What are the "Fundamentals" of Modern Digital Logic Design?

The evolving content of Trinity's one-semester course
in Digital Logic Design

Kevin M. Nickels

Department of Engineering Science
Trinity University

Abstract

As the practice of digital logic migrates away from discrete design and toward the construction of Systems on a Chip and the use of Intellectual Property Cores with programmable or highly customizable logic, how should the study of Digital Logic Design (DLD) evolve? Which of the "fundamentals" are still crucial to the understanding of DLD, and which are outmoded artifacts of the way **we used to do things**? What do we give up as we move from discrete design of digital logic to a architectural approach to digital systems?

This paper describes the evolution of an elective course in DLD in an Engineering Science program. The course is currently split roughly into thirds: combinatorial design, sequential design, and computer architecture. Except for the very first lab experiment, which focuses on instrumentation, all experiments are currently done in the Integrated Design Environment (IDE) provided by the vendor of the Field Programmable Gate Arrays (FPGAs) that comprise the main programmable hardware used for the course. Design entry using graphical components is done in the beginning of the course, with the majority of the projects utilizing VHDL (Very High Speed Hardware Hardware Description Language). Projects relate directly to the course material, and include a VGA (Video Graphics Array - video game, a 10-instruction simple computer, and a line-following robot. The course ends with case studies of contemporary digital designs. In this paper, the topics added and dropped are described, along with potential and actual repercussions on student learning.

Background and Context of Course

Trinity University is a small private liberal arts and sciences University in San Antonio Texas. The Engineering Science Department at Trinity University is a unique one. We offer a broad-based curriculum with a grounding in the "fundamentals" of electrical, mechanical, and chemical engineering, along with some specialization through disciplinary electives. Students earn a B.S. in Engineering Science, and customize their program with help from their academic advisor. More detailed information on the program is given in a paper by Uddin[1].

The "fundamentals" courses, required of all engineering majors, include (in the electronics portion) Electric Circuits, Electric Circuits Laboratory, Electronics I, and Electronics I

Laboratory. In these courses, the students are prepared for the electrical engineering portion of any disciplinary electives they desire. These electives are taken in the junior or senior year, and are typically offerred every other year. In this environment, one course in Digital Logic Design (DLD) is offered, along with a supporting laboratory course. This contrasts to a traditional electrical or computer engineering program, where this same topical coverage would be handled in more depth in two or even three semesters.

Electrical engineering electives are shown below in Table 1. Other than Applied Signals & Systems, no elective course requires other electives as prerequisites. Therefore, the DLD course must assume very little background knowledge beyond RLC circuits and diode and transistor design. Not all students will continue on to Embedded Microcomputer Systems in the spring. Some students will have had mechatronics in their junior year, and some will be in their junior year while in DLD (and may or may not have mechatronics in their senior year). As with other courses in other institutions[2], this diversity in background challenges course design.

Table 1: Electrical Engineering Electives at Trinity

Semester	***Fall***	***Spring***
Even Year	4365/4165 – Digital Logic Design	4369 – Embedded Microcomputer Systems
		4377/4177 – Electronics II (VLSI)
Odd Year	3321/3121 – Signals and Systems	4368 – Applied Signals & Systems
		4390 – Mechatronics

DLD Course Objectives

The DLD course mirrors the program philosophy, and therefore should cover the "fundamentals" of DLD, while keeping current with engineering practice. Objectives for the course reflect this mixture of goals. They include prerequisite knowledge and skills assumed by graduate schools in electrical engineering, as many of our students continue to these. Also, they include an understanding or appreciation of digital systems design in industry, as many of our students will at some point work on these systems. Finally, our students should have a broad appreciation of the impact and scope of digital systems, since this course will be the only exposure that some of our students have to these systems.

DLD Course Content

There have been two main motivators to the evolving design of the course structure. These are to decrease the overall workload seen by the students and the introduction and increasing emphasis on programmable logic in general and field programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs) in particular[3]. Table 2 shows a summary of how the content has changed over the past six years.

The first time this course was offered in its current form, Fall 1998, a popular text by Katz[4] was utilized to provide the main structure. This course covered a fairly traditional sampling from combinatorial and sequential design, with 7400-series and other SSI (small-scale integrated) chips as the design targets. Some attention to programmable logic in the form of PLAs and EEPROMs (Programmable Logic Arrays and Electronically Erasable Programmable Read-Only Memories) was paid. Due to the workload involved with covering combinatorial and sequential design, the computer architecture coverage in this course was rushed. This corresponds fairly well with Katz's advise in the text: combinatorial and sequential logic comprise a good one-semester course.

The 1998 laboratory followed along well until the computer architecture section, where a complex (12 SSI/MSI components) "simple computer" was discontinued in favor of an extension to a basic finite state machine (FSM) project. This was first implemented using SSI components, then re-implemented in a CPLD using the integrated design environment. No computer projects were done during the Fall 1998 semester in the laboratory course.

After the Fall 1998 semester, I was unhappy with several aspects of the course. First, since this was the only exposure many students would have to digital systems, I felt that a more involved final system should be studied, to more closely convey practical systems. Second, the methods studied for FSM design were not utilized in the final complex project, which hurt student motivation to master these methods. Third, there was not sufficient time in the computer architecture section for the students to appreciate the computer as a complex finite state machine, as Katz characterizes it.

In Fall 2000, the next time the course was offered, a different text by Lee[5] was used. This text compressed the sections on combinatorial and sequential design, leading me to supplement these sections. It also dropped some of the classical techniques, such as Boolean algebra, simplification via Quine-McCluskey, and FSM state assignment and minimization. In the time freed up by this, more emphasis was placed on programmable logic, including a section on VHDL (Very-high-speed-integrated-circuit Hardware Description Language), and a large module on the analysis of the Intel 8080 microprocessor as a combination of a large complex FSM and a basis for computer architecture.

The 2000 lab reflected this increased emphasis on CPLDs, only implementing a couple of projects with SSI components, then moving to the IDE to implement several projects using graphical schematic capture, and ending by implementation of the last several projects in VHDL[3]. One practical note is that the text utilized the Xilinx toolchain and FPGAs, while we use the Altera toolchain and boards at Trinity. This led to some minor but annoying misunderstandings in implementing some projects. This set of students, a very strong class, achieved the objectives of the course quite well.

The Fall 2002 course had minor changes from Fall 2000, the most notable in the laboratory, where the IDE now supported on-chip memory (on of the side-effects of offering the course every other fall is that the toolchain can change considerably between course offerings), enabling students to build the simplified 8080 clone without wiring to external memory chips. However, this class of students had a much harder problem understanding the 8080 structure and implementation than the class of 2000, and both I and they felt that the workload was still too high for a single course.

Table 2: Content Summary. Approximate number of hours devoted to various topics in DLD.

Year	**F98**	**F00**	**F02**	**F04**
Text	Katz	Lee	Lee	Hamblen/ Furman
Combinatorial Design				
Number Systems	1	1.5	2	1
Gates	1	0.5	0	0
Boolean Algebra	4	0.5	1	0
Simplification (Kmaps/QM/Espresso/NAND-NAND)	4	2	3	0.5
Timing/Hardware Issues (Delay/Hazards/Tri-State/OC)	2	0	0.5	0.5
Arithmetic Circuits (Add/Sub/Mult)	4	2	2	5
Programmable Logic	1	2	1	4
VHDL	0	3	3	5
Sequential Design				
Latches/Flip-Flops/Timing Issues	4	0.5	2	0
FF Types; Registers; Memory	3	1	1	2
Counter Design	3	0	0	0
FSM Design	6	6	7	6
Computer Architecture				
Architecture/Structure overview	1	3	4	4
Machine Cycle/RTN	1	3	1	2
Controller/Data Path; Busses	3	4	5	3
Controller Design	3	2	3	2
Labs				
Combinatorial Design				
w/SSI Chips	5	2	2	1
w/FPGA	0	2	2	3
Counter/FSM Design				
w/SSI Chips	1	0	0	0
w/FPGA	2	1	1	4
Simple Computer	0	7	7	3
Line-Following Robot	0	0	0	2

Therefore, the Fall 2004 course brought yet another text and more changes in content. The text by Hamblen and Furman[6] utilized the same IDE and CPLD that we were using in the lab, and provided an IP-Core (intellectual property core) toolbox to automate some of the more mundane details of the laboratory projects, such as a mouse interface, debouncing of pushbuttons, and a

VGA signal generator. The classical combinatorial design techniques such as Karnaugh maps and two-level simplification are not covered at all, though I supplemented the text with some of them. Likewise, the classical FSM design techniques are not covered in depth. The final project in this course is a simple "toy" computer called the µP1 that has a single addressing mode, ten instructions, and an adder for an ALU. The course concluded with a survey of several IP-Cores on the market, and a better appreciation for the art of modern digital systems design.

The Fall 2004 laboratory course was reworked to parallel the tutorial style of the text, utilizing the IDE/CPLD for all projects after the first few. Projects included a simple video game (after examples showing a mouse interface utilizing the Mouse IP-Core and a bouncing ball utilizing the VGA IP-Core), the simple computer from lecture, and finally a robotic line-follower.

To summarize the evolution in content from 1998 to 2004, some of the combinatorial and sequential design techniques classically viewed as "fundamental" such as K-maps, Quine-McCluskey, FSM State Assignment and Reduction, and mapping designs into SSI/MSI chips have been dropped, to make way for more emphasis on HDLs, the use of IP-Cores and CPLDs, and the design and study of Systems on a Chip (SoC).

Discussion and Conclusions

The DLD course in the Engineering Science Department at Trinity University has a unique place in the curriculum. It ideally orients students to digital systems, teaching them enough of the "fundamental" skills that the workings of the modern design tools are not "black magic", but leaving enough time and space to achieve mastery of these tools. Students should understand the state of the art in DLD, which implies heavy use of IP-Cores, SoC design, and HDLs. Students should appreciate the scope of the impact of digital systems on our society, and see where electrical engineers can contribute to this impact in a positive way.

These aims are sometimes mutually exclusive, and certainly add up to more than a single semester course. In addition, the move from low-level logic design to systems-level architectural design changes the feel of the course, possibly impacting students who lean toward the "sensor" style of learning[7]. Certainly, proficiency with the skills traditionally associated with **digital logic** design, such as logic simplification, Boolean algebra, and FSM optimization is decreased somewhat, though not to the extent described by Stephan and Sriraman[8]. The lab projects, in particular, illustrate (to the author at least) that students are learning enough of the fundamentals to execute fairly involved digital designs, such as a simple computer (including ALU), video game, and line-following robot. Many instructors feel that in this type of course students have a better feeling for more complex **digital systems** with courses similar to the current structure[9,10].

Through discussion with peers at other institutions (including, of course, following ASEE conferences[2,8-12]), interactions with industry practitioners, discussion with students, and of course formal student evaluations, I have refined the content of Trinity's DLD course continuously over the past six years. I hope that as the practice of digital systems design evolves, so too do our courses.

References

1. M. Uddin, Multidisciplinary Engineering Science program at Trinity University. In *Proceedings American Society for Engineering Education Annual Conference*, 2004.
2. K. Stephan and V. Sriraman, A Digital Electronics Course Using CPLDs for Manufacturing Engineers. In *Proceedings American Society for Engineering Education Annual Conference*, 2004.
3. K. Nickels. Pros and Cons of replacing discrete logic with programmable logic in introductory digital design courses. In *Proceedings American Society for Engineering Education Annual Conference*, June 18-21, 2000, St. Louis, MO
4. R. Katz. Contemporary Logic Design. Benjamin/Cummings, California, 1994.
5. S. Lee. Design of Computers and Other Complex Digital Devices. Prentice/Hall, New Jersey, 2000.
6. J. Hamblen and M. Furman. Rapid Prototyping of Digital Systems – A Tutorial Approach. Second Edition. Kluwer, Boston, 2001.
7. R. Felder and L. Silverman. Learning and teaching styles in engineering. *Journal of Engineering Education* 77 (2), February 1988.
8. J. Greco, Designing a Computer to Play Nim: A Mini-Capstone Project in Digital Design I. In *Proceedings American Society for Engineering Education Annual Conference*, 2004.
9. K. Kramer and D. Maxwell, Projects with applications to Wireless Communications – An Innovative Approach to the Digital Design Course. In *Proceedings American Society for Engineering Education Annual Conference*, 2004.
10. W. A. Chren and B. G. Zomberg. Programmable logic course development in an engineering curriculum. In *Proceedings American Society for Engineering Education Annual Conference*, pages 1154 1158, 1993.
11. R. Coowar. Designing with field programmable gate arrays. In *Proceedings American Society for Engineering Education Annual Conference*, pages 853 859, 1995.
12. D.W. Horning. Integration of digital design tools into a digital design sequence. In *Proceedings American Society for Engineering Education Annual Conference*, pages 1104 1108, 1993.
13. D. Hall, Teaching Design Methodology and "Industrial Strength" EDA Tools in a First-Term Freshman Digital Logic Course. *IEEE Transactions on Education* 41(1), February 1998.

KEVIN M. NICKELS
Dr. Kevin M. Nickels is an associate professor in the Department of Engineering Science at Trinity University. He received the B.S. degree in Computer and Electrical Engineering from Purdue University (1993), and received the M.S. degree (1996) and the Ph. D. (1998) in Electrical Engineering from The University of Illinois at Urbana-Champaign. He is currently working in the areas of computer vision, pattern recognition, and robotics. Dr. Nickels has been a member of ASEE since 1998 and a member of IEEE since 1994.