AC 2012-4637: IDENTIFYING THE CORE CONCEPTUAL FRAMEWORK OF DIGITAL LOGIC

Dr. Geoffrey L. Herman, University of Illinois, Urbana-Champaign

Geoffrey L. Herman earned his Ph.D. in Electrical and Computer Engineering from the University of Illinois, Urbana-Champaign as a Mavis Future Faculty Fellow. He is currently a Postdoctoral Researcher for the Illinois Foundry for Engineering Education. His research interests include conceptual change and development in engineering students, promoting intrinsic motivation in the classroom, blended learning (integrating online teaching tools into the classroom), and intelligent tutoring systems. He is a recipient of the 2011 American Society for Engineering Education (ASEE) Educational Research and Methods Division Apprentice Faculty Grant. He has been recognized with the Olesen Award for Excellence in Undergraduate Teaching from the Department of Electrical and Computer Engineering and the Ernest A. Reid Fellowship for engineering education. He has served as a graduate affiliate for the Center for Teaching Excellence. He is currently the information chair for the ASEE Student Division and the immediate past chair of the Graduate Engineering Education Consortium for Students.

Prof. Michael C. Loui, University of Illinois, Urbana-Champaign

Michael C. Loui is professor of electrical and computer engineering and University Distinguished Teacher-Scholar at the University of Illinois, Urbana-Champaign. His interests include computational complexity theory, professional ethics, and the scholarship of teaching and learning. He serves as Executive Editor of College Teaching, and as a member of the editorial board of Accountability in Research. He is a Carnegie Scholar and an IEEE Fellow. Loui was Associate Dean of the Graduate College at Illinois from 1996 to 2000. He directed the theory of computing program at the National Science Foundation from 1990 to 1991. He earned the Ph.D. at the Massachusetts Institute of Technology in 1980.

Identifying the Core Conceptual Framework of Digital Logic

Abstract

As a relatively new field, computer engineering has yet to reach the maturity of more established disciplines such as physics and chemistry. Consequently, instructors in computer engineering still disagree about what is essential for students (both computer engineering students and others) to learn. In an effort to provide a basis for this discussion, we propose that engineering educators should identify a core conceptual framework for its introductory level courses. We suggest one such core conceptual framework that is built around the three central concepts and skills of state, fixed-length information encoding, and the ability to switch between levels and types of abstractions.

1 Introduction

With the development of concept inventories and other conceptual assessment tools, engineering educators have become increasingly aware of the importance of teaching students about concepts and conceptual frameworks rather than rote skills or lists of facts[1]. Students who possess a consistent core conceptual framework are better able to recall knowledge, apply knowledge, and learn new knowledge, because the framework helps students synthesize their knowledge into a manageable cognitive unit.

In the context of long-established disciplines such as physics and chemistry, instructors commonly agree upon a core conceptual framework for the discipline (e.g., Newton's three laws for mechanics; and molecules, reactions, and conservation of mass for chemistry). However, in younger, emerging disciplines such as computer engineering and computer science, there is no settled core conceptual framework. Furthermore, we are unaware of any widely accepted, rigorous method for creating such a framework; rather it seems that these conceptual frameworks evolve and emerge slowly over time. Consequently instructors disagree about what concepts, tools, and skills are essential for a student to learn in a first course in many computing topics such as digital logic and computer organization.

In this conference presentation, we hope to begin a dialogue to establish a core conceptual framework for digital logic courses that will move disciplinary thinking away from

individual topics to a more interconnected web of concepts that are built upon a few foundational concepts. We present evidence from a Delphi poll of experienced digital logic instructors and textbook authors to define the scope for a "typical" digital logic course and to propose the core concepts of digital logic. Next, we use evidence from a series of studies on students' misconceptions in digital logic to bolster our argument for the centrality of certain concepts. We conclude by suggesting how this conceptual framework can affect both research and instruction.

We believe that establishing an accepted core conceptual framework can empower instructors to make better informed decisions when choosing learning goals for their courses and about what content to keep in their courses. Establishing a core conceptual framework can also focus instruction and help students develop better conceptual knowledge in computer engineering.

2 Background

Recently Goldman et al. conducted a Delphi study to identify a set of core important and difficult concepts in three computing topics: programming fundamentals, discrete mathematics, and digital logic[2]. A panel of 20 digital logic experts proposed 44 concepts and skills which they considered to be central and core to a first course on digital logic and computer organization. Through a series of three ratings negotiations, the panel rated each of these concepts and skills on a scale of one to ten on difficulty (with 1 as least difficult and 10 as most difficult) and importance (with 1 as least important and 10 as most important). The list of important and difficult concepts has since been used to guide the development of concept inventories for these three computing topics[3].

In addition to these ratings, the Delphi process also revealed disagreements about the importance or centrality of different concepts and skills[2]. For example, experts disagreed about whether a digital logic course should be theoretically focused and ignore the physical constraints of real circuits (issues such as active-high versus active-low) or whether it should be practically focused and emphasize the use of design tools such as hardware description languages.

A quick perusal of digital logic textbooks also reveals the same disagreement about the centrality of various topics in digital logic[4-10]. For example, some textbooks teach digital logic

within the bounds of a chosen hardware description language, and others ignore hardware description languages entirely. Further review of these textbooks also reveals differing opinions about the definition of key terms such as *state* and *state transition*[11]. Remarkably, there are also no accepted IEEE standard definition for the terms in digital logic.

These disagreements are the mark of any young, but maturing discipline. We hope to find the common threads and core concepts of digital logic to better frame these ongoing discussions and provide a core conceptual framework.

3 A Fresh Analysis of the Digital Logic Delphi Study

In this section, we reanalyze the results of the digital logic Delphi study of Goldman et al. in an attempt to identify the core conceptual framework of digital logic[2]. Unlike the original intent of the Delphi study, we are not concerned about the difficulty of acquiring certain concepts or skills but only about the importance of each concept or skill.

To begin our new analysis, we sorted the list of topics from the Delphi study and looked for a convenient breaking point in the data (See Table 1). This process led us to identify two potential breakpoints: a rating of 9 or greater or a rating of 8 or greater. We decided to use the breakpoint of 8 or greater for our analysis, because this breakpoint included a set of topics that provided for a range of concepts and skills to be included but still sufficiently narrowed the list of important topics. This breakpoint also proved to be convenient as the first contentious skill (Using CAD tools) falls at the bottom of the list.

Table 1: Top 15 most important concepts and skills from the digital logic Delphi study[2]

Concept/Skill	Importance
1. **State transitions**: *Understanding the difference between the current state and the next state, and how the current state transits to the next state.*	9.8
2. **Converting verbal specifications to state diagrams/tables**	9.8
3. **Functionality of multiplexers, decoders and other MSI components**: *Excludes building larger MSI components from smaller MSI components.*	9.6
4. **Converting verbal specifications to boolean expressions**	9.5
5. **Hierarchical design**	9.5
6. **Relating timing diagrams to state machines, circuits**	9.4
7. **Understanding how a sequential circuit corresponds to a state diagram**: *Recognizing the equivalence of a sequential circuit and a state diagram.*	8.9

8. **Modular design**: *Building circuits as a compilation of smaller components.*	8.9
9. **Number representations**: *Understanding the relationship between representation (pattern) and meaning (value).*	8.6
10. **Analyzing sequential circuit behavior**	8.5
11. **Converting algorithms to register-transfer statements and datapaths**	8.5
12. **Designing control for datapaths**	8.5
13. **Debugging, troubleshooting, and designing simulations**: *Debugging skills with a focus on designing rigorous test inputs/simulations for circuits.*	8.5
14. **Binary arithmetic**: *Topics such as binary addition and subtraction, but not optimized circuits (e.g., carry-lookahead).*	8.4
15. **Using CAD tools**	8.4

We then looked for commonalities among the list of concepts and skills in an attempt to identify underlying concepts that undergird this list of essential topics. This search revealed three candidate core concepts and skills: (1) state, (2) fixed-length information encoding, and (3) ability to switch between levels and types of abstractions.

4 Building a Case for the Three Core Concepts and Skills

In this section we will attempt to build a case for the centrality of our three candidate concepts and skills.

4.1 State

Based on word count in the Delphi results alone, the concept of state appears to be a central concept of digital logic. Several concepts and skills directly address state or depend on state and state machines: (1) State transitions; (2) Converting verbal specifications into state diagrams/tables; (6) Relating timing diagrams to state machines/circuits; (7) Understanding how a sequential circuit corresponds to a state diagram; (10) Analyzing sequential circuit behavior; (11) Converting algorithms to register-transfer statements and datapaths; and (12) Designing control for datapaths.

The argument for state as a central concept in computing has previously been made by Shinners-Kennedy[11]. Shinners-Kennedy argues that a computer essentially does two things, both of which depend on state: (1) store state and (2) manipulate state. From this perspective, a computer could essentially be broken down into the (perhaps oversimplified) diagram in Figure

1. A computer consists of state storing devices (memory and registers), state manipulating devices (arithmetic, logic, and shift units), and these two classes of devices are controlled by a separate state machine (control unit): Essentially a small state machine that controls a larger state machine.

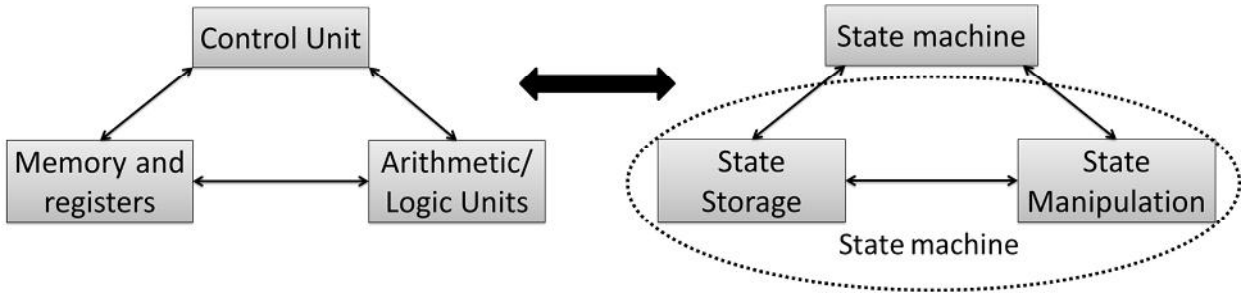

Figure 1: Left – A simplified block diagram of a computer. Right – A demonstration of the centrality of state within the block diagram.

Research into students' misconceptions about state and sequential circuits has also revealed the centrality of the concept of state. In their study of student misconceptions of state, Herman et al. revealed that the typical student uses about four incompatible conceptions of state[10]. Only those students who had a consistent, correct conception of state were able to consistently solve a wide variety of problems related to state and sequential circuits.

4.2 Fixed-length information encoding

The second core concept, fixed-length information encoding, is more subtle in its appearance than state, but it is potentially equally important and powerful in creating a strong conceptual understanding of digital logic. The concept of fixed-length information encoding appears in seven of the fifteen most important topics from the Delphi study. This concept appears most readily in concepts (9) Number representations and (14) Binary arithmetic. Because information in a computer is stored in fixed-length registers, the discussion of number representations and binary arithmetic in digital logic must be situated within that context. Furthermore, some number representations that are unique to computer architectures, such as two's complement representation, can be properly understood only within the context of fixed-length representations.

The concept of fixed-length information encoding also undergirds the design of MSI components (3) and of state machines (7). Multiplexers, decoders, read-only memory (ROM), and random-access memory (RAM) are all components that incorporate the concept of addressing – a form of fixed-length information encoding. The selection bits of a multiplexer, the data inputs of a decoder, the address bits of ROM and RAM all encode information in a fixed-length representation. Similarly, the state of a state machine is encoded within the fixed-length representation of the flip-flops or latches within a sequential circuit implementation.

The importance and interconnectedness of information encoding within these components has been revealed in students' misconceptions. For example, prior research has shown correlations between students' struggles with differentiating address and data bits for RAM and their understanding how the state can be minimally encoded in a state machine and their understanding of how to assign select bits within multiplexers[12].

The concept of fixed-length information encoding appears in the process of hierarchical design (5), modular design (8), and debugging and troubleshooting (13). In order to design efficient and effective communication between components, students must understand how to encode information with a fixed number of bits. Conversely, if students understand how information is encoded with a fixed number of bits in modular designs, they can more easily test and debug their circuits.

Finally, the fixed-length information encoding concept appears whenever computer programs and instructions are treated like any other kind of data: microinstructions in control units, machine-level instructions, compilers, and even Turing machines. This concept is pervasive in computing.

4.3 Ability to switch between levels and types of abstractions

The final core idea lies somewhere between a concept and a skill. Digital logic instructors value the development of rigorous, structured design and evaluation skills – particularly the ability to move fluidly between levels and types of abstractions. This skill manifests in almost every concept and skill, from the basic digital abstraction of treating voltages as 1s and 0s to the large scale abstractions of datapaths and control units.

Several concepts require students to convert information from one abstraction to another: (2) converting verbal specifications to state diagrams/tables; (4) converting verbal specifications to Boolean expressions; (6) Relating timing diagrams to state machines/circuits; and (7) Understanding how a sequential circuit corresponds to a state diagram. Conversions between different number representations (e.g., binary to hexadecimal) also provide an opportunity for students to practice switching between types of abstractions.

Other concepts require students to represent information with different levels of abstraction: (5) hierarchical design; (8) modular design; (13) debugging, troubleshooting, and designing simulations; and (15) Using CAD tools. When students are creating larger circuits, they must learn how and when to use abstraction to simplify a design problem. Similarly, when students encounter unexpected bugs when designing with CAD tools, they must be able to unpack these layers of abstraction to discover why a module operates incorrectly, or why two modules fail to communicate properly.

Whether students are changing the type or level of abstractions, they need to understand that different abstractions present different strengths or weaknesses for the design or analysis processes: Different representations of information can make solutions more transparent. For example, a Boolean function can be represented by different abstractions such as a Boolean expression, a truth table, or a Karnaugh map, but one representation is easier than the others to read or use in different circumstances.

5 Conclusions and Implications for Instruction

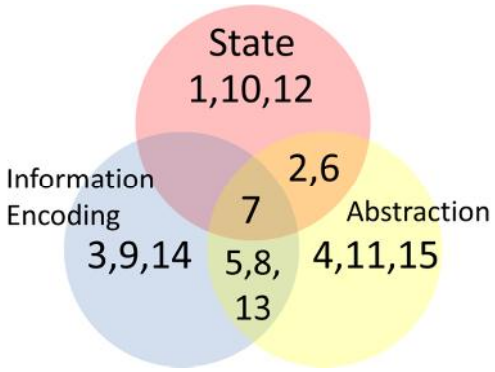

Figure 2 – Visual classification of the top 15 most important concepts and skills in digital logic and how they relate to our three core concepts.

We believe that the core conceptual framework of digital logic and computer architecture can be summarized with three core underlying concepts and skills: (1) state, (2) fixed-length information encoding, and (3) ability to switch between levels and types of abstractions. These core concepts can provide a basis that will help students organize and understand the concepts and skills of digital logic.

5.1 Benefits for student learning

Generally, students tend to organize their knowledge from a course in a disjoint fashion (i.e., fail to make connections between related concepts) or a linear fashion based on the order that topics were presented (See Figure 3 for a linear order for digital logic). However, experts tend to organize their knowledge with hierarchical structures or as an intricate interconnected web of concepts (See Figure 4)[15]. Students often struggle to think of each concept in digital logic as an isolated problem or idea[14]. So, rather than using their knowledge of one concept to aid their understanding of the next, students often relearn the same idea for each new presentation of these central concepts. For example, when students learn about the selection inputs of a multiplexer, they do not readily recognize that the data inputs of decoder or the address inputs of a RAM embody the same concept, fixed-length information encoding. Alternatively, students struggle to solve structurally identical problems – problems that share the same solution strategy. A student who can build a large (e.g., 16-to-1 multiplexer) from smaller multiplexers (e.g., 4-to-1 multiplexers) will struggle to assign the address bits when building a large (16x1) RAM from smaller (4x1) RAMs even though these two problems are structurally identical.

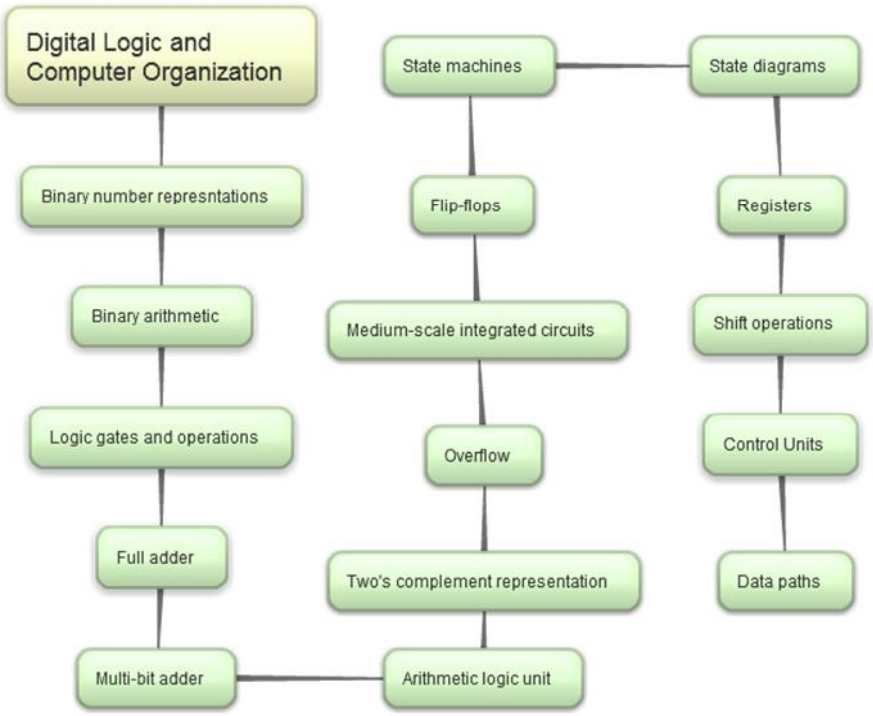

Figure 3: Caricature of the linear knowledge structure of a student

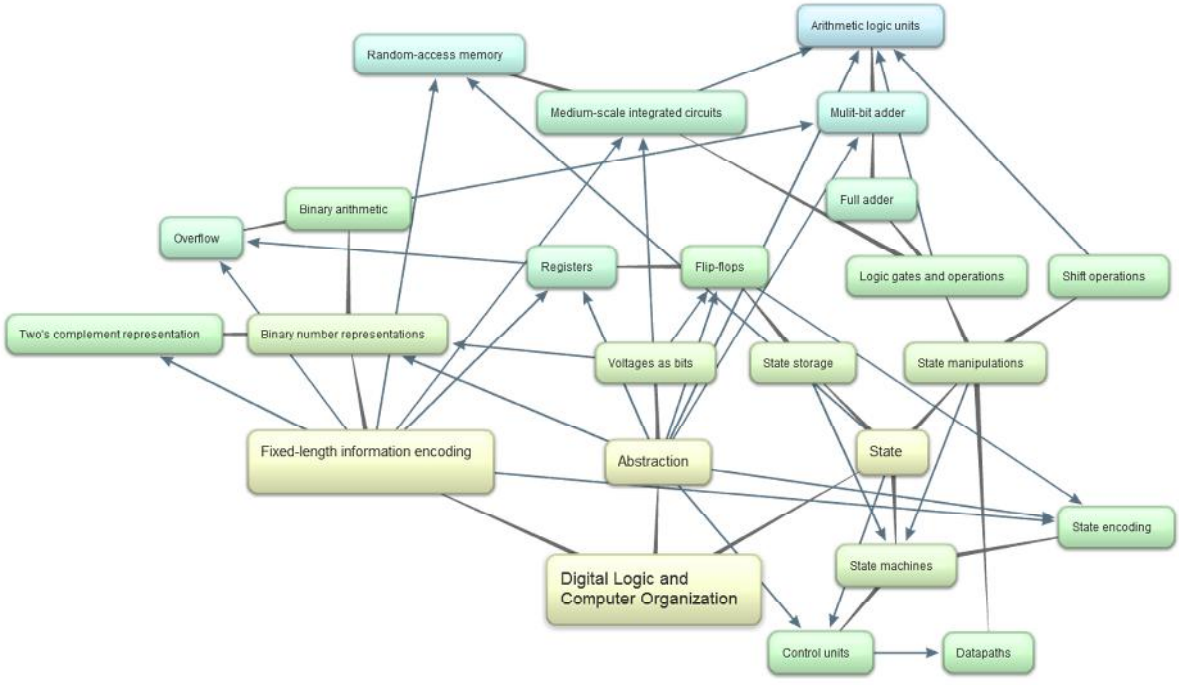

Figure 4: Example, non-comprehensive interconnected knowledge structure of an expert

Computer science majors and electrical engineering majors are often required to take a digital logic course, but most probably do not know why this course is required and are demotivated when they do not see the relevance of what they are learning[16]. By teaching students a focused, conceptual core, instructors can improve students' motivation by helping students to see how the concepts and skills that they learn will be applicable throughout their learning and careers. Each of the three core concepts that we identified can be applied through computer science and electrical and computer engineering. For example, the concept of state permeates communications, control, and signals and systems in Markov models and the design of digital filters. The concept of state is also critical in understanding recursion, iteration, and the assignment of values to variables in computer programming. The concept of information encoding is also invaluable in communications. The ability to move between levels and types of abstractions is valuable in all fields of engineering, but is particularly valuable in the design of large computer programs.

By emphasizing a conceptual framework in our instruction, we can also better prepare students for future learning. By illustrating and emphasizing the interconnectedness of concepts in digital logic, we can help students look for these similarly complex webs of knowledge in other disciplines.

5.2 Benefits to instructors

Digital logic, like many engineering and science courses, is susceptible to "content creep" as each advance in the field or each change of instructors adds new content to the course. During our Delphi poll some of the experts complained that they could not cover all of the "basics" in a first course on digital logic. Part of the challenge of fighting against content creep is that often instructors have no framework for discussing or establishing the "basics." What minimal level of knowledge do we want our students to know when they leave our digital logic courses? This core conceptual framework provides a focal point to help establish the basics.

With a core conceptual framework, digital logic instructors have a reliable way to decide what topics or skills are necessary and what topics or skills are negotiable. If instructors can focus their instruction on these three concepts and skills, they can more easily ascertain how well

their students have learned the core material. Instructors who teach more advanced courses can also know what they can expect students to have mastered.

Finally, the identification of core conceptual frameworks can facilitate the design of curriculum and the evaluation of those curricula for accreditation. Many curriculum design procedures recommend "spiral curricula" that teach students the same concepts repeatedly, but add depth and breadth to these concepts with each pass. The creation of tightly focused conceptual core can facilitate the creation of this curriculum spiral. In order to create valid and reliable assessments of courses and curricula, we similarly need to know what topics and skills are essential to our curricula and what topics and skills are peripheral. Assessments should then focus on the core skills to create short, but meaningful assessments. Core conceptual frameworks can provide this clarity and meaning to assessments. We believe that this initial effort can begin a conversation to bring greater clarity to the instruction in digital logic.

5.3 Future research directions

This paper documented an initial effort to establish a core conceptual framework for digital logic. It relied upon a Delphi poll and misconceptions research data. Future research could further validate this core by better understanding how experts organize their knowledge of digital logic. For example, we could interview several digital logic instructors and ask them to create their own concept maps of the subject. Alternatively, we could replicate previous knowledge organizations studies and ask faculty and students to organize problems based on their similarity of solutions[17]. This problem organization can help reveal experts' tacit knowledge structures. As we learn more about how experts of digital logic organize their knowledge, we can build a stronger case for the core conceptual framework of digital logic.

References

1. Litzinger, T., Lattuca, L. R., Hadgraft, R., & Newstetter, W. (2011) Engineering education and the development of expertise, *Journal of Engineering Education, 100* (1), 123–150.
2. Goldman, K., Gross, P., Heeren, C., Herman, G. L., Kaczmarczyk, L., Loui, M. C., & Zilles, C. (2010). Setting the scope of concept inventories for introductory computing subject. *ACM Transactions on Computing Education*, *10* (2), 5:1–29.
3. Herman, G. L., & Loui, M. C. (2011). Administering a digital logic concept inventory at multiple institutions. *Proceedings of the* 2011 *American Society for Engineering Education Annual Conference and Exposition*, (pp. AC2011-1800). Vancouver, BC. June 26-29.
4. Irwin, J. D. & Kerns Jr., D. V. (1995). *Introduction to Electrical Engineering*. Prentice Hall.
5. Vahid, F. (2006). *Digital Design*. Hoboken, NJ: Wiley, John & Sons, Incorporated.
6. Brown, S. & Vranesic, Z. (2009). *Fundamentals of Digital Logic with VHDL Design*. McGraw Hill Higher Education.
7. Wakerly, J. F. (2006). *Digital Design: Principles and Practices*. Upper Saddle River, NJ: Pearson Prentice Hall.
8. Givone, D. D. (2003). *Digital Principles and Design*. McGraw Hill.
9. Marcovitz, A. B. (2008). *Introduction to Logic and Computer Design*. McGraw Hill Higher Education.
10. Hwang, E. O. *Digital Logic and Microprocessor Design with VHDL*. Toronto: Thomson, 2006.
11. Herman, G. L., Zilles, C., & Loui, M. C. (2011). Flip-flops in students' conceptions of state. *IEEE Transactions in Education, In Press*. DOI: 10.1109/TE.2011.2140372
12. Shinners-Kennedy, D. (2008). The everydayness of threshold concept: State as an example from computer science. In *Threshold Concepts within the Disciplines*. Sense Publishers, 119–128.
13. Herman, G. L. & Handzik, J. (2010). A preliminary pedagogical comparison study using the Digital Logic Concept Inventory. *Proceedings of the Fortieth ASEE/IEEE Frontiers in Education Conference*, (pp. F1G-1 to F1G-6). Arlington, VA, October 27-30.
14. Longino, J. T., Loui, M. C., & Zilles, C. (2006). Student misconceptions in an introductory logic design course. In *Proceedings of the 2006 American Society for Engineering Education Annual Conference and Exposition.*
15. [Insert citation from Ambrose]
16. Herman, G. L., Goldberg, D. E., & Somerville, M. (2012). Promoting students' intrinsic motivation in the lecture/discussion classroom, *In Proceedings of the 2012 American Society for Engineering Education Annual Conference and Exposition. In press.*
17. Chi, M. T. H., Feltovich, P. J. & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices, *Cognitive Science*, *5,* 121–152.