VLSI at USC
Prepared by Melvin A. Breuer
Charles Lee Powell Professor of Electrical Engineering and Computer Science
June 2010

Preface: Over the past 45 years I have advised many students regarding courses to take, helped others in getting their first engineering job, and discussed what areas are "hot." As to the latter topic, I never give a student an opinion on what area seems to have the largest growth, or where jobs will be plentiful in 2-3 years. My advice in what a student should pursue is usually: (a) work in an area that you think you will enjoy and where you think you have good insight and understanding, and (b) strive to be as good in this area as you can be with the talents and ability you have been given. But I also get somewhat easier questions to deal with like: "What is VLSI about?" or "What is EEXXX about?" or "Is EEYYY hard and does it have a project?" Well, to answer some of these questions I have prepared this document. It is not an official USC document; it is only my opinion and in some cases recommendations. But maybe it will help new M.S. and Ph.D. students get started in our EE VLSI program.

1. Introduction

There have been several technological revolutions over the last 2000 years, including the invention of printing press in the 15th century, the Industrial Revolution of the 18th century when the world turned from an agrarian society to one more oriented to manufacturing, mining and transport. Currently we are at the beginning of the *information technology revolution* that started in about 1960, made commercially successful by the invention of transistors and integrated circuits, and now referred to as the age of very large scale integration (VLSI).

In 1959-60 I was an M.S. Student at UCLA. I went to Boston for an interview at M.I.T. The plane flew at about 600 mph. Last year I went to Washington D.C. Guess what! Again I flew at 600 mph. Has flying gotten better in these 50 years? Well maybe safer and more fuel efficient, but back then I had more leg room, better and free food, no charge for luggage, and was not treated as cattle. Where is the progress?

Work on my on-campus job and M.S. Thesis required that I do a lot of Fortran II programming. Since there were few computer users on campus in those days, I often had exclusive access to the IBM 650 from 2 to 4 AM. Of course, today I have instantaneous access to computing power and storage that is at least 8 orders of magnitude large than I had back then. Unlike other revolutions that depend on a quantum jump in technology, such as the invention of the telephone and discovery of AC electricity, the IT revolution is unique in that every 15 months or so the hardware you recently purchases becomes obsolete in terms of the latest performance. Clearly the airline industry is still waiting for a technological revolution.

The Ming Hsieh Department of Electrical Engineering here at the Viterbi School of Engineering offers a stellar program in VLSI Systems and Circuits Engineering (VLSI-S&CE). Many of the faculty have received teaching and research awards, and have

written textbooks used around the world. The tenure track faculty is complemented by a full-time teaching staff who have extensive industrial background and “teaching genes”, as well as researchers from our sister institution, the Information Science Institute (ISI) situated in the Santa Monica Marina, and home to the VLSI MOSIS project.

Below I summarize the components involved in the design and manufacturing of a VLSI-based system, and the core courses in our curricula that train students to be successful engineers in these areas. I hope that this memo will help you in making good decisions regarding your carrier and studies at USC.

2. System flow

Figures 1 and 2 indicate simplified versions of the system flow involved in creating a new digital VLSI ASIC chip. This is primarily a top-down process, while the design of fully custom chips pays more attention to physical design consideration throughout the process and generally involves more designer-driven design rather than RTL synthesis of critical components.

Specification: The process starts with a system specification (spec) of the functionality of a chip. In addition, the spec usually includes information and/or constraint attributes such as technology, packaging, power dissipation, die temperature, clock rate, manufacturing cost, and performance. Often non-system designers develop part of these specs, usually in response to detailed discussions with potential customers. For example, the spec for the electronics in a new hearing aid might be developed in part by an expert in digital signal processing and an otologist. While our curricula does not directly address issues associated with the development of specs, needless to say that background in other areas of EE, such as communication theory and signal and image processing are very useful. One reason for this is that while over 90% of chips *manufactured* each year may be characterized as being general purpose microprocessors, over 90% of chips *designed* each year are not microprocessors.

Temporal and logical synthesis with constraints and goals

System Synthesis: The first step in the design process is the human transformation of an implementation independent specification and the functionality and performance specs into a flow of both parallel and sequential conditional computational steps e.g., data and control flow graphs. Key tasks during this process are (1) the selection of a suitable architecture, (2) decomposing the required functionality over the components of this architecture including decision about which functionality is implemented in hardware and which in software (to be run on a general purpose or embedded processor), and (3) scheduling the various activities so as to meet the performance specs. Part of this process deals with determining and analyzing various algorithms for instantiating the desired behaviors. For example there are many ways of building a Viterbi decoder. In addition to the algorithmic solution, one must also consider the non-behavioral aspects of the spec, such as imposed constraints and performance objectives. Since the actual design space is quite vast, automation is often used to prune the space and to predict

values of attributes such as gate count and power dissipation. Often languages such as hardware C or System C are used. The next step in the design process, namely behavioral synthesis, is highly intertwined with system synthesis.

Behavioral Level Synthesis: The next step is to take the system description as a collection of interacting machine descriptions or data and control flow graphs and produce the system architecture in terms of what components are used to perform required operations, how much sharing is desirable or possible, how are the various operations scheduled, where the intermediate results of computation are stored, how these components communicate with each other, etc.

The three main problems typically addressed in behavioral synthesis systems are allocation of resources, scheduling of processes, and binding of variables to hardware entities.

The binding of variables to hardware entities includes defining the bit-width of various elements. Some bit-widths may be determined at the higher behavioral level, but sometimes domain-expert algorithm designers work at the behavioral level and leave bit-width analysis to the hardware designers.

Often languages such as Verilog, VHDL and System C are used to describe the behavior of a chip.

Register Transfer Level Synthesis: System and behavioral synthesis are somewhat like programming in a high level language and applying compiler-like transformations to optimize the program (in terms of its size or its execution time); you have all the variables, memory and operators you want. But a chip consists of a finite number of gates, flip-flops and memory, though this finite number is getting to be in the billions. Thus a behavioral description must be transformed into one that relates variables, constants and operators with hardware entities such as registers, memories, busses and blocks of logic. This transformation process, which over the past 15 years has been partially automated, is referred to as register-transfer-level (RTL) synthesis. At this point various computer-aided design (CAD) tools can be employed that predict attributes of the final circuit if the design process were to be continued. Some of these attributes include power dissipation, gate count, area, yield and performance. So, at this point, a designer can make an intelligent decision as to whether to continue refining this design, or consider alternative behavioral and/or register solutions.

The inputs to this step are a set of operations described as transfer of values between registers and functional units through a hierarchy of interconnect structures including buses and multiplexers under the command of a hardware controller. The outputs of this step are blocks of combinational logic separated by sequential circuit elements (latches and flip-flops). An important task performed in this step is the encoding and realization of the controller block (specified as a finite state machine or a collection of interacting machines) in hardware. Here is where your knowledge of FSM's is essential.

Logic synthesis: Logic synthesis is the process of taking what we normally think of as a logic design consisting of gates, flip-flops, and clusters of flip-flops referred to as registers, and manipulating this digital circuit so that it retains its functionality, but now satisfies various constraints and optimizes various objectives. First, you need to recognize that 2-level AND-OR on NAND-NAND logic is not encountered very often in real circuits. Most logic structures are multi-level, and include both primitive and complex gates. Area, testability and performance (delay) are of prime concern. Thus, logic synthesis, which also is a highly automated process, consists of operations such as re-timing, library mapping/binding, fan-out optimization and gate sizing.

If the area, performance, or power does not meet your specifications, then re-design at the RTL or behavioral level is needed.

Well, at this point we finally have a design in terms of things we understand, namely gates and flip-flops. Now lets go get out our soldering iron and build it in hardware.

Physical synthesis

Physical design is the process of mapping the output from temporal and logic synthesis into a form that can be sent to a factory for manufacturing. The end product of physical synthesis is actually a set of masks in the form of a GDSII file. A mask is somewhat analogous to a film negative that has millions of black rectangles exposed on a transparent sheet. One main difference is that a set of masks costs several millions of dollars, so you do not get many chances to “capture that moment.” Physical synthesis consists of several processes that are mostly automated, usually highly inter-related, but for simplicity, described here as separate entities.

Partitioning: Normally a design is created where some objectives are the focus of attention, but when the design is mapped into silicon, other factors become more important, such as chip area, power dissipation, signal delay and testability. One of the first steps in physical design is to partition the circuit into manageable units. The simplest example is to partition a design into several chips if it cannot “fit” onto one chip. On-chip partitions might aggregate circuitry that runs off of a common clock, circuitry that requires a voltage different from the rest of the logic, or circuitry that is more susceptible to noise.

Floorplanning: The layout of a chip is similar to the floorplan of a large home: there are usually several bedrooms, a living room, a dining room off of the kitchen, and a master bathroom, another for children and one for guests. These must be strategically placed according to total areas, the area and aspect ratio of the property, convenience, etc. The same is true for a chip, where, for example, a 64-bit ALU can be layed-out many different ways. Chip floorplans consists of allocating specific resources and budgets to the various large modules in a design, such as control, I/O, image processing, ALU, wireless trans-receiver, and memory. These allocated entities consists of items such as

area, aspect ratio, location of I/O terminals, power, and available metal layers for interconnect.

Placement: Eventually elements such as transistors, gates and flip-flops must be assigned to physical coordinates on the surface of a die so that they are electrically isolated from one another, yet they can be electrically interconnected so that they implement the desired specification and the die area is near minimal. A simplified version of the placement problem is: given 1 million rectangles of various aspect ratios, place them inside of a square boundary so that no rectangle overlaps another one, and the area of the square is minimal.

Interconnect (routing): What makes the placement example given above a simplified case of the VLSI placement problem is that an additional objective to be achieved is a placement so that all terminals of each and every signal net can be made electrically common. This wiring of nets must be achieved while (1) not shorting any signal to another one, (2) minimizing the maximum signal propagation delay, (3) avoiding crosstalk, and much more. In some cases, there will be areas of the physical design that are limited by interconnect and wire-planning must be done in conjunction with floorplanning and placement.

Miscellaneous Processes

Library development: Normally placement and interconnect processes deal with a "gate" level model of circuit elements. In reality, signals are processed via transistor circuits, and transistors and interconnect are defined by a set of rectangular images associated with various layers of integration, such as metal 1, 2, 3, …, polysilicon and oxide. Library development consists of developing a family of circuits, e.g., that all implement a 3-input NAND function, but have different delay and drive capability. While much of this process is done manually, many tools exist to analyze these designs in terms of electrical and logical characteristics. One popular tool is SPICE (circuit-level) simulation. The results of this analysis is that the each gate in the library has a variety of abstract "views" to support timing analysis, digital simulation, physical design and verification.

Mask generation: Once all elements have been placed and interconnected, the elements are replaced by their library images and the entire design is converted into a GDSII file. Due to the fact that feature sizes of transistors and wires are in the range of 45 nm, and the wavelengths of light produced by lasers used in transferring these images onto silicon are comparable in scale, manufactured features differ from mask features. Thus a new field has emerged dealing with such concepts as double-patterning and optical proximity correction (OPC).

Testing: Gordon Moore noted that a major shift in scaling of VLSI feature sizes occurs about every 18 months. Over the last few years we have gone from 65nm in 2007, to 45 nm in 2010 and soon to 32 nm in 2013. Why didn't we just skip the 45nm technology node and come out with 32 nm instead? Well the answer is that it is very difficult and

costly to move from one node to another in terms of research, development and manufacturing equipment. Simply put, it is very costly. Thus, while researchers are currently working on 22nm and 14 nm nodes, the yields are still very low. That is, if one manufactured one million transistors, few if any would work as desired. It usually takes several years of tuning a technology to learn how to increase the yield to very high levels; this is the process known as “yield learning”. I leave it to the reader to calculate the minimum required probability of manufacturing a good transistor so that if a chip has one-billion transistors, the probability of all of them are good is 0.5. Well the answer looks something like 0.999…9. As a result, it is estimated that from 30-70% of complex (multi-million transistors) chips manufactured today using the latest technology nodes are defective. Yet they all look the same. Manufacturing test is the process of separating out the good chips from the bad ones. Again, most aspects of test development and testing itself have been automated.

Verification: Going back once again to my days at UCLA, as it turned out, near the end of my tenure many people had learned how to program so the University got new machines and reverted to a close shop mode of operation. In this situation, I was forced to write my program in my lab, use a key punch machine to transcribe my program onto punch cards, walk my deck of cards across campus and leave it at the IN window. The next day I would return and if lucky my deck was there with a print out telling me all of my syntax errors. I would correct these and resubmit. Returning the next day, there might be nothing for me. Returning the following day I might have the results of a run, but because of a programming error I had to again resubmit my program. Thus weeks could go by before I would get a successful run. This process taught me the importance of writing a program correctly the first time. And yes, one day I had to write a program in SIR, an assembly language that supported floating-point operations, and guess what, it worked properly the first time.

Unfortunately, life is too easy now, and designers get instantaneous responses to their programs. Thus designers tend to work in the mode of trial and error. Hence each of the items mentioned above are subject to human errors and must be checked. Thus verification is a primary tool associated with design. Verification must address many aspects of a design, such as its temporal attributes, its electrical characteristics, naturally the logic itself, and finally the physical characteristics. Properties must be checked that were not intended to be part of the design, such as simultaneous switching noise, and ringing due to inductance. Verification is one of the last frontiers in CAD that has not been successfully automated.

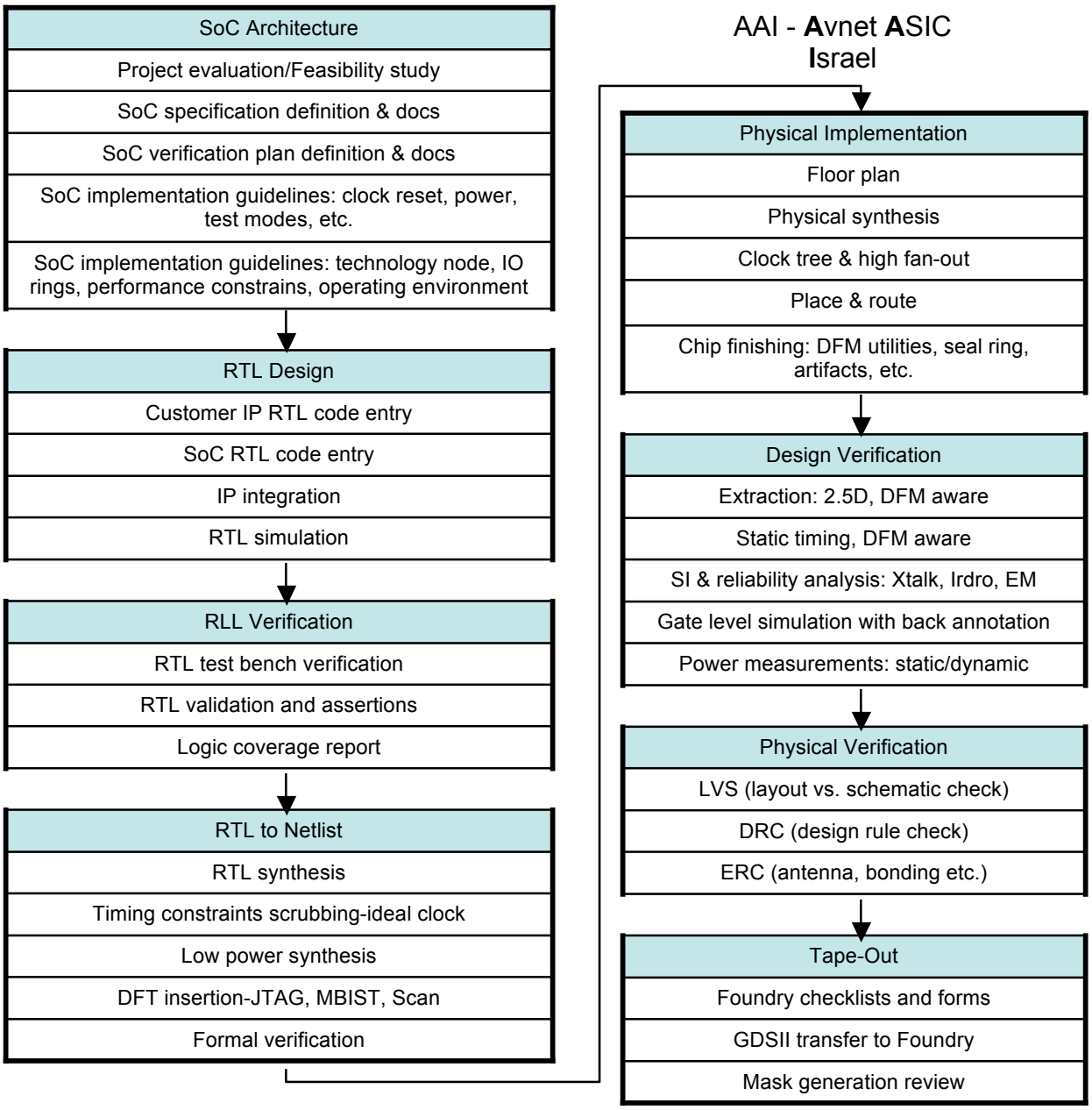


Figure 1: Avnet ASIC Israel SoC CAD Flow

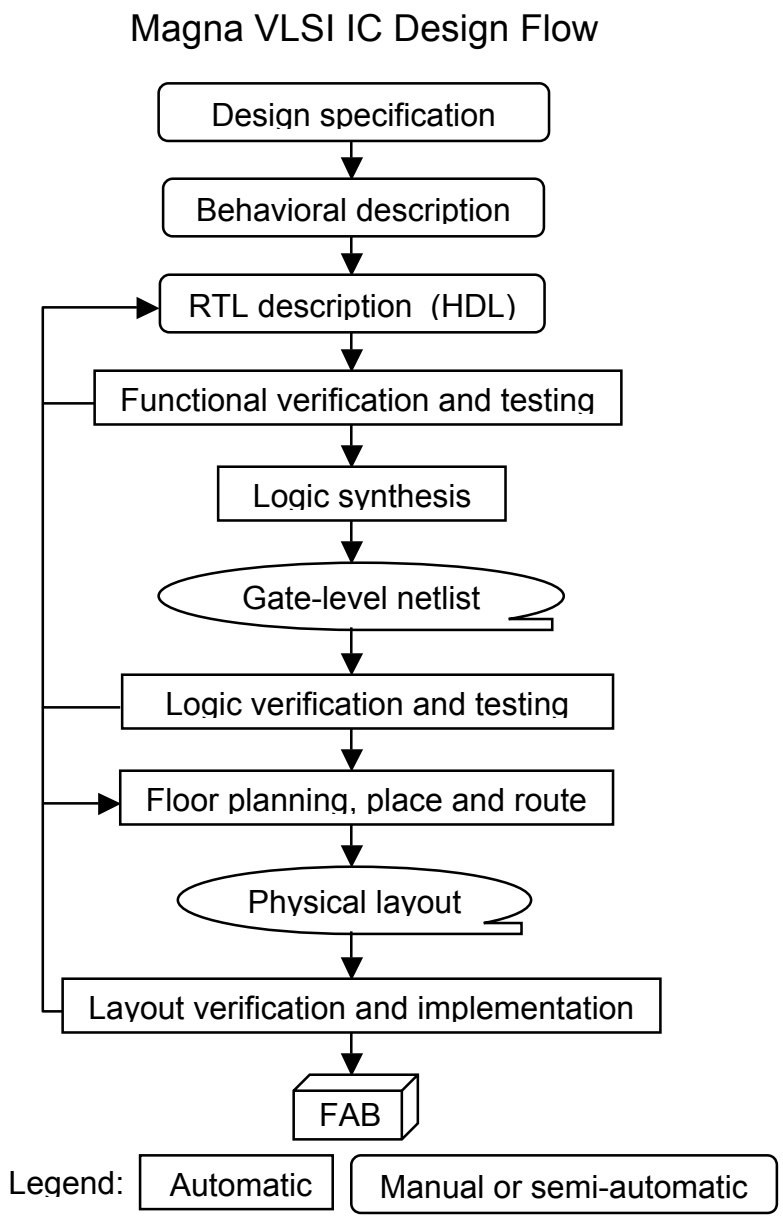


Figure 2: Magna VLSI IC Design Flow

3. Foundation for silicon system construction

Figure 3 indicates the main constituents needed to build VLSI systems. At the heart of the system is the **manufacturing/fabrication technology** itself. CMOS circuitry has been the driving force in this field over the past 40 years, and will continue to be for at least the next 15 years. Of course, radically different technologies are being developed, such as spintronics, nano-, bio-, chemical- and quantum-gates. The cost of constructing a new wafer fabrication facility is over $1 billion. Our courses EE 504L deal with aspects of the fabrication process.

Another pillar needed to support the development of VLSI systems is **verification**. There are several reasons why this is such an important component. First is that VLSI systems are very complex. They may consist of millions of lines of computer code and billions of gates. Thus they require the interaction of hundreds of designers. Secondly, many aspects of the design process deal with electrical models, and these are often approximations to reality. For example, while Spice simulations are fairly accurate for small circuits, because of run times, large circuits cannot be “Spiced.” So, if you build a complex system that relies on using one approximate model followed by another one, etc., there is a good chance the final product does not operate exactly as one intended. Verification is primarily dealt with we our CAD III class.

Design is the process discussed previously in our design flow, and is the main topic of EE477 and 577ab. The focus here is primarily on the following topics: (477) CMOS devices, logic gates, noise margins and delay calculations; (577a) advanced cell design and sizing-logical effort, data path design, sequential logic cells, pipelining and time borrowing, memory design, dynamic logic families, power calculations and minimization, and advanced issues related to delay and noise; (577b) SRAMs, DRAMs, ROMs, Flash, clocking and timing, Verilog, radiation effects, error correcting codes, processor architecture, high-speed signaling

As discussed previously, a large fraction of newly manufactured chips are defective, hence each must be **tested** to determine whether of not it is a good of bad chip. Since the development of such a test for each new product is very difficult, chip designers add extra logic to their designs so as to enhance the testability of the final chip. Thus, not only is test an essential aspect of VLSI technology, but also it has become an integral part of design and a designer’s responsibility, rather than something addressed by test engineers after the design is completed. (The same is true for verification.) Topics covered in our test course EE 658 include design for test, built-in self-test, JTAG/boundary scan, test generation, fault diagnosis, yield, reliability, and fault simulation.

What must be obvious to the reader is that producing a billion transistor chip is not done manually. Thus **computer-aided design** is the underlying foundation that makes fabrication, verification, design and test feasible. Our courses in this area consists of CAD I, II and III.

CAD I is intended for all M.S. and Ph.D. students following the VLSI curricula. It is design to give a concise overview of most of the key subjects in CAD, including: verification; high-level and low level synthesis; partitioning; floorplanning; placement; routing.

CAD II is intended for all Ph.D. students in this area, and for advanced M.S. students who wish to enter the CAD development field and/or differentiate themselves for the "masses". The focus of this course is on the principle mathematical techniques employed in CAD systems such as: graph theory; combinatorics; algorithmic techniques such a linear and integer programming, dynamic programming, branch-and-bound, backtrack programming, and greedy algorithms; heuristic techniques such as simulated annealing and genetic algorithms; and representational techniques such as binary decision diagrams.

CAD III addresses many of the same subjects discussed in CAD I, but now goes much deeper into the development of the underlying procedures using the tools developed in CAD II. Again, this class in intended for all Ph.D. students in this area, and for advanced M.S. students who wish to enter the CAD development field.

Finally, what is required of all students entering this program is a solid knowledge of logic design. While I have found that all students have studied this subject many years ago, and think they remember most of this material, in practice I have rarely found a students who really understands this material at the level required for this program. For example, while most students can identify the prime implicants of a function drawn on a Karnaugh map, they cannot define and implicant, define a prime implicant, define a redundant prime implicant, or explain why almost all books on logic design discuss prime implicants. You will be ready to enter the CAD and test courses when you can solve the following type of problems without referring to a book.

Given a word description of a process to be compute by a finite state machine, carry out the following tasks:

1. *Develop a Mealy machine model for this process*
2. *Find a minimal state equivalent Mealy machine*
3. *Encode the states*
4. *Determine the characteristic function for each state variable, and express it as a minimal SoP and PoS expression.*
5. *Determine the excitation functions for each state variable, assuming the machine will be implemented using JK-flip-flops. (I know, only D-flip-flops are used today.)*
6. *Develop a near-minimal logic (gate) circuit implementation for this problem.*

Since CAD implies solving design problems using software programs, there will be some program development in the CAD classes. Thus a solid background in C and/of C++ is needed. Eventually, before graduating, you must become familiar with the Unix operating system, data structures, and scripting languages like Perl and Tcl.

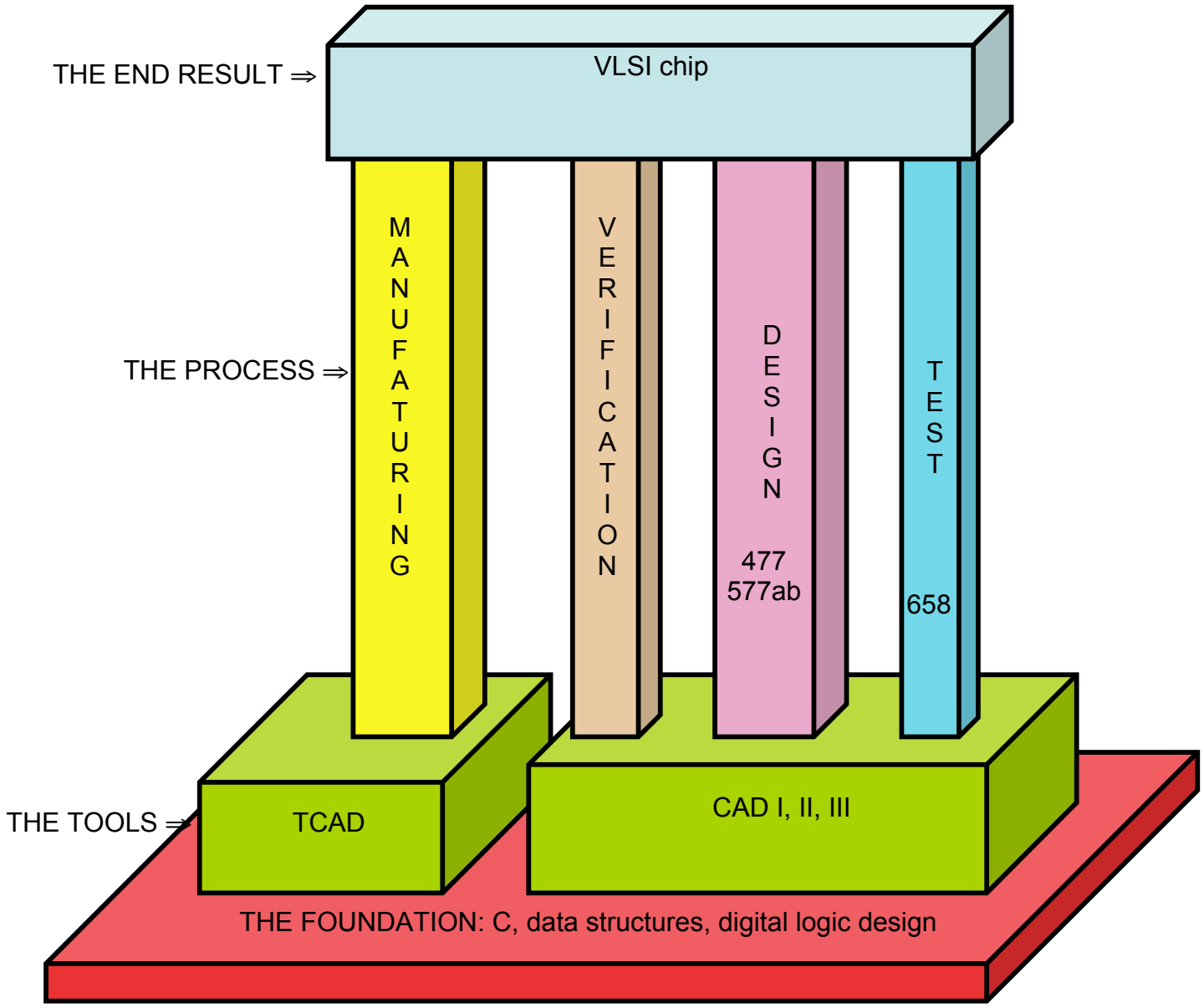


Figure 3: VLSI System Design: Foundation, Tools and Processes

4. Description of core and recommended courses for students focused on VLSI

Figure 4 indicates suggested programs for the first two years of study for students in the VLSI/CAD/Test area.

Fall	Spring
EE CAD I (4)	EE 577a (3)
EE 477 (4)	CSCI 455 (3)
Remedial class and/or English	EE 552 Asynchronous circuits (3)

Fall	Spring
EE 577b (2)	Analog circuits and/or fabrication (3)
EE 658 Testing (3)	Computer architecture and/or networks
Application: Communication, signal processing, biomed	

(a)

Fall	Spring
EE CAD I (4)	EE 577a (3)
EE 477 (4)	CSCI 455 (3)
Remedial class and/or English	EE CAD II (3)

Fall	Spring
EE 577b (2)	Analog circuits, fabrication, computer architecture, networks or asynchronous circuits (3)
EE 658 Testing (3)	Application: Communication, signal processing, biomed (3)
EE CAD III (3)	

(b)

Figure 4: (a) Schedule for students only taking CAD I; (b) Schedule for those taking CAD I, II, III.

Core courses

- **EE 477L MOS VLSI Circuit Design (4, FaSp)**
  Analysis and design of digital MOS VLSI circuits including area, delay and power minimization. Laboratory assignments including design, layout, extraction, simulation and automatic synthesis. *Prerequisite:* EE 328Lx or EE 338.

- **EE 577ab VLSI System Design (a: 3, FaSp; b: 3, FaSp)**
  a: Integrated circuit fabrication; circuit simulation; basic device physics; simple device layout; structured chip design; timing; project chip; MOS logic; system design silicon compilers. *Prerequisite:* EE 477 or passing a placement exam; b: VLSI design project; chip level design issues: power and clock distribution, packaging, I/O; design techniques; testability; chip fabrication and test.
- **EE 658 Diagnosis and Design of Reliable Digital Systems (3, Fa)**
  Fault models; test generation; fault simulation; self-checking and self-testing circuits; design for testability; fault tolerant design techniques; case studies. Prerequisite: graduate standing.
- **EE CAD I Overview of Computer-Aided Design for Digital VLSI Systems (4, Fa) Formerly EE680, labeled as EE599 for Fall 2010.**
  Design flow and need, attributes techniques associated with CAD related problems, including verification, high- (behavioral and register) and low-level (gate) synthesis, partitioning, floor planning, placement and routing. *Corequisite:* EE 477.
- **EE CAD II (EE581) Techniques and Models Used in Computer-Aided Design Tools for VLSI Circuits (3, Sp)**
  Mathematical techniques employed in computer-aided-design systems, including: graph theory, algorithmic and heuristic techniques for combinatorial problems, data structures and modeling. *Prerequisite*: EE CAD I.
- **EE CAD II Advanced topics in Computer-Aided Design for Digital VLSI Systems (3, Fa) Formerly EE 681.**
  An in-depth analysis of CAD techniques used for verification, high- and low-level synthesis, floor planning, placement and routing, including unique aspects related to 3-D, power and clock distribution, and noise.

Highly related digital systems courses

- **EE 552 Asynchronous VLSI Design (3, FaSp)**
  Asynchronous channels and architectures; implementation design styles; controller synthesis; hazards, and races; Petri-nets; performance analysis, and optimization; globally asynchronous locally synchronous design. Open only to graduate students. *Prerequisite:* EE 477.
- **EE 560 Digital System Design-Tools and Techniques (3, Sm)**
  ASIC design, FPGAs, VHDL, Verilog, test benches, simulation, synthesis, timing analysis, post-synthesis simulation, FIFOs, handshaking, memory interface, PCI bus protocol, CAD tools, design lab exercises. *Prerequisite*: EE 457, EE 454L; *Recommended preparation:* familiarity with CAD tools.

Highly related silicon related materials courses

- **EE 480 Introduction to Nanoscience and Nanotechnology (3, Fa)**
  Next-generation nanoscale materials and electronic devices: nanoscale fabrication and characterization, nanomaterials, nanoelectronics, and nanobiotechnology. *Prerequisite:* EE 338.
- **EE 504L Solid-State Processing and Integrated Circuits Laboratory (3, Fa, Sp)**
  Laboratory oriented with lectures keyed to practical procedures and processes. Solid-state fabrication and analysis fundamentals; basic device construction techniques. *Prerequisite:* BSEE.
- **EE 506 Semiconductor Physics (3)**
  Semiconductor bonds, crystallography, band structure assumptions, group theory, band structure results, k.p. method, quantum wells, wires and dots, superlattices, amorphous, organic semiconductors, defects, statistics, surfaces. *Prerequisite:* MASC 501.

Application areas

- **EE 519 Speech Recognition and Processing for Multimedia (3, Fa)**
  Speech production, acoustics, perception, synthesis, compression, recognition, transmission. Coding for speech, music, and CD-quality. Feature extraction. Echo cancellation. Audio, visual synchronization. Multimedia, internet use. *Prerequisite:* EE 483.
- **EE 483 Introduction to Digital Signal Processing (3, FaSp)**
  Fundamentals of digital signal processing covering: discrete time linear systems, quantization, sampling, Z-transforms, Fourier transforms, FFTs and filter design. *Prerequisite:* EE 301.
- **EE 567 Communication Systems (3, Fa)**
  Analysis of communication systems operating from very low to optical frequencies. Comparison of modulation and detection methods. System components description. Optimum design of communication systems. *Corequisite:* EE 464 or EE 465; *Recommended preparation:* EE 441.
- **BME 451 Fundamentals of Biomedical Microdevices (3, Fa)**
- **BME 523 Measurement and Processing of Biological Signals (3, Fa)**

5. Text used in classes and for preparation

- *Preparatory*: John Wakerly, **Digital Design: Principles and Practices**, Prentice Hall, Third or later edition. You must master almost all of the material in this text before entering our program.
- *Testing*: Miron Abramovici, Melvin Breuer and Arthur Friedman, **Digital Systems Testing and Testable Design**, IEEE Press
- *Testing*: Niraj Jha and Sandeep Gupta, **Testing Digital Systems**, Cambridge University Press.
- *CAD*: Laung-Terng Wang (ed.), **Electronic Design Automation: Synthesis, Verification, and Test,** Morgan Kaufman.
- *VLSI*: Neil Weste and David Harris, **CMOS VLSI Design: A Circuits and Systems Perspective**, Addison-Wesley.

- *VLSI:* Peter A. Beerel, Recep O. Ozdag and Marcos Ferreti, **A Designer's Guide to Asynchronous VLSI**, Cambridge University Press.

6. What are employers seeking?

Below I have edited two calls for employment I received some time ago to give you an idea of what employers are looking for in a new employee in addition to good **communication** and **software** skills. I have also reproduced an announcement about a new technology to illustrate the dynamic nature of technology.

- *Advanced Technology Solutions for Automotive & Semi Conductor Industries*
  An ever-growing range of cutting-edge technologies are emerging into production in the automotive electronic and semiconductor sectors. This makes these areas two of the most exciting and dynamic manufacturing sectors in the world.
  Today, Automotive & Semiconductor industries are facing a very difficult economic environment, together with heterogeneity increasing product quality, increased dependence on outsourcing, growing competition, and an endless list of business partners. The automotive electronics and semiconductor market sees enormous worldwide demand for its products, which brings tremendous opportunity for semiconductor companies. This increasing pressure to keep pace with the competition results in a reduction development time and the constant need for new and innovative products. However, not all products will do well in the marketplace as the consumer is extremely selective, and in particular on price and performance, leaving semiconductor companies deal with the triple challenges of increasing complexity, shorter lead times and growing competition. In fact, the need for enhancing quality, creating product differentiation while meeting safety and other regulations, is becoming more challenging
  This kind of situation has momentum for Advanced Technology Solutions, which provides solutions for the automotive and semiconductor industry to achieve the needed performance enhancements and improved efficiency. The partnership with our organization provides industry with the latest technological solutions in the areas of: Automotive Electronics, Engineering Design, Product Engineering, car infotainment and telematics, instrument clusters, industrial automation, drive, chip design, embedded software, Analog Mixed Signal Design, Electronic Design, Physical design, semiconductor design outsourcing, ASIC/SoC development, VLSI design, FPGA design solutions and support services such as Verification & Validation, offshore testing and automotive embedded software, a viable option as a lever on the technology know-how can the automotive industrial & semiconductor solutions. We also help in accelerating the development cycle and push the envelope of innovation.
  The use of this Advanced Technology Solutions, the automotive and semiconductor industries can work with a variety of business partners and suppliers in tandem to design and produce vehicles and focus on providing innovative working silicon chip design and embedded software support.

- *CN6194AL Bright Algorithm Development Engineer*
  NEW positions with a commercial company based in Hertfordshire working on the design and license of silicon and software IP for today's multimedia and communication devices.

  You'll have the opportunity to work on the next generation of advanced graphics technology, and will become part of a dynamic, motivated and very successful design team and organization.

  We are looking to recruit an academically bright graduate engineer or mathematician with either a good degree, MSc or PhD and experience or a definite interest in generating innovative implementations of mathematical functions.

  Primarily we are looking for you to be responsible for:

  - research into cutting edge techniques for adders/multipliers & algebraic mathematical functions
  - coding of a tool to generate RTL/VHDL
  - block level specification
  - verification/synthesis

  Ideally you'll have a good degree and a strong mathematical background, 1-3 years in a related field either at university or in industry and software in C++. Additional experience of algorithm development, implementation of adders/multipliers, VHDL/Verilog/RTL or ASIC design tools would be great.

  In return the company is offering you the chance to be a part of their continuing success and join a small team who are influencing the progress of the whole organization.

- Finally, below I give you an example of a forthcoming technology, which demonstrates why this industry is so dynamic and why we are continually developing new tools for synthesis and analysis.

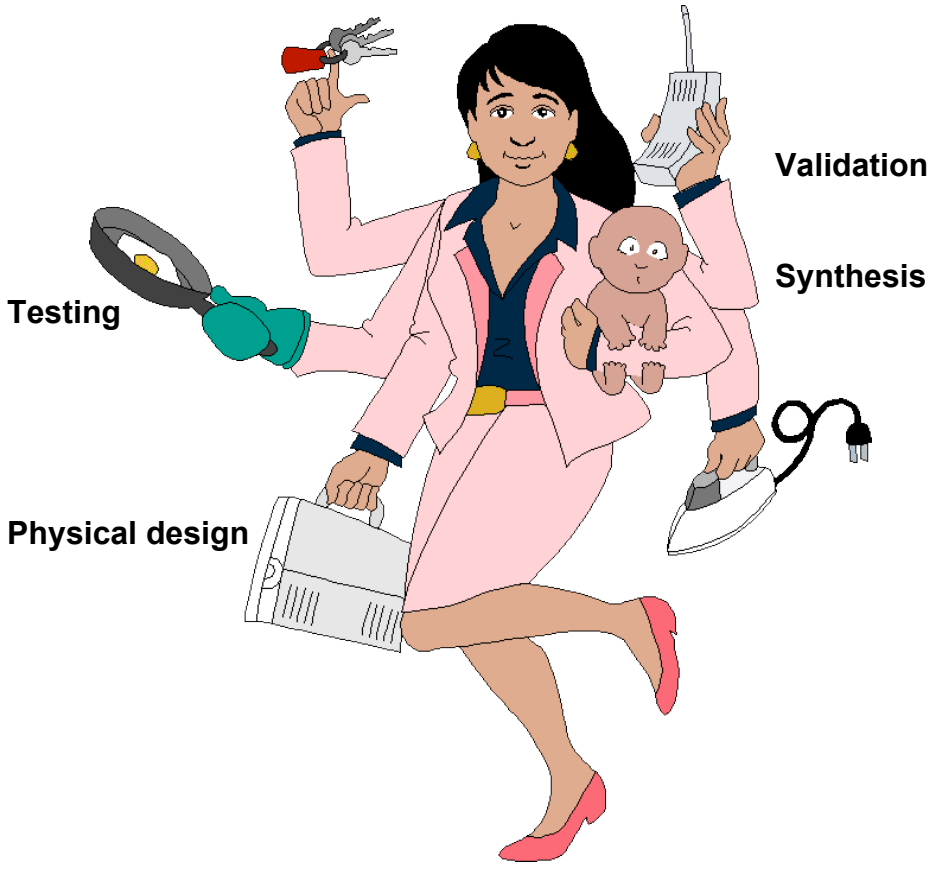
  EE Times
  06/18/2010
  SAN JOSE, Calif. -- Grandis Inc. has been awarded $8.6 million from the Defense Advanced Research Projects Agency (Darpa) for the second phase of a research project developing spin-transfer torque random access memory (STT-RAM) chips.

STT-RAM is a next-generation, MRAM technology. The first phase, supported by an award amount of $6.0 million, began in 2008 and was scheduled to last two years.

The program is being carried out by collaboration between Grandis, the Universities of Virginia and Alabama, and the College of William and Mary. The National Institute of Standards and Technology and the Naval Research Laboratory have provided additional support.

The contract has covered STT-RAM materials and processes, as well as STT-RAM architecture and circuit blocks. During Phase II, work will include test and verification of STT-RAM integrated memory arrays.

“STT-RAM has huge potential as the only non-volatile Random Access memory which scales beyond 10-nm. It is also the only technology fast enough to replace the existing DRAMs. It can replace embedded SRAM and flash at 45-nm, DRAM at 32-nm, and ultimately can replace NAND,’’ stated Farhad Tabrizi, CEO and president of Grandis (Milpitas, Calif.), in a statement.



A Designer must be able to do it all