2006-1672: TECHNOLOGY EDUCATION AND THE NEW FRONTIER OF DIGITAL ELECTRONICS

Richard Furtner, Purdue University

Neal Widmer, Purdue University

Technology Education and the New Frontier of Digital Electronics

N. Widmer
R. Furtner

Purdue University

Abstract

Throughout the ages, man has learned, discovered and built using the resources available at the time. Education is about providing the knowledge and skill set necessary for the next generation to continue and advance this process.

Engineering technology education must introduce the student to the latest methods that are being used in industry. Yet basic underlying principles of any subject are also important, especially if they are necessary to support current and future practices. As we make technological progress, many subjects that were at one time foundational to the practices of the day may become less important to present and future employers.

Just as transistors replaced vacuum tubes, and calculators supplanted slide rules, another example of this phenomenon is occurring in the field of digital electronics today. Hardware description languages and large digital ICs are replacing discrete logic gates.

This paper proposes a process, using digital electronics as an example, which keeps a curriculum and its graduates current by prioritizing the skills which are most important to current employers. An attempt is made to sort out the vital digital electronics topics from the less relevant, and to propose the necessary topics for today's students.

Introduction

Digital electronics is an area in which rapid changes are occurring. Moore's law has caused the discrete-gate logic of the 70s and 80s to be superseded by multimillion-gate CPLDs, FPGAs, and ASICs today [1]. Design methodologies for these large chips began with schematic entry design techniques in the late 1970s and early 1980s. Schematic entry of digital circuits was largely supplanted by the use of high-level hardware description languages in the mid-1990s, especially VHDL and Verilog.

In the past, electronic systems had to compete in the market on the basis of cost, quality, and durability. In the modern arena we know that the system that is dominant today will be slower and cost more than the system that will be here tomorrow. Customers are not as likely to own a device for as long, due to the tendency to upgrade.

Today, many large CPLDs and FPGAs are designed using a system-on-chip methodology[2] [3], in which large, predesigned VHDL or Verilog blocks are acquired, and then "stitched" together and verified.

Because of these factors, the emphasis on digital designers has shifted to minimizing time to market, rather than fully optimizing all digital circuitry. The optimum circuit often means the smallest microcontroller, programmable logic device, or ASIC that is capable of implementing the design within the market window.

Because of this new emphasis placed on digital designers, many topics traditionally taught in early digital design classes may be less of a priority to current employers than some of the concepts traditionally taught in optional, upper-level elective courses.

For instance, the need to describe a logic circuit using the minimum SOP expression, and minimize the number of SSI or MSI ICs has become less important to current employers. Therefore, concept of simplification should be taught, but the traditional methods may be less necessary. This is because development software packages use sophisticated algorithms to maximize the use of resources in the target device.

Therefore, the hours spent teaching Boolean algebra theorems, reducing expressions using KMAPs, and transforming expressions to use all NAND gates may be better spent on other topics that impact the practical application of digital systems today.

Devices that were developed to achieve these optimization goals of the past are also now becoming less relevant to employers. For example, the JK flip flop was developed because it allowed for more efficient excitation circuitry than using D flip-flops to implement state machines and counters. In modern FPGAs, CPLDs, and ASIC libraries, JK flip flops are often not used.

Consequently, the hours spent on JK operation and classic JK state machine design could be spent on present state-next state concepts. The systems made up of TTL SSI and MSI ICs are typically legacy systems. The problem is no longer one of choosing the correct building block (IC) with the right combination of bit width, synch/asynch inputs, trigger edge, etc, but rather describing the operation of a module that will meet the needs of the system.

It may be necessary to attempt to analyze the topics covered in typical digital design sequences, and attempt to gauge their *relative relevance* to current and future employers, given the current design methodologies and schedule constraints.

Universal Fundamentals

Whenever advancing technology requires educators to cover a new topic, the alarm is sounded that the fundamentals must not be overlooked. This is true, but it is also a fact that the fundamentals must be reviewed to see if they are still foundational to the current technology, and whether they are essential to understanding systems that are founded upon them. This section attempts to distinguish between the topics that are less relevant to the future practices of digital electronics. Table 1 presents a list of fundamentals that are vital to understanding digital systems, along with the ways these topics manifest themselves in modern systems as well as the past.

Universal Fundamentals	Relevant Skills to Today's Designers	Less-used Skills
Number systems	Binary, Hex, Decimal	Octal
Codes(BCD, ASCII, etc	Code conversion	Math with codes
Boolean Equations	New operators (&, #, !)	Ambiguous operators (•, +)
		Algebraic simplification
		Karnaugh-Map Simplification
Demorgan's Theorem	Alternate symbols	SOP to NAND:NAND conversion
Minimizing/optimizing	Fitting to proper part	Minimizing chip count
Logic Diagrams	Block symbols, generic Symbols	Universal NAND gates, TTL ICs and TTL data sheets, IEEE symbols.
Signal Flow	Signal assignment (connection)	Point to point wiring
Active HIGH/LOW Inputs and outputs	Good labeling	Reading TTL data sheets, block symbols
Timing diagrams	Reading	Generating
Level triggered Latches	Software feedback issues	Hardware feedback issues
Edge triggered flip-flops	Registers	JK Flip flops
Present/Next state tables	CASE, IF/ELSE	Excitation tables
Synchronous/Asynch. Counters	Modular development	Asynchronous FF counters
Cascading	Count Enable/Terminal Count decoding	Creating large counters out of "MSI" counters (74160, 161)
Synchronous/Asynch. Control inputs	Clock events	
Setup/Hold time Maximum Freq Propagation delay	Clock skew and path length Static Timing Analysis	Analysis of discrete TTL circuits
IC characteristics	CMOS, low/mixed voltage	TTL, 5 volts only
State machines	Enumerated type variables (named states)	JK flip flop design techniques Design with discrete logic State Minimization Skills

Table 1 Topics in Digital Systems Sequences

Another way of viewing relevance might be to survey potential employers of new graduates, and perhaps current digital designers. The survey would consist of showing these individuals currently-taught skills in the school's digital design sequence. By asking them to rank their preferred skills for today's graduates to have, in terms of high, medium, and low, an assessment can be made as to the relevance of the topics currently taught to today's employers and designers. (Since employers often want all graduates to have all skills, a requirement may have

to be made that for every highly desired skill, there must be at least one corresponding low-ranked skill)

An attempt to perform an initial 'relevance ranking' of currently taught digital design topics, based solely on one author's 20+ years of digital, ASIC, and embedded systems design experience, is shown in Table 2 below. This list is not meant to be all-encompassing, but merely representative of typically introduced digital topics.

Topic	Individual Skill	Where typically introduced	Relative Skill Relavance
Discrete Logic	Boolean Equations with 'mathematical' representation of and/or (. And +)	Dig. Fundamentals I/II	Low
	Boolean Equations with 'C' representation of and/or (& |)	Upper-Level Electives	Medium
	Truth Tables	Dig. Fundamentals I/II	High
	Simplification using Boolean Laws	Dig. Fundamentals I/II	Low
	K-map simplification	Dig. Fundamentals I/II	Low
	Quine Mclusky, Prather simplification	Upper-Level Electives	Low
	Demorgan's gate equivalents	Dig. Fundamentals I/II	Medium
	Implementing truth tables with VHDL/Verilog	Upper-Level Electives	High
	Simulating truth tables with VHDL/Verilog	Upper-Level Electives	High
"MSI" Logic (Muxes, Decoders, ALUs, Parity Detectors)	Implementing with discrete gates	Dig. Fundamentals I/II	Low
	Implementing with TTL ICs	Dig. Fundamentals I/II	Low
	Implementing in VHDL/Verilog	Upper-Level Electives	High
Flip-Flops/ Registers	Implementing Circuits with D Flip-Flops	Dig. Fundamentals I/II	Medium
	Analyzing Circuits with D Flip-Flops	Dig. Fundamentals I/II	Medium
	Implementing Circuits with JK Flip-Flops	Dig. Fundamentals I/II	Low
	Analyzing Circuits with JK Flip-Flops	Dig. Fundamentals I/II	Low
	D Latches	Dig. Fundamentals I/II	Medium
	N-bit Registers	Dig. Fundamentals I/II	Medium
	Setup/Hold time principles	Dig. Fundamentals I/II	High
	Setup/Hold time of TTL flip-flops	Dig. Fundamentals I/II	Low
	Static Timing Analysis	Upper-Level Electives	High
Counters	Binary (Principles)	Dig. Fundamentals I/II	High

Topic	Individual Skill	Where typically introduced	Relative Skill Relavance
	BCD/Decade (Principles)	Dig. Fundamentals I/II	High
	Designing binary/Decade with discrete D flip-flops	Dig. Fundamentals I/II	Medium
	Designing binary/Decade with discrete JK flip-flops	Dig. Fundamentals I/II	Low
	Creating mod-N counter with a discrete mod-M (M>N) counter	Dig. Fundamentals I/II	Low
	Cascading Counters	Dig. Fundamentals I/II	High
	Discrete (D FF) Ripple Counters	Dig. Fundamentals I/II	Low
	Grey-Code & One-hot	Dig. Fundamentals I/II	Medium
	Random sequence, LFSR	Upper-Level Electives	Medium
State Machines	State Tables	Dig. Fundamentals I/II	High
	State Transition Diagrams	Dig. Fundamentals I/II	High
	Mealy-Moore implementations	Dig. Fundamentals I/II	Medium
	Designing with discrete logic (D & JK flops)	Dig. Fundamentals I/II	Low
	Designing with VHDL/Verilog	Upper-Level Electives	High
	Alternative State Encodings (One-hot, grey)	Dig. Fundamentals I/II	Medium
	Redundant state minimization	Upper-Level Electives	Low
	Designing in VHDL/Verilog	Upper-Level Electives	High
Memories	Static Ram/ROM/EEPROM	Dig. Fundamentals I/II	Medium
	Dynamic RAM	Dig. Fundamentals I/II	Medium
	DDR	Upper-Level Electives	High

Table 2 – Typically Taught Digital Topics, and their Relative Relevance to Today's Employers

Table 2 highlights the discrepancies between the relative usefulness of skills taught in many digital fundamentals courses. Many skills which are not as heavily utilized by today's digital designers, and are not a high priority with current employers, are still emphasized in courses on digital fundamentals. (These skills are highlighted in grey in the table). Also, many skills needed by today's digital design and test engineers are only covered in upper-level elective courses, which might not be taken by all students who will encounter digital logic.

This 'skills inversion' was also shown in an unpublished survey of users of digital design textbooks by Pearson's Press. ***Less than 40% of responding EET departments currently cover high-relevance skills like CPLD programming or VHDL (or plan to in the future).*** The complete results of this survey are shown in Table 3 below:

		All	2yr	4yr	ET	EET	CET
Are you covering CPLD programming in	CPLD programming is covered now	23%	19%	29%	15%	**24%**	44%

		All	2yr	4yr	ET	EET	CET
your course?	It will be covered 2 years from now	25%	28%	19%	27%	**37%**	17%
	I do not anticipate covering it	52%	52%	52%	58%	**39%**	39%
	Total Responses	158	109	48	33	**51**	18
Are you covering VHDL programming in your course?	VHDL programming is covered now	15%	14%	17%	9%	**15%**	17%
	It will be covered 2 years from now	31%	30%	31%	33%	**40%**	22%
	I do not anticipate covering it	54%	56%	52%	58%	**44%**	61%
	Total Responses	156	107	48	33	**52**	18

Table 3 – Results of a Publisher's Survey of Users of Digital Electronics Textbooks

Purdue ECET Department Digital Design Sequence

The purpose of this paper is not to pass judgment on the merits of any particular topic or sequence of topics in digital circuits. Rather, the goal is to demonstrate that a 'skills inversion' is likely present in the teaching of digital circuits, and that a strategy to address some of this phenomenon may need to be implemented.

For example, the ECET department at Purdue University used to teach assembly-language programming in their *first* microprocessor course. Once industry needs dictated that knowledge of the C-language was a more desired skill, the first microprocessor course was modified to emphasize C programming. Only subsequent microprocessor courses introduced assembly-language programming.

At Purdue, the freshmen start out in a first semester course that teaches number systems and logic functions in the first two weeks. Lab experiments through week 4 still use TTL ICs and breadboards as an easy way to quickly implement and test the basic concepts of digital logic without needing to learn the computer tools required for advanced development.

By week 5 the students are learning graphic entry techniques using Altera software and by week 6 they are describing circuits using AHDL. The benefit of using AHDL is that it is much more "beginner friendly" than VHDL, while demonstrating most of the important concepts of describing logic circuits using text. The tedious details of VHDL are much more easily conquered in more advanced classes after mastering the basics in AHDL. The middle segment of the first course teaches the basics of counters, sequence modification, cascading, timing diagrams, and frequency division. This is currently being done with TTL IC counters in lab.

Counters described using AHDL are covered in the second semester course. The last segment of the course covers the operation and AHDL description of decoders, encoders, multiplexers, and demultiplexers. The final two weeks are spent completing a team project that uses the many

building blocks covered in the course to implement a system. The Altera tools allow very substantial, interesting projects to be developed by first semester students.

The second required digital course builds on all areas of the first course with an emphasis on counters using AHDL. It also expands into other areas such as multivibrators, A/D and D/A conversion and memory systems.

Proposed Flow

The plans at Purdue are to move the introductory courses even further toward the future needs of industry. The next generation of CPLD hardware platforms will require the next generation of software tools, and that means increased complexity. The course must be adjusted to begin teaching the tools sooner.

In order to create time to accomplish this, the topics of Boolean simplification, NAND implementation of combinational circuits, K-Maps and JK flip-flops will be phased out. The laboratory segment will move away from the standard breadboard and DIP ICs, using the CPLD to demonstrate basic logic circuits and counters. Hierarchical graphic block diagram techniques will be taught to build more complex systems from the basic circuits described using AHDL.

The cost of CPLD hardware platforms is coming down. There are now basic development boards on the market for around $50 and the industrial quality software, complete with simulation tools, is available at no cost to the students. At this level of capital outlay, the students can be required to buy the equipment they need for lab at a price comparable to buying a parts kit of TTL ICs, which will no longer be necessary. They will have the advantage of owning their CPLD and be able to use it for upper division design courses and senior projects as well.

Bibliography

1. Green, Mike, "Standing in the shadows of giants", ElectronicsWeekly.com, July 18th, 2005. Downloaded on January 17th, 2005 from the ElectronicsWeekly website at: http://www.electronicsweekly.co.uk/Articles/Article.aspx?liArticleID=35857
2. A.P. Niranjan, Paul Wiscombe. "Islands of Synchronicity, a Design Methodology for SoC Design," Design, Automation and Test in Europe Conference and Exhibition Designers' Forum (DATE'04), 2004, p. 30064
3. Atitallah, Kadionik, Ghozzi Nouel, and Masmoudi, "Real-Time Video System Design Based on the NIOS II Processor and uClinux", IP Based SoC Design Conference & Exhibition (IP-SOC 2005), 2005