VLSI Projects

Adnan Aziz
Electrical and Computer Engineering
The University of Texas at Austin

1 Introduction

Your task in this project is to use the skills you have been acquiring through the lectures and labs to design a fairly sophisticated module—an intellectual property (IP) core.

The purpose of the project is threefold:

1. It is worth a large fraction of your grade (but this should be the least important item);
2. working on this project should be training on how to go about approaching a design project; and
3. the project should yield results:
    - experimental results on the efficacy of proposed VLSI architectures, and
    - suggestions for improving these architectures.

Projects can be done individually, or in groups of 2–3; naturally, I will expect more from group projects.

2 Timeline

I do not want a student going off on a tangent, only to learn at the end of the semester that this happened. On the other hand, I don't want to stifle creativity by monitoring things too closely. Most of all, **I do not want rush jobs**, where everything is crammed into a few days at the end of the semester. You need time to develop these projects.

Project selection: You should make a decision as to the projects you are interested in working on and turn in hardcopy in my office of your specifications document and a timeline by **Friday, 10.27.07, 11:59am**.

Intermediate reports: I would like you to turn in hardcopy of a design document by **Tuesday, 11.6.07**, in class.

Final report: The final report should consist of the following, which you think of as the individual chapters:

- Specifications document
- (Optional) Marketing document
- Design document
- User document
- Testing strategy and results
- Optimization strategy and results
- Source code and layout

The report is due in hardcopy on **Friday, 12.7.07 by 11:59 am**. in my office.

3 Report details

3.1 Project report—details

3.1.1 Specifications

The specifications document should include a high-level overview of the IP block you are implementing; a description based on a diagram or set of diagrams is the best way to do this. It should also include the a summary of the logical interface the block presents to its environment.

In addition, the document should include the area, power, and performance numbers you are targeting. If you base your work on an existing design, you should be able to come up with estimates on these parameters; otherwise, back-of-the-envelope calculations are fine. It's not imperative that you meet the numbers in the specification document.

The specifications document should not discuss the implementation; its focus is the functionality that you will implement, and the cost of this functionality.

3.1.2 Marketing

Optionally, you can include a marketing document. This should include an estimate of how many chips/cores you expect to sell, what price people will pay for them, how much they will cost to design and build, and how you will get customers to know about your product.

The text has a good discussion of design economics in Chapter 8, especially Section 8.5. EEtimes and Dataquest are other standard places to get information from.

3.1.3 Design

The design document should include a description of how you will implement the specification—a set of figures is the best way to convey this. The implementation discussion should include the basic architecture and algorithms, as well as the floorplan, and circuit technology, etc.

You should also make notes on the optimization techniques you expect to use and their implications to your design, and the trade-offs they will entail. For example, if you have long interconnects, you may want to state that you intend to overcome problems resulting from crosstalk by shielding, and hence all long nets should have enough space between them for such shielding lines.

All choices should be justified, on the basis of references to portions of the book/research papers, and by logical arguments.

The design document should also include an overview of the tool suite you will be using, the naming conventions for variables/modules/files, the regression control strategy,[1] and an issue tracking mechanism (which could be just entries in a text file).

Think of the design document as something you would give to an engineer just joining the project to help him/her come up to speed.

(Design documents also spell out a regular system of "code reviews," where designers have to explain what they have done to their colleagues, at a very detailed level, e.g., a walk-through of RTL code. We won't have review process is probably too involved for a class project.)

The specifications and design documents do not have to be exactly what you turned in; indeed I would expect the design document to evolve as you discover problems and find improvements with your approach.

3.1.4 User document

The user document describes how end-users are to integrate the IP block into their designs—think of it as being like the datasheet you get with a chip.

In particular, the user document should include detailed information on interfacing to the block, i.e., the timing on the different signals. It should describe the power, area, delay numbers at various operating points, and the loading capacitance and drive strengths on the input-output signals.

[1] I like CVS, and have written a very brief tutorial on using CVS which you can read at `www.ece.utexas.edu/~adnan/cvs_notes.txt`

3.1.5 Testing

In this chapter, you are to describe the set of tests you applied to your design to check for logical errors, and your coverage metrics. Classify the bugs you encountered, and how you corrected for them. In addition, discuss the traces you applied to determine the critical path, and compute the delays.

For some projects it may make sense to write a high-level model in C or C++ and do performance simulations (e.g., determine the average latency and drop rate through the Benes fabric as a function of load, and buffering). If this is the case, include results from these simulations.

3.1.6 Optimization

Include a discussion of all the steps you took to improve performance, and the magnitude of improvements that you saw. I am particularly interested in novel techniques that gave your better performance that the descriptions that you based your approach on.

4 Evaluation

Your grade on the project will be based on a number of factors, particularly the originality and quality of the work. Other considerations include clarity of the written report, attention to detail.

You may want to read my notes on technical writing to avoid common mistakes that engineers regularly make in writing—`www.ece.utexas.edu/˜adnan/writing.html`

5 Project list

Below, I give sketches of projects that I would like to see work done on.[2]

Bear in mind that the project description is not complete, and you are responsible for making reasonable assumptions and decisions about the project.

5.1 Optimized MIPS

The text describes a simple implementation of a MIPS processor in Chapter 1 and also Appendix A.10. The goal of this project is to compare a baseline unoptimized implementation of the MIPS processor given in the book to an optimized implementation that uses architectural features such as pipelining/branch prediction/caching/multi-issue, as well as circuit optimizations such as domino logic.

To ensure that your optimized design functions correctly, you will need to turn in a significant verification plan, including functional coverage checks, test-bench, and assertions.

The MIPS project done by Singhal *et al.*, posted on the class page should give you ideas on how to proceed. Naturally, I expect you to go a lot further, since you have their work to get ideas from.

- “Computer Architecture, Fourth Edition: A Quantitative Approach.” John L. Hennessy and David A. Patterson. 2006.

5.2 On-silicon delay characterization

As variability increases, there is growing interesting in making adaptive chips, where parameters such as supply voltage and body biases can be set post-manufacturing to overcome the effects of parametric variation.

The goal of this project is to study the cost and accuracy of on-chip delay characterization structures. I’d like you to survey the state-of-the-art, as well as perform your own experiments.

For example, Dhar *et al.* introduce an adaptive voltage scaling controller that uses an inexpensive ring oscillator to measure speed. There could be multiple ring oscillators placed throughout the design. The gate delay would be approximated based on the delay of the nearest ring oscillator.

[2] You are welcome to suggest your own project; however it must meet with my approval.

Another promising approach would be to implement delay characterization based on Razor by Ernst *et al.* By using a shadow latch and comparator logic, Razor has mechanisms to monitor when a delay error has taken place. In the context of an FPGA, a test input could run through the CLB in successively faster clock cycles until there is a delay error. Additionally, neighboring CLBs could perform the shadow latching and comparator logic need for Razor testing using existing CLB resources.

- S. Dhar, D. Maksimović, and B. Kranzen. Closed-loop adaptive voltage scaling controller for standard-cell ASICs. *International Symposium on Low Power Electronics and Design*, pages 103–107, 2002.
- D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. Kim, and K. Flautner. Razor: Circuit-Level Correction of Timing Errors for Low-Power Operation. *IEEE Micro*, 24(6):10–20, 2004.

The test literature (International Test Conference, Fault-Tolerant Computing) would also be a good place review.

5.3 Pattern matching

The goal of this project is to study hardware engines for pattern matching. The engine is configured with respect to a set of pattern strings $P = \{p_0, p_1, \ldots, p_{n-1}\}$, and it takes as input a string of characters π; it is to determine if the string π matches any of the strings in P.

The check can be performed exactly using a finite state machine; However, this approach leads to large FSMS—assuming 8-bit characters, the transition table is $256 \times S$, where S is the number of states in the FSM (and is roughly equal to the sum of the lengths of the pattern strings).

There are approximate match techniques, and the goal of the project is to determine how effective these techniques are from the perspective of false positives and hardware cost. (These techniques can also be applied to regular expressions, and if you have time you can try to generalize to them.)

Techniques to experiment with: hashing and Bloom filtering.

- Introduction to Algorithms. T. Cormen, *et al.* MIT Press.
- Hashing Techniques for Finite Automata. H. Zeineddine. E382M Term Paper, Spring 2007.
  - I've put Hady's report, as well as two files containing patterns as well as regular expressions here: `http://www.ece.utexas.edu/~adnan/search/`
- Deep Packet Inspection Using Parallel Bloom Filters. S. Dharmapurikar, P. Krishnamurthy, T. Sproull, J. W. Lockwood. Hot Interconnects 11, 2003.

5.4 Subthreshold logic

You can operate a design with supply voltages below the device V_T. This can save power, since it reduces leakage as well as the $C \cdot V^2$ switching energy.

The goal of this project is to determine how effective subthreshold logic is at power reduction. Ideally, I'd like you to design a simple MIPS, and compute the energy consumption for the different components with different supply voltages for various calculations. Array sorting and matrix multiplication are two core computations that you can experiment with.

- A 180-mV Subthreshold FFT Processor Using a Minimum Energy Design Methodology. A. Wang and A. Chandrakasan. IEEE Journal of Solid State Circuits, Jan. 2005.
- Computing with subthreshold leakage: device/circuit/architecture co-design for ultralow-power subthreshold operation. A. Raychowdhury, B. C. Paul, S. Bhunia, and K. Roy. IEEE Transactions on VLSI Systems 13(11).

5.5 TuneFPGA

For this project, you will implement a dual-Vdd FPGA in hardware. Each FPGA CLB will have a mechanism to select either a high-Vdd power supply or a low-Vdd power supply. First, you will need to run schematic level simulations to determine the best implementation of the tuneable FPGA CLB. Then, you will implement the resulting FPGA in hardware. Depending on the size of the group, this project should also implement a complete FPGA architecture, including a routing network, configuration programming infrastructure, a clock network, and reset logic.

- TuneFPGA: Post-Silicon Tuning for FPGAs. Submitted to ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, `www.ece.utexas.edu/~bijansky/fpga.pdf`
- Field Programmability of Supply Voltages for FPGA Power Reduction. F. Li, Y. Lin, and L. He. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(4):752764, 2007.
- The Design of a SRAM-based Field-Programmable Gate Array-Part II: Circuit Design and Layout, P. Chow, S. Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja,, IEEE Transactions on Very Large Scale Integration Systems, vol. 7, no. 3, pp. 321330, 1999.

5.6 Higher arithmetic

We'll discuss the implementation of adders and multipliers in detail in this class. However, we won't talk about how complex functions are implemented.

So if you've ever wondered how a 2$ pocket calculator computes sines, cosines, logs, etc. in the blink of an eye, this is the project for you.

It would be grossly inefficient to use Taylor series expansions for computing transcendental functions. Instead there are much better representations, and CORDIC makes use of one particular representation, which allows sines and cosines to be computed with nothing more than additions, shifts, and a single multiplication (by a constant); it very high precision with very little computational cost ($O(n)$ work for n bits).

The text book talks about computing CORDIC in Chapter 8. You can also read a short, easy to follow account of CORDIC here:

`http://www.worldserver.com/turk/computergraphics/FixedPointTrigonometry.pdf`

Note that the article sidesteps the issue of approximability of angles by the sum $\sum_{i=0} i = N(-1)^{a_i} \cdot \tan^{-1} 2^{-i})$. The approach works because $\tan^{-1} 2^{-i} < 2 \cdot \tan^{-1} 2^{-(i+1)})$, so each successive iteration yields an angle that's less than half of what it was before.

5.7 Model reduction for emulation-based verification

In principle, emulation—the use of reconfigurable hardware to mimic the design—overcomes the performance limitation of simulation. The underlying idea is simple—since gates operate concurrently, an emulator can in a single cycle perform an operation that would take many CPU cycles on simulator.

Emulation, as it is currently practiced, has key limitations of its own.

1. Emulators tend to be capacity limited.
2. The time taken to synthesize and download a design to the emulator is very large.
3. Communication between the emulator and the host is a frequent bottleneck.
4. It is difficult to get visibility into the design.

We would like to study the effectiveness of manual abstraction techniques. As a concrete example, consider replacing a complex scheduler by one which makes choices arbitrarily. This cannot introduce any false positives, since any action the original scheduler could have performed will continue to be an action the new scheduler can implement. From a functional perspective it makes no difference what choice a scheduler makes, and so this replacement does not lead to false positives either.

Other candidates for reduction include: branch-predictors, cache replacement policies, reducing cache/register file size, reducing word-widths, replacing complex arithmetical logic with just an XOR, etc.

One strategy to measure the effectiveness of these reductions would be to implement a DLX-style processor, keeping a record of bugs found, and tests that found them. Then reduce the model using the techniques suggested above, and report on how much faster simulation was, and how many of the original bugs showed up.

- J. Burch and D. Dill. Automatic Verification of Microprocessor Control. In *Computer Aided Verification*, July 1994.
- C. Chang, K. Kuusilinna, B. Richards, and R. Brodersen. Implementation of bee: A real-time large-scale hardware emulation engine. In *ACM Symposium on FPGAs*, 2003.
- S. Chatterjee, C. Weaver, and T. Austin. Efficient checker processor design. In *IEEE Microarchitecture*, 2000.
- D. Chiou, H. Sanjeliwala, D. Sunwoo, Z. Xu, and N. Patil. Fpga-based fast, cycle-accurate, full-system simulators. In *Workshop on Architectural Research using FPGA Platforms*, Austin, TX, February 2006.

5.8 Hardware accelerated Monte Carlo simulation

In its simplest form, an option gives the purchaser the right to buy an object (which could be a stock, or a commodity, we'll assume stock for simplicity) for a fixed price at a given time in the future. More generally, options exist wherein the purchaser can buy the commodity for a fixed price at any point up to a given time, or at the lowest price up to the given time, etc.

When the purchase time is fixed, interest rates are constant, and the object price follows Brownian motion, the Black-Scholes formula gives an analytical way to determine the fair price of the option. This situation is rare, and analytical techniques do not exist for general option pricing.

Monte Carlo simulation can be used to get an idea of the fair price; it is computationally challenging, and the goal of this project is to use hardware acceleration for pricing. It is most natural to use a finite time step for the simulation.

One approach is to derive the exact distribution of the stock price. Given a distribution for a discrete random variable X (the stock price), and a distribution for a discrete random variable Y (its change), the distribution for $X + Y$ is derived by convolving the two distributions—direct convolution can get expensive (quadratic in the range of the two variables), and FFT-based convolution may be a good way to proceed. You may want to consider various distributions for the increment, not just binomial, but something with a heavy tail.

Another approach is to simulate a large number of trials, and determine a distribution based on the trial outcomes.

- `en.wikipedia.org/wiki/Binomial_options_pricing_model`
- Approximate Option Pricing. P. Chalasani, S. Jha, and I. Saias. Algorithmica. 1999.
- Mathematics for Finance: An Introduction to Financial Engineering. Capinski and Zastawniak.
- Randomized Algorithms. Motwani and Raghavan.

6 Miscellaneous

A project of this nature will naturally build upon existing work. You are encouraged to build upon existing code/results, and in your report you may copy/adapt from others papers. However, you must explicitly make it clear that you have done so; failure to report this will be considered **plagiarism** and will be dealt with severely.