VLSI Design

Fall 2020
Final Project Information

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 project should result in a VLSI design with

- Experimental results on the efficacy of the proposed VLSI architectures, and
- Suggestions for improving these architectures.

Projects can be done in groups of 3–4; naturally, I will expect more from projects from larger groups.

2 Timeline

Selecting a project

You need to decide on the project topic soon. Although we have only covered some of the information you need to complete the project at this time in the semester, this deadline is intended to get you started. You only need to decide on **what** you will be designing, and you'll learn the **how** by the time you need to implement your project. For the topic, you could think of interesting areas where VLSI chips can be applied (just focus on the digital parts of the real-world applications), or base the topic on something that interested you in your research or work experience.

You only need to turn in a sheet of paper with your topic, a brief description of what you would like to implement, and the names of the teammates at for the project outline. For the final project report, submit a brief report (one per team), with design files, simulation data, etc. by Dec. 15, 2017. The report could be in the form of a presentation (slides) if you wish. I will work with each of the teams to converge on the final deliverables. Please don't worry about the project being too big (or complex); we'll carve out a piece that can be implemented in a semester, and from which you will learn a great deal.

Final report

The final report should include the following.

- Specifications document
- Design document
- User document

- Testing strategy and results
- Optimization strategy and results
- Source code and layout

Details of each of these items are given below.

Deadlines

- Week of Sept. 17, 2020: Teams meet with Prof. Abraham to discuss specification of project, roles of team members in project
- Oct. 8, 2020: Detailed outline of project, deliverables
- Week of Nov. 12, 2020: Teams meet with Prof. Abraham with Interim project report (Preliminary design document, results of design review)
- Dec. 8, 2020: Final project report; projects should have been demonstrated to the instructor or TA previously (lab sessions are a good venue to demonstrate the projects)

3 Details of Final Report

3.1 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 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.

3.2 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, 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 will not have a formal review process, since it is probably too involved for a class project, and the class size is too large. However, each team should conduct their own reviews.

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.

3.3 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.

3.4 Testing

In this section, 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.

3.5 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 on which based your approach.

4 Project Topics

The following is a list of topics compiled from faculty, graduate students and projects in previous semesters. I encourage you to pursue your own ideas, and talk to me if you have any that are not listed here.

4.1 SRAM with Sleep Transistors

One of the major techniques used to control sub-threshold leakage is using sleep transistors. In essence, sleep transistors are used for power gating. Logic runs at low Vt, and the gates are faster and leaky; sleep transistors are high Vt. They are switched off when idle (usually NMOS alone is used) and can save 2-1000$\times$ leakage power.

Your goal is to design a 32 kbit SRAM (128 rows, 256 columns, 8 bit words) which uses sleep transistors to reduce leakage power. There are a number of ways you can go: a single huge sleep transistor, a sleep transistor per cell, or a sleep transistor per 4 cells, etc. There are power–delay tradeoffs between these, which you should explore.

Some papers that may help:

- B. Mohammad, M. Saint-Laurent, P. Bassett and J. A. Abraham, "A Cache Design for Low Power and High Yield," *ISQED*, 2008, pp. 1-10.
- B. H. Calhoun, F. A. Honore and A.P. Chandrakasan, "A Leakage Reduction Methodology for Distributed MTCMOS," *IEEE JSC*, vol. 39, May 2004, pp. 818-826.
- A. Ramalingam, B. Zhang, D. Z. Pan and A. Devgan, "Sleep Transistor Sizing Using Timing Criticality and Temporal Currents," *Proc. Asia South Pacific Design Automation Conference (ASPDAC)*, vol. 2, Jan. 2005, pp. 1094-1097.
- S. Vangal, M. Anders, N. Borkar and E. Seligman, "5-GHz 32-bit integer execution core in 130-nm dual-Vt/CMOS," *IEEE Journal of Solid State Circuits*, Nov. 2002, pp. 1421-1432.
- V. Khandelwal and A. Srivastava, "Leakage Control Through Fine-Grained Placement and Sizing of Sleep Transistors," *IEEE Transactions on Computer-aided design of integrated circuits and systems*, 2007, pp. 533-536.

4.2 Divider

Division is the hardest of the four basic arithmetic operations. As Intel famously demonstrated, implementation of division algorithms is not always straightforward. Nonrestoring, SRT, Newton-Raphson, and Goldschmidt are all possible methods for the implementation of division.

For this project, review all of these methods. Select at least one to implement. Provide both hand worked examples and simulation results demonstrating the correctness of your approach and implementation. Additionally contrast your implementation with each of the other methods and provide the reasons for your selection, as well as the advantages and disadvantages provided by your approach.

Some references:

- B. Parhami, "*Computer Arithmetic*," *Oxford University Press*, 2000.
- J. E. Robertson, "A new class of digital division methods," *IRE Trans. Electronic Computers*, 1958, pp. 218-222.
- K.D. Tocher, "Techniques of multiplication and division for automatic binary computers," *Quarterly Journal of Mechanics and Applied Mathematics*, 1958.

4.3 Fast Fourier Transform Kernel

During the last 10 years, a lot of effort has been concentrated on mapping the FFT architecture to silicon while making tradeoffs in performance, silicon area, the number of I/O pins, and other manufacturing issues. The objective of this project is exploring a FFT architecture and implement it.

Implementation. The chip should take the time-sampled data input at a set sampling frequency of your choice and output the correct bin counts for all the points in the FFT, within the range of error tolerance. For real-world applications, you are encouraged to aim for 64/128-point high precision FFT kernels which are compatible with the wireless industry's protocols. However, to keep the project simple, 16/32-point precision would be good enough. Make your own specification for precision, power, area, and I/O bit width. After completing the FFT design, verify it by following the next paragraph.

Testing. Create a testing benchmark structure to test the FFT core with reasonably good coverage using all types of signals (sine waves, noise, dc) and their random combinations, below the chosen Nyquist frequency.

At the algorithm level, you may choose from radix-2, radix-4, and specialized FFT implementations, etc. The final chip must be presented in layout level after synthesis, and a code file alone would not be sufficient.

Project deliverables will include the FFT specification definitions, test files, testing benchmark, test outputs reports (both simulated and physical), a Cadence layout of the FFT hardware, code, or a schematic abstraction of your layout, and a report of your algorithm (in the form of a paper or pseudo-code).

Below are several references on FFT hardware implementations.

- E. E. Swartzlander, W. K. W. Young and S. J. Joseph, "A radix 4 delay commutator for fast Fourier transform processor implementation," *IEEE Journal of Solid-State Circuits*, vol. 19, Oct. 1984, pp. 702-709.
- K. Maharatna, E. Grass and U. Jagdhold, "A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM," *IEEE Journal of Solid-State Circuits*, vol. 39, Mar. 2004, pp. 484-493.
- M. Sala, F. Salidu, F. Stefani and C Kutschenreiter, "Design Considerations and Implementation of a DSP-Based Car-Radio IF Processor," *IEEE Journal of Solid State Circuits*, Jul. 2004, pp. 1110-1118.
- S. Ishiwata and T. Yamakage, "A single-chip MPEG-2 codec based on customizable media embedded processor," *IEEE Journal of Solid State Circuits*, Mar. 2003, pp. 530-540.

4.4 All Digital PLL

Phase-locked loops (PLLs) are used to recover timing information from a signal—they are ubiquitous in communications, and are also used for timing recovery on boards and chips. Analog PLLs are very hard to design because they use feedback, and are very sensitive to noise and operating parameters.

The goal of this project is to design an “all digital PLL” which is an implementation of the PLL with all digital components, and compare its performance (measured in lock time and phase noise) and costs (in terms of area, power, delay) to a traditional analog PLL.

References:

- R. J. Baker, H. W. Li, D. E. Boyce, “CMOS Circuit Design, Layout, and Simulation,” *IEEE Press*, 2007.
- R. B. Staszewski et al., “A First Digitally-Controlled Oscillator in a Deep-Submicron CMOS Process for Multi-GHz Wireless Applications,” *IEEE Dig. RFIC Symp.*, June 2003, pp. 81-84.
- R. B. Staszewski et al., “Digitally-Controlled Oscillator (DCO)-Based Architecture for RF Frequency Synthesis in a Deep-Submicron CMOS process,” *IEEE Trans. Circuits and Systems II*, vol. 50, Nov. 2003, pp. 815-828.
- R. B. Staszewski, et al., “All-digital PLL and transmitter for mobile phones,” *IEEE J.SSC*, vol. 40, Dec. 2005, pp. 2469-2482.

4.5 Design of Robust CMOS Circuits for Soft-error Tolerance

With the continued scaling of technology, lower supply voltages and increasing operating frequency, integrated circuits become increasingly susceptible to single event upsets (SEU) caused by transient noise or high energy particles. A SEU may cause a bit flip in some latch or memory element, thereby altering the state of the system, leading to a ‘soft error’. The main objective of this project is to introduce more robustness with some redundancy in circuits to make them less susceptible to undesired errors. The focus is to explore various circuit level as well as system level techniques to reduce the effect of soft errors for logic and memory circuits.

Some of the references are:

- R. C. Baumann, “Soft errors in advanced computer systems,” *IEEE Des. Test. Computer*, vol. 22, May/Jun. 2005, pp. 258-266.
- R. C. Baumann, “Radiation-Induced Soft Errors in Advanced Semiconductor Technologies,” *IEEE Trans. Device and Materials Reliability*, vol. 5, Sept. 2005, pp. 305-316.
- A. U. Diril, “Circuit Level Techniques for Power and Reliability Optimization of CMOS Logic,” *PhD Dissertation, Department of Electrical and Computer Engineering, Georgia Institute of Technology*, May 2005.

4.6 FIR Filter

Digital filtering is one of the fundamental operations of digital signal processing. Digital filters can be used to improve signal quality, separate multiple signals, or extract information from within a signal. Applications include audio processing, video processing, and image processing.

A basic FIR filter implements the following function:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

Filter design involves the following steps.

- Determination of the requirements for the filter
- Filter coefficient calculation
- Realization of the filter structure
- Analysis of the filter performance
- Hardware implementation

For this project, select a digital filtering application, then research the current state of research for these filters using IEEE Xplore. Recent issues of the IEEE Transactions on Solid State Circuits are a suggested starting point. You may implement an improvement to a published filter design, or implement two (or more) published designs contrasting and comparing the tradeoffs between the designs.

Your report should include discussion of the specifications that your filter design will meet such as data rates and frequency. A description of the coefficient calculation and a graphical representation of the structure of your filter design should be included as well as a discussion of the word length choice. Finally the hardware implementation and testbench should be provided and discussed in detail.

A general reference:

- R. E. Crochiere and L. R. Rabiner, "Interpolation and decimation of digital signals - A tutorial review," *Proceedings of the IEEE*, vol. 69, March 1981, pp. 300-331.

Some recently published research:

- Y. Wang and K. Roy, "CSDS: A New Complexity Reduction Technique for Multiplierless Implementation of Digital FIR Filters, " *IEEE Transactions on Circuits and Systems - I*, vol 52, September 2005, pp. 1845-1853.
- F. Bruekers and T. Kalker, "Reduction of Symmetric Complex Filters, " *IEEE Transactions on Signal Processing*, vol 58, January 2010, pp. 200-208.

4.7 On-chip Interconnection Network

In this project, you will get an in-depth understanding of the VLSI design of modern on-chip interconnection network. To begin with, the following article serves as a good introduction: "Architectural Choices in Large Scale ATM Switches," by J. Turner and N. Yamanaka, in the IEICE Transactions, 1998.

The major task of this project is to select and implement a switching architecture. For instance, in the article above, a Batcher-Banyan based, self-routing network is chosen. Many techniques have been proposed; please spend some time on selecting among them. You are encouraged to invent new architectures and algorithms and analyze their strengths and drawbacks. Here are some more articles that may be useful.

- Shin and Hodges, "A 250-Mbit/s CMOS Crosspoint Switch," *IEEE JSSC*, vol. 24, April 1989, pp. 478-486.
- Akata et al., "A 250-Mb/s CMOS Crosspoint LSI for ATM Switching," *IEEE JSSC*, vol. 25, Dec. 1990, pp. 1433-1439.
- Chemarin et al., "A High-speed CMOS Circuit for 1.2-Gb/s 16x16 ATM Switching," *IEEE JSSC*, vol. 27, July 1992, pp. 1116-1120.
- O'Neill et al., "A 200Mhz CMOS Broad-Band Switching Chip," *IEEE JSSC*, vol. 28, March 1993, pp. 269-275.

Please actively search IEEE Xplore or Google for new ideas and build upon them!

Once the architecture is defined, you may employ the skills developed in the labs to implement a prototype (physical level). In view the limited time, you may put most of the efforts on the core algorithm and structure and size down the whole system. Please consider how to establish your testing benchmark of your switch from the very beginning. Again, your testing benchmark should have fairly good coverage. As to the benchmark setup, you may use C/C++, or scripture languages like TCL/TK, PERL, etc. As this is more in the flavor of an open topic, your final grade will be based on your ideas, implementation workload, and testing mechanisms, etc. Especially, your implementation should demonstrate fair workload worthy of a serious project in our graduate class.

4.8 Design of Circuits for Sub-threshold Voltages

For ultra low power and portable applications, design of digital subthreshold logic has been investigated with transistors operated in the subthreshold region (supply voltage corresponding to logic 1, which is less than the threshold voltage of the transistor). In this technique, the subthreshold leakage current of the device is used for computation. Standard design techniques suitable for superthreshold design can be used in the subthreshold region. However, it has been shown that a complete co-design at all levels of hierarchy (device, circuit, and architecture) is necessary to reduce the overall power consumption while achieving acceptable performance (hundreds of kilohertz) in the subthreshold regime of operation. Your goal in this project is to choose a suitable application, such as an adder, multiplier, FFT module etc, and implement it using sub-threshold voltage logic.

Some references:

- A. Wang and A. Chandrakasan, "A 180mV FFT Processor Using Subthreshold Circuit Techniques," *Proc. IEEE ISSCC*, Feb. 2004, pp. 292-529.
- H. Soeleman and K. Roy, "Ultra Low Power Digital Sub-Threshold Logic", *Int'l Symp. Low-Power Elect. and Design*, Aug. 2009, pp. 94-96.

- H. Soeleman and K. Roy, “Digital CMOS Logic Operation in the Sub-Threshold Region”, *IEEE Great Lakes Symp. VLSI*, March 2000, pp. 107-112.
- H. Soeleman, K. Roy, and B. Paul, “Robust Sub-Threshold Logic for Ultra-Low Power Operation”, *IEEE Trans. VLSI Sys.*, Feb. 2001, pp.90-99.
- B. Paul, H. Soeleman, and K. Roy, “An 8X8 Sub-Threshold Digital CMOS Carry Save Array Multiplier”, *European Solid State Circuits Conf.*, Sept. 2001. pp. 377-380.

4.9 Implement a Sub-threshold Voltage Cell Library

A standard cell library (SCL) contains the basic building blocks for designing an integrated circuit. It has a fixed set of well-characterized logic blocks. Once an integrated circuit is built using the library, the behavior of circuit can be determined based on information within the individual cells from the library. This information includes parasitic capacitance, area, and delay. In order to qualify as a standard cell library, it has to include NAND, NOR, inverter, and D flip-flops. SCLs are commonly employed by Application Specific Integrated Circuit (ASIC) designers due to robustness and flexibility of the library, resulting in quick turnaround times.

This purpose of this project is to build a low power standard cell library using sub-threshold voltages. One direction is to base the library on the 45nm technology library available for the course. Another possibility is to base it on a TSMC 0.18μ library (you would need to discuss this with the instructor and sign an NDA). Your report should explain in detail how you created the library. Some supporting material is available from past projects (see the reference below).

Some references:

- S. Sundareswaran, J. A. Abraham, A. Ardelea, and R. Panda, “Characterization of Standard Cells for Intra-Cell Mismatch Variations,” *ISQED*, 2008, pp. 213-219.
- H. Soeleman and K. Roy, “Ultra Low Power Digital Sub-Threshold Logic,” *Int’l Symp. Low-Power Elec. and Design*, Aug. 1999, pp. 94-96.
- H. Soeleman and K. Roy, “Digital CMOS Logic Operation in the Sub-Threshold Region,” *IEEE Great Lakes Symp. VLSI*, March 2000, pp. 107-112.

4.10 Complex Arithmetic Functions (CORDIC)

In this class the implementation of adders and multipliers are discussed in detail. However, there is no discussion of how complex functions are implemented. 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 provides very high precision with very little computational cost (O(n) work for n bits).

The text book talks about computing CORDIC in Chapter 8. And, note that the reference article sidesteps the issue of approximability of angles by the sum $\sum_i = N(-1)^{a_i} \cdot tan^{-1}2^{-i}$. The approach works because $tan^{-1}2^{-i} < 2 \cdot tan^{-1}2^{-(i+1)}$, so each successive iteration yields an angle that is less than half of what it was before.

References:

- `http://en.wikipedia.org/wiki/CORDIC`
- J. E. Volder, “The CORDIC Trigonometric Computing Technique,” *IRE Trans. on Electronic Computers*, 1959, pp. 330-334.
- S. Abdulla, N. Haewoon, M. McDermott and J. Abraham, “High Throughput FFT Processor with no Multipliers” *Proceedings, Int’l Conference on Computer Design (ICCD)*, 2009, pp. 485-490.

4.11 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.

Some references:

- `http://en.wikipedia.org/wiki/Binomial_options_pricing_model`
- P. Chalasani, S. Jha, and I. Saias, “Approximate Option Pricing,” *Algorithmica*, 1999.
- M. Capinski and T. Zastawniak, “Mathematics for Finance: An Introduction to Financial Engineering,” *Springer*, 2003.
- R. Motwani and P. Raghavan, “Randomized Algorithms,” *Cambridge University Press*, 1995.

4.12 Synthesis of Niagara Processor Core

The intent of this project is to do a top-down design of a synthesizable block in the processor core and to achieve low power. One of the following blocks in SPARC-T1 core from SUN will be used as a target module for this project: EXU (Execution Unit), LSU (Load Store Unit) and IFU (Instruction Fetch Unit). You are expected to design it using 45nm technology. The project activities will include reducing leakage and dynamic power, and estimating area.

References:

- `http://www.opensparc.net/opensparc-t1/download.html`
- `www.opensparc.net/pubs/t1/docs/OpenSPARCT1_DVGuide.pdf`
- `users.ece.utexas.edu/~mcdermot/vlsi-2/OpenSPARCT1_Micro_Arch.pdf`

4.13 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. You should 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.

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

References:

- R. Tayade and J. A. Abraham, “On-chip programmable capture for accurate path delay test and characterization,” *Int’l Test Conf.*, 2008, pp. 1-10.
- S. Dhar, D. Maksimovic and B. Kranzen, “Closed-loop adaptive voltage scaling controller for standard-cell ASICs,” *Int’l Sym. Low Power Elec. and Design*, 2002, pp. 103-107.
- 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*, vol. 24, 2004, pp. 10-20.

4.14 Comparison of Circuit Families

There are many kinds of circuit families used in digital systems. One of the famous logic families is static CMOS. It has good noise margins, is fast, consumes relatively lower power, insensitive to device variations, easy to design, and widely supported by CAD tools. Other circuit families are also used for the high speed operation. For example, the dynamic circuit families were used in high performance processors.

Compare Static CMOS, Pseudo-nMOS, CVSL, Dynamic, Domino, Dual-rail Domino, CPL, DCVSPG and so on with different criteria: the number of transistors (area), static power consumption, ability to cascade (compose), robustness, and the existence of dynamic nodes. Find suitable applications for each of them by using HSPICE.

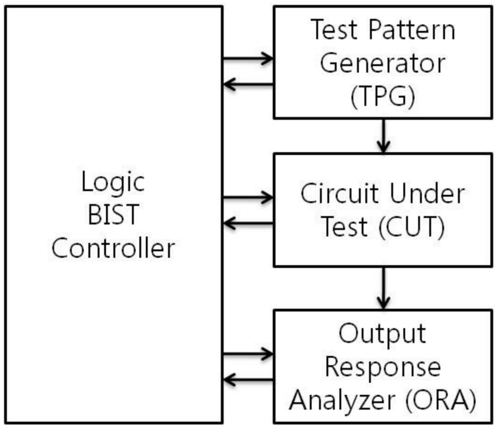
The textbook, Chapter 6 would be a starting point.

Mentor:

4.15 Logic Built-In Self-test (BIST)

With development of the VLSI technology, the process scale goes down. Moreover, the logic becomes highly complex. In this situation, it is difficult to test the VLSI logic. It is because the test stimuli through external pins of the chip cannot access to the internal logic of the chip completely.

Furthermore, the outputs of the complex sequential logic are determined based on its current state. Thus, the complete test for the chip through its external pin may be impossible. To solve these problems, several testing methods were suggested such as ad-hoc, scan and BIST. BIST has an advantage against them, especially for complex systems on a chip. BIST is an inexpensive testing method with the high fault coverage. The following picture shows a typical BIST system.



You could start with the information on test in the textbook. Some other references are:

- S. Hwang, J. A. Abraham, "Optimal BIST Using an Embedded Microprocessor," *IEEE ITC*, 2002. pp. 736-745.
- A. Chatterjee and J. A. Abraham, "Test generation, design-for-testability and built-in self-test for arithmetic units based on graph labeling," *J. Electronic Testing*, 1999, pp. 351-372.
- R. Dandapani, J. H. Patel and J. A. Abraham "Design of Test Pattern Generators for Built-In Test," *IEEE ITC*, 1984, pp. 315-319.
- L.-T. Wang, C.-W. Wu and X. Wen, "VLSI Test Principles and Architectures: Design for Testability," *Morgan Kaufmann*, 2006.

- M. Abramovici, M. A. Breuer and A. D. Friedman, "Digital Systems Testing and Testable Design," *IEEE Press*, Piscataway, NJ, 1994.

4.16 Sequential Circuit Design

If one can design sequential logic which is 1.3x faster, this would represent 10%-15% performance improvement. This is why minimizing sequential overhead is important in pipelined systems. Sequential circuits commonly used for synchronization in pipelines are flip-flops and latches.

The intention of this topic is designing your own optimized sequential elements. Please refer to some sequential logic such as hybrid-latch flip-flop, semi-dynamic flip-flop and sense amplifier flip-flop. You can characterize your circuit and measure its setup time, hold time, clock to Q delay, data to Q delay, PDP and EDP.

Make an effort to design a low-power or high performance one. It will also be a good experience for you to compare the latch based design and the flip-flop based design with your own elements.

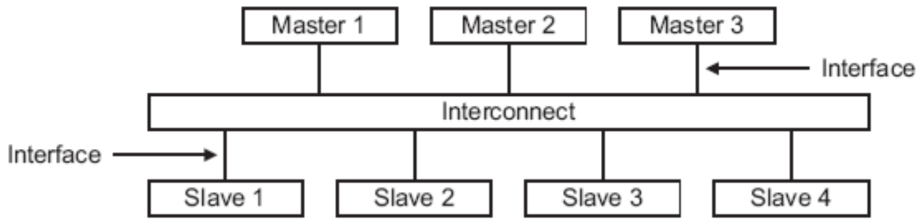
References:

- S. Sundareswaran, R. Panda, J. A. Abraham, Y. Zhang and A. Mittal, "Characterization of sequential cells for constraint sensitivities," *Int'l Symp. Quality Elec. Design*, 2009, pp. 74-79.
- V. Stojanovic and V. G. Oklobdzija, "Comparaive Analysis of Master-Slave Latches and Flip-Flop for High-Performance and Low-Power Systems," *IEEE J.SSC*, 1999, pp. 536-548.

4.17 Implementation of AMBA3 AXI

The complexity of the embedded systems has increased dramatically, and they need a high speed and smart bus system. The reason is that in an embedded system, the communication architecture such as the bus plays an important role in orchestrating data and control signal transactions among the system components. Recently, the AMBA 3 bus protocol was introduced by ARM Inc., and it provides several advanced features, such as multiple outstanding requests, which accelerate its system performance by reducing the bus-waiting time for each component.

The objective of this project is to design your own AMBA AXI interconnect. The AMBA3 AXI bus system needs several elements such as decoder, arbiter, slave interface, and master interface. It is very similar to the wishbone bus in that it is using a handshaking protocol. The following figure shows the interface and interconnection of the bus system.



References:

- ARM Ltd., “AMBA®AXI Protocol v1.0 Specification,” March 2004.
- ARM Ltd., “AMBA®specification (rev2.0),” 1999.
- M. Posner and D. Mossor, “Designing Using the AMBA^{TM}3 AXI^{TM} Protocol,” *Synopsys*, April 2005.

4.18 Specialized hardware for Artificial Intelligence

According to Wikipedia, an AI accelerator is a class of specialized hardware accelerator or computer system designed to accelerate artificial intelligence applications, especially artificial neural networks, recurrent neural network, machine vision and machine learning. This is a very broad and growing topic. New accelerators are being proposed for different applications, for increased performance and reduced power consumption.

In order to make the project feasible, the team should pick one class of accelerators and design it (ideally using a Hardware Description Language, Verilog for example). A couple of very recent tutorials (below) can serve are the start.

References:

W. J. Dally, C. T. Gray, J. Poulton, B. Khailany, J. Wilson and L. Dennison, ”Hardware-Enabled Artificial Intelligence,” 2018 IEEE Symposium on VLSI Circuits, Honolulu, HI, 2018, pp. 3-6, doi: 10.1109/VLSIC.2018.8502368.

Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, Jeremy Kepner “Survey of Machine Learning Accelerators” Arxiv:2009.00933.