as time spent for simulation, DRC, ERC, and NCC.

### **4. IC Fabrication**

We expect to receive funding from the MOSIS Educational Program to fabricate a few TinyChip projects. If your chip is fabricated, you will receive 5 packaged parts in the winter. Priority for fabrication will be given to teams on the following basis:

1. Layout fits on a 40-pin MOSIS Tiny Chip and is wired to the pad frame.
2. Layout passes all DRC and LVS tests and simulates successfully
3. At least one teammate is on campus in the fall and is committed to testing the chip.
4. Among teams meeting the above qualifications, the teams receiving the highest grades will have priority to fabricate.

### **5. Deliverables**

Your team is responsible for the following deliverables on the dates described above. Rubrics are given for each. If you feel there has been serious inequity between the work you and your teammate deliver, contact the instructor.

### **Project Proposal**

A 2-page proposal describing what you plan to build. It must be specific enough that the instructor can determine when you demonstrate your project that it meets the specs of the proposal. The proposal should also include a table listing all the inputs, outputs, and bidirectional pins on the chip. It should include a floorplan, drawn to scale with dimensions labeled, indicating the top-level blocks and whether each of these blocks is synthesized or custom. To produce a credible floorplan, you will need to do some preliminary design and probably create a slice plan for the datapath.

___ / 4	Unambiguous requirements
___ / 1	I/Os fit on chip
___ / 3	Floorplan backed by preliminary design showing design fits on chip
___ / 1	At least one synthesized block and one datapath or array
___ / 1	At least one new cell
___ / 10	

### **Verilog Checkoff**

Schedule a check-off with the instructor to demonstrate that the Verilog is complete and simulates successfully with a self-checking testbench. Be sure that your simulations demonstrate complete and convincing operation of the functions specified in the proposal. Your Verilog should be readable and suitably commented.

___ / 4	Cleanly written Verilog
___ / 1	Synchronous design with proper 2-phase clocking
___ / 1	Synthesized and custom blocks are separate
___ / 1	Each block is entirely behavioral or entirely structural
___ / 3	Convincing test vectors
___ / 1	Self-checking test bench
___ / 4	Verilog passes test vectors
___ / 15	

### **Schematic Checkoff**

Schedule a check-off with the instructor to demonstrate that the schematics are complete and simulate successfully using the same self-checking testbench. The schematics should include at least one custom leaf cell, at least one custom block, and at least one synthesized block. They should also include a pad frame, modified from the MIPS frame to accommodate the necessary inputs and outputs. It is recommended to call your top-level modules “core” (without the pad

frame) and “chip” (with the pads). Both should have the same I/Os as the Verilog. If you have analog blocks, show SPICE simulations demonstrating correct operation.

- \_\_\_ / 2      Cleanly drawn schematics
- \_\_\_ / 1      Pad frame
- \_\_\_ / 1      Custom leaf cell
- \_\_\_ / 1      Custom block
- \_\_\_ / 1      Synthesized block
- \_\_\_ / 4      Chip schematic simulates in Verilog self-checking testbench
- \_\_\_ / 10

### Block Layout

At this point, your leaf cell should be complete, the synthesized block should be generated using SOC Encounter, and the custom block should be at least 50% complete. Pay attention to making clean and efficient layouts with no DRC or LVS errors.

- \_\_\_ / 2      Leaf cell layout cleanly drawn and passes DRC and LVS
- \_\_\_ / 1      Synthesized block generated with encounter
- \_\_\_ / 2      Custom block drawn cleanly and at least 50% complete
- \_\_\_ / 5

### Final Project Checkoff

Demonstrate a complete layout fully routed to a pad frame. Use fat power wires where appropriate. Show that the chip passes DRC and LVS (with “Join nets with same name” turned off). Note that LVS does not understand the sizes of the I/O transistors in the padframe, so check sizes in the core but not at the chip level. Show that the schematics (including the pad frame) still simulate correctly in the self-checking testbench. Show that the GDS output can be read back in and pass DRC (excluding the 144 pad frame DRC warnings) and LVS. Most of the points are for your core layout, so if you are running out of time, focus there.

- \_\_\_ / 2      Chip schematic simulates in original self-checking testbench
- \_\_\_ / 1      Chip passes DRC
- \_\_\_ / 3      Chip passes LVS
- \_\_\_ / 2      Chip layout is clean, efficient, manufacturable
- \_\_\_ / 1      Chip includes padframe
- \_\_\_ / 1      Chip writes out GDS and reads back in and passes DRC and LVS
- \_\_\_ / 5      Relative difficulty of the project (3 = typical)
- \_\_\_ / 15

### Project Report

Your final report should provide all the information another engineer would need to know to understand and test your chip after fabrication. The following content is recommended:

- Cover page
  - Project title, designers, and a chip plot
- Introduction
  - A brief high-level description of the chip function and the manufacturing process
  - Summary of results: does it simulate and pass DRC, LVS?
- Specifications
  - Table of inputs, outputs indicating name, direction, bus widths
  - Theory of operation of the chip relating the outputs to the inputs
- Floorplan
  - Compare the actual floorplan to the proposal and explain discrepancies
  - Slice plan for datapath(s)
  - Pinout diagram indicating names and pin numbers for each pin
- Verification
  - Does Verilog pass testbench?
  - Do schematics pass testbench?
  - Does layout pass DRC and LVS?
  - Does GDS load correctly and pass DRC and LVS?
  - If the chip contains any analog blocks, show the HSPICE simulations
  - Explain any discrepancies or concerns
- Postfabrication test plan
  - How will the chip be tested if it is fabricated?
- Design Time
  - Summary of design time spent on each component of the project
- File Locations (paths on Tera to find)
  - Verilog code
  - Test vectors
  - Synthesis results
  - All Cadence libraries
  - HSPICE simulations of any analog blocks
  - GDS
  - PDF chip plot
  - PDF of your report
- References (if necessary)
- Appendices:
  - Verilog code and testbench (in monospaced font such as Courier 8 pt, with columns lining up and lines wrapping cleanly)
    - Include test vectors unless they are unreasonably long
  - HSPICE testbench(es), if applicable
  - Legible schematics of the cells you produced (labeled with cell name)
    - Do not include large synthesized blocks that are unintelligible
    - Do not include muddlib cells
  - Color layouts (legibly printed, labeled with cell name)
  - Other materials generated during the project

\_\_\_ / 1    Cover Page  
 \_\_\_ / 2    Introduction

__ / 2	Specifications
__ / 2	Floorplan
__ / 1	Verification
__ / 2	Test Plan
__ / 2	Design Time
__ / 1	File Locations
__ / 1	Verilog, testbench, vectors
__ / 1	Schematics
__ / 1	Layouts
__ / 1	Other (if applicable)
__ / 3	Writing: concise, unambiguous, grammatical, graceful
__ / 20	

Avoid schematics or layout with black backgrounds that use a huge amount of toner and are hard to read. One way to produce reasonably attractive schematics and layouts is to print them to PDF or EPS (CUPS-PDF writes the files to your home directory). Use the convert command to change the PDF to JPEG if you need to paste it into Word or Powerpoint:

```
convert ~/job_884-untitled_document.pdf chip.jpg
```

Use scp to move the results to your computer. LaTeX users can handle EPS directly and avoid this conversion step. Adobe Illustrator also performs conversions and cropping well.

## Project Presentation

You will give a 10-minute conference-style presentation of your chip during presentation days. Your presentation should explain your design and results to your classmates. It should include a functional overview, the chip pinout and floorplan, simulation and verification results, design time and area budgets, a description of the design challenges, and a top-level chip layout. The presentation should be in PowerPoint or PDF format for projection in class. Design the presentation so that a freshman would find it interesting.

__ / 2	Functional Overview
__ / 1	Pinout & Floorplan, Simulation & Verification, top-level layout
__ / 1	Design time and area budgets
__ / 1	Design Challenges
__ / 3	Speaking and visuals
__ / 2	Interesting and accessible to audience
__ / 10	

## 6. Hints

In the past, we have encountered a few interesting problems:

1) Many teams didn't take the time to understand two-phase latches before writing their Verilog code. Be sure you use registers built from two-phase latches as is done in the MIPS Verilog. Never use `always @(posedge clk)` because we have no tools to check for clock skew and ensure your hold times are satisfied, and because this construct synthesizes onto an edge-triggered flip-flop, which is not in muddlib.

2) If you write asynchronous logic, I won't help debug it and the synthesis tools may not synthesize it correctly. Everything in your circuit should either be combinational logic or a register built from 2-phase latches. All cyclic paths should contain a register.

3) Some schematics of synthesized controllers with enabled resettable registers (flopnr) failed to simulate correctly, giving x's instead of properly resetting. The problem occurs because synthesis generates a structure consisting of a multiplexer followed by the two latches. The output of the register feeds back to control the select signal of the multiplexer. During reset, both mux data inputs are 0, theoretically giving proper operation independent of the select signal. However, in simulation, the x on the select signal results in an x on the mux output in simulation. We have not found a solution to synthesize better logic. However, a workaround has been to initialize the latches at startup to a non-x value. This is a bit dangerous because it could conceal a legitimate uninitialized latch problem caused by improper resets in the Verilog. However, if the Verilog simulates correctly, the workaround is plausible. Change the schematic netlist for the latch\_c\_1x to add the following lines:

```
//THESE LINES ARE CHANGED FROM ORIGINAL LATCH
reg rst;
initial begin
    rst = 1; #5; rst = 0;

end

nmos jamb(masterb, cds_globals.gnd_, rst);
//END CHANGED SECTION
```

We are still looking for a better solution to synthesize logic that lacks this pitfall. One idea is to avoid flopnr modules in the Verilog code for synthesized blocks.

4) Some layouts generated with SOC Encounter from synthesized controllers failed to pass LVS. This occurred when the synthesis generated a schematic with a constant value of 0 produced by connecting VDD to the input of an inverter. Encounter doesn't route power to signal pins, leaving the inverters with floating inputs. A workaround is to complete the route by hand, connecting vdd! to the inverter input with metal2 in the layout. Then the schematic must also be corrected by renaming VDD to vdd!.

These problems occur when you have a signal that is a constant. Look for synthesis warnings after `check_design` about signals connected directly to logic 0 or logic 1 and optimize those signals out of your Verilog code by hand, then resynthesize and be sure you've eliminated these warnings.

5) Take synthesis warnings seriously. Unless you understand them well and can be certain they are unimportant, they are usually a hint of a serious problem lurking in your code.

6) As you may become painfully aware, CAD tools are imperfect. Keep regular backups of your library lest it become corrupted (very rare, but potentially catastrophic).

## 7. Chip Plot

Screen shots are sufficient for the report, but you'll need higher resolution if you want an attractive poster of the chip. If you take the time to make a high resolution file, share it with Willie Drake and he can print one for each team member upon request.

Note: this process will make a VERY big PDF that may not display well in most programs. Photoshop, Illustrator, and GIMP were the only programs on my computer that could render it correctly.

1. Export a GDS file (follow instructions in lab 4)

2. Open Electric

```
java -jar /courses/cmosvlsi/20/tools/electricBinary-9.07.jar
```

3. In Electric, File --> Import --> GDS II (Stream)...

4. Select the GDS file from step 1

5. Select "mocmos" as the layout technology

6. In the Explorer tab, double click chip{lay}

7. File --> Export --> CIF (Caltech Intermediate Format)... as chip.cif

8. Ignore the thousands of errors it will report

9. Run cif2pdf

```
/courses/cmosvlsi/20/tools/cif2psc -l /courses/cmosvlsi/20/tools/layermap.txt chip.cif
```

10. scp your chip.cif.pdf file to an ECF machine and view it in Illustrator to check that it is good.

11. If you want an even spiffier plot with nice transparency and outlines of each layer, do the following more time-consuming steps.

```
/courses/cmosvlsi/20/tools/cif2psc -l /courses/cmosvlsi/20/tools/layermap.txt -nt chip.cif
```

Copy to an ECF machine and open in Illustrator

- Ungroup everything
- Select a metal3 wire.
- Use Select \* Same Fill Color to select all.
- Group.
- Set Transparency to 70%.
- Use Windows \* Pathfinder \* Merge to merge all rectangles on the layer (seems very slow for M2 on a TinyChip)
- Set the outline color to a darker shade and an appropriate stroke width
- Repeat for Metal2, Metal1, Poly