ECE 464/564 ASIC DESIGN PROJECTS

Dr. Paul D. Franzon

Introduction

The purpose of the project is for you to demonstrate your skills in designing a substantial digital system using the Verilog/Synthesis techniques taught in class. This document describes the projects and turn-in requirements. These projects are supported by the tutorials prescribed with the class homeworks.

Projects can be done individually.

Submission Instructions:

- **Project Plan.** Paper report due in class as per the class schedule. It must follow the format attached. There is a 10% penalty for not following the format.
- **Project Verilog and synthesis files.** Submitted electronically on the date indicated in the class schedule. Please turn in the following:
  - All Verilog files **AS ONE FILE**
  - Synopsys view_command.log file from complete synthesis run
  - Simulation results file showing correct functionality
  - Project report
- **Project Report.** Complete report to be turned in electronically with project files. It must follow the format attached. There is a 10% penalty for not following the format.
- **Demo.** There will be live demos following project turn in. The main purpose of these is so you can demonstrate to the TA what works and support your claimed results.

Collaboration

You are free to share ideas, such as high level approaches with others. However, you are **forbidden from sharing Verilog code, or code fragments, with any other student**. You are also **forbidden to work on the code, or any code portion, as a group**, and then turn in individual solutions based on that joint work. We will be running the results through code comparison tools that are very good at detecting very broad code similarities. Violators will be punished, most likely earning zero, amongst other punishments. Given how much the project is worth, this will severely hurt your final grade. You can share with your partner, but not with other groups or individuals.

General Instructions

Project Plan

Your written Project Plan is due in class on as per the class schedule. **A specified format for report is attached.** Items to be included in this report include the following:

- Block diagrams of your design, clearly identifying any design hierarchy, all registers, all module I/O. Neat block diagrams are expected, not rough hand-drawn sketches. Include a written description of the function of each module. E.g. For a multiplier, which multiplier algorithm are you planning to use.
- A project execution plan, including the following:
  - Who is responsible for each module
  - A verification plan. How do you intend to verify that the design works correctly? Who will be responsible for system verification?
- A risk assessment plan. Where are the greatest risks in this project? E.g. What areas of the design are you unsure of? Where are your greatest concerns in executing this project? Note, this section is intended for your benefit (so you can focus your efforts on the parts you are least sure of). Do not expect written feedback on your risk assessment plan. However, feel free to ask questions of the teaching team.
- A milestone chart, including anticipated dates for completing module design, design integration and design verification.

Grading for this report will be as follows (out of 10):

9-10: All major design elements correctly identified; behavioral code complete with test cases. Neat and clear documentation.

7-8: At least one major design element missing, or C code incomplete, or confusing and poor documentation.

5-6: Scrappy, but honest, capturing some elements but conveying no real understanding of the design.

0-4: Extremely poor attempt.

A further 2 points is subtracted if the format below is NOT used.

Do not wait for feedback from us before proceeding with the project. If you need feedback, bring a copy of your report to myself or the TA for discussion. The main purpose of this milestone is to get you going early.

Main Project Report

Your main project report is due as per the class schedule. There are penalties for late turn-in. This date is set by the desire to complete grading these before you need to prepare for the final exam.

Your project report is to include the following:

- Written description of your approach, including block diagram of your design, description of your verification strategy and a discussion of any part of the design, synthesis or verification that you consider "tricky", novel or noteworthy.
  - Specifically note and document your *area* and your *throughput*

- Full listings of the following:
  - Verilog files, including test fixture
  - Synthesis scripts
  - Extracts from View_command.log generated by design_analyzer, including the results from the read command and timing verification. (If you use dc_shell, you will have to rerun the script in design_analyzer in order to obtain this).
  - Plots from the final design.
  - Simulation run results (waveforms or equivalent)
  - High level model of the design, if appropriate

The grading scheme for the final report are attached below. The last ten points are awarded PRIMARILY on your demonstrated ability to maximize the design goal given. Some points MAY be awarded to groups that make a contribution to the class project as a whole, beyond their individual design (this is rare). There are no other ways to earn these last ten points.

Demonstration

After turning in your project, you are to schedule a demonstration with the TA. During this demonstration, you will be "given back" the code you turned in, run it on two examples, including the hidden "666" example, AND run it through synthesis, at least the read command. If the code outputs differ from what we expect or the synthesis shows major problems, you will lose substantial points as per the grading rubric. Please note the following:

- If you want to change your code for this demonstration, you will be graded as being LATE with a 10-20% penalty depending on the date
- This is your ONE chance to explain any discrepencies observed.

Other Instructions

- You are NOT required to run the verify command in Synopsys. This takes lot of time and adds no educational value to the project (obviously in real life, you would run this command.)
- Make sure to use the 45 nm library, so we can compare area and performance accurately.

Checklist For Neophyte Verilog Designers

Module Design

1. Did you DESIGN BEFORE CODING?
2. Have you clearly identified the purpose of each register (flip-flop)?

   A common beginner error is to have unnecessary flip-flops. (This is usually a symptom of NOT designing before coding). A flip-flop is only needed when data is stored between clock cycles. Excessive flip-flops makes a design LARGE, POWER HUNGRY and SLOWS down synthesis.

3. Do you use non-blocking assignment when assigning to flip-flops?
4. Do you make sure that every variable is only assigned within ONE procedural block or continuous assignment statement?

   This will cause a LINT warning from Synopsys.

5. Do you make sure that you have NO feedback within combinational logic.

   This will cause "timing arcs" in synthesis. An example is…

   ```
   always@(B or C)
     A=B+C;
   always@(A or E)
     C=A+E
   ```

   Instead A or C must be registered.

6. Do you avoid unintentional latches by having no implied memory in combinational logic?
7. Do you avoid having 2D arrays (register arrays) in your modules?

   Put large 2D arrays into your SRAM. Put smaller ones into their own register file module.

8. Did you "right-size" the module?

   i.e. Make it as small as reasonable while containing critical paths, sharable logic, registered paths (see 9) and being "reasonable" to understand.

9. Does every path connecting input and output go through a flip-flop?
10. Are critical paths contained within one module, not split across two of them?
11. Did you have any while or FOR loops?

    The ONLY valid construct is to use a for loop to iterate through an array of bits. Most attempts by neophytes to use these constructs are invalid.

12. Did you DESIGN BEFORE CODING?

Design Hierarchy

1. Did you avoid “glue” logic in modules containing other modules?
2. Did you bring all signals connecting to things outside your design up through the ports of your “top” module?
3. Did you instance your SRAMs only in your test fixture?

Simulation

1. If saving the variables takes too much disk space, save a .trn file instead …$shm_open("count.shm"); $shm_probe(test, "AS"); (later $shm_close this). You can also choose to only save a subset of variables – see the manuals.

Synthesis

1. Did you set “current_design=top” before running “compile”?
2. Did you set “current_design=top” before running “report_area”?
3. If the compile is slow try setting “map_effort = low” and try a much slower clock.
4. Make sure you have a correct synopsys setup file and correct references to design ware, if used. (See http://www.synopsys.com/products/designware/buildingblock.html and for the Verilog files, refer to /afs/bp/dist/synopsys_syn/dw/sim_ver)
5. Did you look at the warnings and errors after the “read” statement?
6. Did you look at the timing reports?
7. If reset global signal has a high “fan-out” that is OK. This signal can be asynchronous.
8. If you get cell errors before “compile” they are usually OK. If they remain after compile (do a “report_cell”) then you have problems (usually caused by design ware).
9. Ignore cell conflict warnings after the translate command.

Synopsys gives a lot of warnings. Some of them are critical to fix. Many are informational only. Major synthesis issues include: Unintentional Latch, Timing Arcs, improper use of tri, Wired-Or, Incomplete Sensitive Lists, No connection to Input Port (LINT-34, typically). You will lose lots of points if you have any of these warnings or if you do not meet setup and hold constraints.

Minor synthesis issues include the following: Unreachable default case; Undeclared logic; and No reset.

You might see the warning:

Warning: Cannot find the design 'hist_calc' in the library 'WORK'. (LBR-1)

Warning: Unable to resolve reference 'hist_calc' in 'hist_eql'. (LINK-5)

We are getting similar warning for all the modules under top.

You can NOT ignore that. It means a big chunk of your design is not being synthesized. However, it is easily fixed by executing current_design top before you run compile. The default current_design is the most recent module read in.

Ignorable Warnings:

-Output Port not connected

-Warning: Design rule attributes from the driving cell will be set on
the port. (UID-401)
-Warning: Cell xxxx conflicts with xxxx in the target library. (OPT-106)

-Warning: Design 'hist_calc' contains 1 high-fanout nets. A fanout number of 1000 will be used for delay calculations involving these nets. (TIM-134)

-Warning: /afs/bp/dist/synopsys_syn/dw/sim_ver/DW02_cos_function.inc:360: Function 'DWF_cos' with non-empty body is mapped to 'COS_TC_OP'; body will be ignored. (VER-136)

-Warning: /afs/bp/dist/synopsys_syn/dw/sim_ver/DW02_cos_function.inc:374: Function 'cos' with non-empty body is mapped to 'COS_TC_OP'; body will be ignored. (VER-136)
-Warning: Design 'counter' inherited license information from design 'DW02_cos_8_8'. (DDB-74)
-Warning: Design 'counter' is being converted to a limited design. (DDB-75)
-Warning: Current design named counter is hidden. (UID-408)
-Warning: Verilog 'assign' or 'tran' statements are written out. (VO-4)

-Warning: signed to unsigned conversion occurs (VER_318)

LINT-2, LINT-3, LINT-28, LINT-29, LINT-31, LINT-58

The following warning is ignorable:

Warning: There are conflicts between cells in libraries NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm:NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm.db (/afs/eos.ncsu.edu/lockers/research/ece/wdavis/tech/nangate/NangateOpenCellLibrary_PDKv1_2_v2008_10/liberty/520) and NangateOpenCellLibrary_PDKv1_2_v2008_10_fast_nldm:NangateOpenCellLibrary_PDKv1_2_v2008_10_fast_nldm.db (/afs/eos.ncsu.edu/lockers/research/ece/wdavis/tech/nangate/NangateOpenCellLibrary_PDKv1_2_v2008_10/liberty/520). (**OPT-187**)

There are other ignorable warnings but they depend on the context of the code. You need to READ the warning. The plain English language in the warning is generally understandable. You need to put them in the context of your code.

Changing Port Delays

Depending on the details of the technical project, you might need to do that. Here is an example…

```
set SRAM_DELAY [expr 0.02 + 2.000]

set_input_delay $SRAM_DELAY -clock $clkname [get_ports S1_ReadBus1[*]]
```

DesignWare

If you need to use multipliers or dividers, those in DesignWare are available. Please review the documentation in the resources section.

Name	Student ID	Core (90)	Performance/Area (10)	Total (100)

Bonus Data:

Core Group Points

Requirement	Grading Scheme	Grade
Does the project run the first data set correctly?	40 = yes, with correct timing 30 = no, but almost does (minor problems only)	(40)
Does the project run the second data set correctly?	10 = yes 0 = no (Scale depending on severity)	(10)
Does the project synthesize without any significant problems?	20 = yes 10 = major problems, that would indicate functional flaws in final chip, including timing 0 = no results obtained from synthesis	(25)
What is the quality of the project report?	10 = High quality report in all regards, professionally presented, conveying all required content 5-9 = Some missing content, incorrect format, or scrappy presentation 1-4 = Major deficiencies	(10)
What is the quality of the Verilog code?	5 = Understandable. Reasonable use of comments. Reasonable test fixture and verification suite. 1-4 = Difficult to understand or follow 0 = almost structureless	(5)
	Core Sub-Total (90)	

Written Comments:

Submit: Verilog RTL **as one fine**, testbench RTL, final netlist as .db., view_command.log, + .txt file with clock name and top module name, (please tar gzip these)

Preliminary report format – Your first page MUST match this format

- **Attach other pages as needed**

/10

Project Responsibilities:
StudentID. Name 1:
StudentID. Name 2:

Summary Risk Plan:

Schedule:

Brief Description of High Level Code. Include # lines and performance data.

Brief Description of Mode of operation, including selected algorithms

High level sketch. Add details on the following pages if necessary

Final Project Report First Page. Must match this format (Title) – attach other pages as needed

Project Responsibilities (module names, other assigned tasks): StudentID. Name 1: StudentID. Name 2:

Delay (ns to run provided example). Clock period: # cycles":	Area: (um^2) Logic: Memory: N/A	1/(delay.area) ($ns^{-1}.um^{-2}$)
Delay (TA provided example. TA to complete)		1/(delay.area) (TA)

Abstract
Abstract should briefly summarize that the hardware function is (remember a future employer might be reading this), what your approach was, and the main results achieved.

Project Title

Student names

Abstract

Repeat this if you want a stand-alone report without the previous page.

Note. This outline is not intended as a rigid structure but as guidance.

1. Introduction

- What hardware is being designed here (remember, you might want to show this to a future employer who has not read the description I produced).
- Summary of key innovations if any claimed
- Summary of results achieved.
- Structure of the rest of this report

2. Micro-Architecture

- Hardware "algorithmic" approach used.
- High level architecture drawing, and description of data flow
- Details on claimed innovations

3. Interface Specification

- Detailed description of interface to your design, to data sheet standards.
- Include a table listing each signal, its width and function
- Include an interface timing diagram

4. Technical Implementation

- Discussion of high level modeling and results achieved
- Discussion of any hierarchy below that
- Discussion, if needed, of detailed implementation

5. Verification

- Description of approach used to verify correctness.

6. Results Achieved

- Throughput, area, energy, etc.

7. Conclusions

- Summary of project and key results