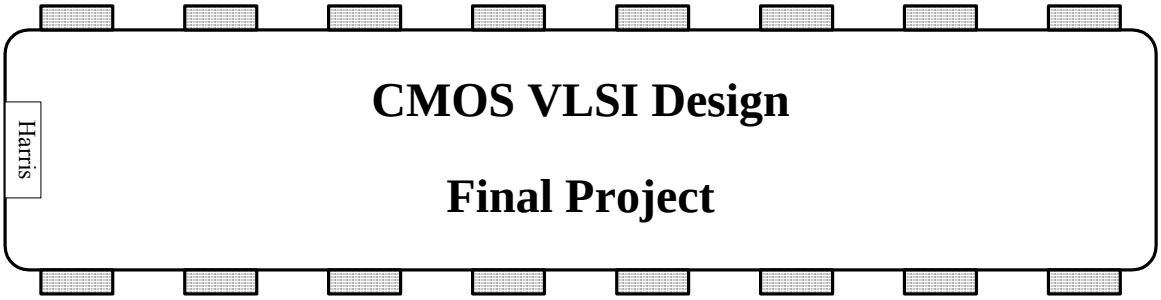

1. Overview

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:

Project Proposal Due
Verilog Checkoff
Schematic Checkoff
Block Layout
Final Project Checkoff
Report Due
Project Presentations

2. Project Requirements

Your project must fit on a 1.5 x 1.5 mm 40-pin MOSIS "TinyChip" fabricated in a 0.6 μm process. That means your project cannot exceed 5000 x 5000 λ including I/O pads. Therefore, the core of your project must fit in a 3400 x 3400 λ 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!

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

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 fewTinyChip projects. If your chip is fabricated, you will receive 5 packaged parts in the fall. 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:

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.

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.

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.

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.

Final Project Checkoff

Demonstrate a complete layout fully routed to a pad frame. Use fat power wires were 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.

80% of the points will be given for having a "core" layout that passes the checks. Another 10% will be given for "chip" including the pad frame and chip assembly, and 10% for the GDS. If you are running out of time, focus on your core layout.

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
- 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
  - Where 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 (labeled with cell name)
  - Other materials generated during the project

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 your Charlie folder or a FTP client 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.

6. Grading

Your project will be graded as follows:

Proposal	10%
Verilog Checkoff	15%
Schematic Checkoff	10%
Block Layout	5%
Final Checkoff	30%
Final Report	20%
Presentation	10%

If you feel there has been inequity between the work you and your teammate deliver, contact the instructor.

As we all know, CAD tools are imperfect. Keep regular backups of your library lest it become corrupted (very rare, but potentially catastrophic).

7. Hints

In Spring 2010, we 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 (flopenr) 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 flopenr 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.