Computer Science Education
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ncse20

How do students misunderstand number representations?

Geoffrey L. Herman [a], Craig Zilles [b] & Michael C. Loui [a]

[a] Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Illinois, USA

[b] Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA

Available online: 15 Sep 2011

To cite this article: Geoffrey L. Herman, Craig Zilles & Michael C. Loui (2011): How do students misunderstand number representations?, Computer Science Education, 21:3, 289-312

To link to this article: http://dx.doi.org/10.1080/08993408.2011.611712

Computer Science Education
Vol. 21, No. 3, September 2011, 289–312

How do students misunderstand number representations?

Geoffrey L. Herman[a]*, Craig Zilles[b] and Michael C. Loui[a]

[a]*Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Illinois, USA;* [b]*Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA*

(*Received 18 April 2011; final version received 5 August 2011*)

We used both student interviews and diagnostic testing to reveal students' misconceptions about number representations in computing systems. This article reveals that students who have passed an undergraduate level computer organization course still possess surprising misconceptions about positional notations, two's complement representation, and overflow. Contrary to common opinion, these misconceptions are widespread and reveal the need for instruction that specifically targets these misconceptions. In addition, these misconceptions will serve as the basis for the creation of a standard assessment tool called the digital logic concept inventory. This concept inventory will provide a rigorous and quantitative metric to assess the effectiveness of new teaching methods.

Keywords: concept inventory; misconceptions; number representations; pedagogy

1. Introduction

In undergraduate courses in computer science and computer engineering, students learn to convert between binary, decimal, and hexadecimal representations. They learn to add and subtract numbers in unsigned binary and two's complement representation and then how to detect overflow in these arithmetic operations. Upon completing these courses, most students are able to perform these procedures proficiently. Yet why do some students not understand what they are doing?

A panel of expert instructors and textbook authors has previously rated a conceptual understanding of number representations as one of the top ten most important concepts that students need to understand upon completing their first course in digital logic or computer organization

*Corresponding author. Email: glherman@illinois.edu
A very brief version of this article appeared at the Frontiers in Education Conference in 2010 (Herman, Loui, & Zilles, 2010b).

ISSN 0899-3408 print/ISSN 1744-5175 online

DOI: 10.1080/08993408.2011.611712
http://www.informaworld.com

(Goldman et al., 2010). They specifically stated that students need to understand the relationship between representation (pattern) and meaning (value) for number systems that are used in computers. Although there was strong disagreement from some members of the panel, the panel concluded that they expected that most students would achieve mastery of this topic prior to completion of the course. The panel ultimately rated number representation related concepts as the easiest concepts in digital logic and computer organization (Goldman et al., 2010).

Based on this consensus opinion, we decided to investigate (1) whether the panel has an accurate perception of students' mastery of number representations and, if not, (2) how do students misunderstand number representations? We questioned the panel's general perception of the difficulty of number representations, because some panel members disagreed with the majority and because education research has consistently revealed that instructors are often unaware of the exact nature of their students' struggles (Halloun & Hestenes, 1985). Our skepticism is in large part motivated by the results of the force concept inventory (FCI) in physics education (Hestenes Wells, & Swackhamer, 1992). The FCI was a simple, rigorously developed multiple-choice test that assessed only students' conceptual understanding of the *force* concept in mechanics. Although it did not test students' problem-solving skills or every concept covered by a mechanics course, it provided compelling evidence that even physics students at well-regarded institutions possessed alarming misconceptions about the force concept (Hestenes et al., 1992; Hestenes & Halloun, 1995). Consequently, the FCI became a major motivation for the adoption of interactive engagement pedagogies in physics education (Evans et al., 2003; Hake, 1998; Mestre, 2005).

We are creating a digital logic concept inventory (DLCI) that we hope will similarly raise awareness of students' misconceptions in digital logic and motivate the adoption of better pedagogies in the computer science and engineering education community (Ben-Ari, 2005; Clement, 2004; McCracken et al., 2001). This study provides evidence that challenges the perception that number representations are an easy concept for students to learn. In addition, we hope that the DLCI will provide the impetus for change in digital logic instruction, by providing a way to rigorously, objectively, and empirically compare conceptual learning. By comparing conceptual learning gains, researchers and instructors can compare the effectiveness of different teaching methods.

The DLCI tests students' conceptual understanding about number representations, Boolean logic (Herman, Kaczmarczyk, Loui, & Zilles, 2011), medium-scale integrated circuits (Herman, Loui, & Zilles, 2011b), and state and sequential circuits (Herman, Loui, & Zilles, 2011a; Herman, Zilles, & Loui, 2009). This article presents how we

discovered the misconceptions for the number representations questions on the DLCI. We present a mixed-methods study where we discovered students' misconceptions through standard qualitative methods and then tested for the prevalence of these misconceptions with the DLCI. We believe that these results can be used directly by instructors of digital logic and computer organization courses to inform interventions that address students' misconceptions about number representations.

2. Background

Because of a lack of misconceptions research on students' misconceptions about number representations in the computer science domain, we surveyed the mathematics education literature for relevant literature. Mathematics education research frequently describes students' conceptual knowledge as fragmented and unreliable (Clement, 1982; Wollman, 1983). Students use the model that comes most naturally to them in a given context, even if it means they use contradictory interpretations for similar problems. Students are often able to mask their conceptual deficiencies in mathematics, because they are adept at using memorized procedures. Wollman (1983) comments that "it is possible, therefore, to develop such skills as graphing or solving simultaneous equations without developing skill in using algebra to describe and predict natural phenomena".

Mathematics education research has revealed that students' conceptual difficulties stem from overgeneralizations that students derive from the first examples that they encounter. For example, students believe that multiplication always results in a larger product (Fischbein, Deri, Nello, & Marino, 1985), fractions become greater when the numbers in the numerator or denominator increase (e.g. $\frac{2}{4} > \frac{1}{2}$) (Stafylidou & Vosniadou, 2004), or that there is a finite number of numbers between two integers (Merenluoto & Lehtinen, 2004). Students also struggle to understand the purpose of using different representations of numbers and how these representations are related (Khoury & Zazkis, 1994; O'Connor, 2001; Vamvakoussi & Vosniadou, 2007). For example, students struggle to understand that fractional notation can represent rational but not irrational numbers, but that decimal notation can represent both rational and irrational numbers.

We define a *fraction* to be any number that contains non-integer parts: rational or irrational. Fractions are particularly difficult for students to learn, and students possess persistent misconceptions about fractions even through secondary education (Moloney & Stacey, 1997;Moskal & Magone, 2000; Ni & Zhou, 2005; Stafylidou & Vosniadou, 2004; Steinle & Stacey, 1998). Researchers discovered these misconceptions by asking

students from grades (K-10) either to choose which number was greater of two numbers or to order several numbers from least to greatest (Sackur-Grisvard & Leonard, 1985; Steinle, 2004). Through the students' explanations, researchers found that students possess two common sets of misconceptions about fractions. These misconceptions can be called *longer-is-larger* and *shorter-is-larger* (Steinle & Stacey, 1998).

Students who held the *longer-is-larger* misconception emphasized the length of the numeral string in their reasoning over the weighting of the symbols in the string (Moskal & Magone, 2000; Steinle & Stacey, 1998). For example, these students might believe that 2.01 is greater than 4.3 solely because 2.01 has more symbols in the string. This type of reasoning is valid for integers, but it is an invalid overgeneralization when applied to fractions.

Students who held the *shorter-is-larger* misconception over-emphasized the size of the fractional pieces (i.e. tenths, hundredths). These students knew that hundredths were smaller than tenths and believed that any number that contained hundredths must be smaller than any number that contained tenths. For example, 2.51 would be smaller than 2.5, because the weight of the hundredths place is smaller than the weight of the tenths place. This type of reasoning is also an invalid over-generalization as the students placed too much emphasis on the presence of various weights in the numbers rather than on the combination of the symbols and their respective weights.

3. Methodology

This section describes the selection of interview subjects, the interview protocol, and the terminology used for the remainder of the manuscript. This section is adapted from previous publications (Herman, Loui, & Zilles, 2011b; Herman, Kaczmarczyk, Loui, & Zilles, 2008).

3.1. Subjects

In Spring 2008, Fall 2008, and Spring 2009, the authors interviewed 9 undergraduate students, 6 undergraduate students, and 11 undergraduate students, respectively, at a large Midwestern university. More interview subjects were added until interviews ceased to reveal more misconceptions. All students were recruited from two, large, three-credit courses on digital logic and computer organization: one each in the Department of Computer Science and the Department of Electrical and Computer Engineering. Both courses used the same textbook (Mano & Kime, 2008) and each had about 200 students per semester. All interviewed students were traditional age (18–22) undergraduates majoring in computer science, electrical engineering, or computer engineering who had just

completed one of the digital logic courses and had earned grades of B or C (1.7 to 3.3 on a 4.0 scale). These students were selected because their understanding was likely to be less complete than students with grades of A (>3.3) (i.e. more likely to have misconceptions). Pilot interviews confirmed these expectations.

We also present results from administrations of the DLCI for those misconceptions that are tested by the DLCI (Herman, Loui, & Zilles, 2010a; Herman, Zilles, & Loui, 2011). The DLCI has been administered to almost 700 students at six institutions across the US.

3.2. *Interview process and questions*

Each student was interviewed individually for 1 h about many concepts in digital logic and computer organization. Interviews were conducted in a modified ''think-aloud'' format: students were instructed to vocalize their thoughts as they solved problems and responded to questions (Ericsson & Simon, 1984). Students were paid for their participation, and all students gave written consent to be interviewed under Institutional Review Board approval (University of Illinois at Urbana-Champaign number 07026).

Students interviewed during each semester were asked a slightly different set of questions (or the questions were simply formatted differently) based on the analysis and findings from the previous round of interviews. Following the example of the mathematics misconception literature (Steinle, 2004), we primarily asked students to compare or rank multiple numbers in different bases.

Students interviewed in 2008 were given 15 pairs of numbers; each number was represented in base 2, 10, or 16. For each pair, they were asked to indicate which number was greater. Subjects were also given a series of exercises in which they were asked to add or subtract two numbers in unsigned binary representation or two's complement representation. Subjects were also asked to indicate which addition operations caused an overflow error. Finally, they were asked to provide a rationale for why computers might use two's complement representation.

Students interviewed during Spring 2009 were asked to solve the multiple-choice number representation questions from the alpha version of the DLCI (Herman et al., 2010a). These multiple-choice questions were adapted from the questions and responses of the students during the earlier interviews. The interview questions and their acceptable answers will be introduced when needed.

3.3. *Data analysis*

Interviews were analyzed using a four-step grounded theory approach as described by Kvale (1996), Miles and Huberman (1984) and Strauss and

Corbin (1998). The analysis protocol is described in depth in previous works (Herman et al. 2008). The three authors of this article analyzed the data.

Step (1) – Interview transcripts were prepared so that only the subjects' responses to the number representation problems were analyzed for this study.

Step (2) – All researchers analyzed the interviews independently without a predetermined coding scheme (Strauss & Corbin, 1998). Principles of grounded theory were used to uncover the subjects' misconceptions, because this paradigm allows the misconceptions to emerge from the data without an a priori theoretical framework that would influence the observations.

Step (3) – The three researchers met and discussed every annotation and observation that they had made. To ensure the accuracy and completeness of our coding, a unanimous decision was needed for an annotation to be included for coding or rejected from coding.

Step (4) – After all interviews were discussed, the preliminary code names and definitions were refined, and the refined list of codes and definitions were analyzed by the researchers independently to identify the thematic elements of the codes. All researchers then met again to discuss the thematic elements that they had noted. A unanimous decision about the presence of a theme or misconception was needed for it to be included in the final list of themes and misconceptions.

3.4. Terminology

This section defines terminology that is used for the remainder of the article. The term *student* describes any person who has recently learned digital logic or is currently learning digital logic. The term *subject* describes any student who participated in the interview portion of the study. All subjects are given pseudonyms such as "Subject 1."

Numbers were represented in positional notation through a parenthetical notation where $(a_2a_1a_0.a_{-1}a_{-2})_r$ should be interpreted as $a_2a_1a_0.a_{-1}a_{-2}$ in base or radix r with a period for the radix point. The *weight of a position* is defined to be the radix r raised to the power indicated by the index of the position i (i.e. r^i). For example, the number $(32)_{10}$ has a 3 that is multiplied by its weight $10^1 = 10$ and a 2 that is multiplied by its weight $10^0 = 1$. The position with weight 10 is called the tens place and the position with weight 1 is called the ones place. The summation of the each symbol multiplied by the weights of their positions is called the *value* of the number.

Bits in a binary code are often grouped as 4-bit chunks called *nibbles* when converting between hexadecimal and binary representations.

Two's complement representation is one way to represent negative numbers with a binary code. Two's complement can be interpreted as a weighted code that matches traditional positional notation except that the most significant bit has a negative weight rather than a positive weight. For example, 1001 in 4-bit two's complement representation can be interpreted as $1 \cdot (-2^3)) + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = -8 + 1 = -7$. Alternatively, for an n-bit two's complement representation, we represent a negative integer $-m$ by the (unsigned) binary representation of the value $2^n - m$. For $m = 7$, the 4-bit two's complement representation is the binary representation of $2^4 - 7 = 9$, which is 1001. Positive integers in two's complement behave like positive integers in unsigned binary representation, except that the most significant bit must be 0. Any number in two's complement representation cannot be properly interpreted without knowledge of the number of bits (n) used by the representation.

Two's complement representation is an advantageous representation primarily because it simplifies the hardware implementation of subtraction, and it simplifies the addition of positive and negative numbers. The representation also has only one representation for zero, unlike signed magnitude or one's complement representation. It is debatable whether this single representation of zero is a benefit.

The *two's complement operation* is a mathematical procedure to convert the representation of a positive number into the representation of a negative number with the same magnitude, and vice versa. The two's complement operation is often explained by using two steps: (1) complement all of the bits (change 1s to 0s and 0s to 1s) and (2) add 1. For example, in 4-bit two's complement representation of +5 (0101) becomes −5 (1011) with the following steps ($0101 \rightarrow 1010 \rightarrow 1011$). The operation is justified by noting that the two steps are equivalent to subtracting the positive integer m from $2^n - 1$ and then adding 1. The two's complement operation transforms the binary representation of m into the binary representation of $2^n - m$, and vice versa. *Signed magnitude representation* (also called *signed binary*) and *one's complement representation* are two alternative methods for representing negative numbers with a binary code.

4. Results

Subjects had generally strong procedural skills, but they still possessed many conceptual weaknesses about numbers. For example, every subject could correctly complete addition tasks and most could correctly complete subtraction tasks on numbers in unsigned binary and in two's complement representation. On the contrary, these subjects struggled to interpret the sums or differences that they found when completing these tasks. Similarly, most subjects were adept at converting numbers between

bases if asked to perform the conversion, but comparison tasks revealed faulty methods for comparing numbers and misconceptions about how to interpret these numbers. The following sections highlight the more common of these misconceptions.

4.1. Positional notation misconceptions

The interviews showed several misconceptions and novice behaviors regarding positional notation.

4.1.1. Changing the base does not change the weights (unchanging-weights)

Several subjects did not understand how changing the base of a number affected the weights of the positions. For example, subjects were asked to compare $(2B)_{16}$ with $(31)_{10}$ (note that $(2B)_{16} = 2 \cdot 16 + 11 = (43)_{10}$). When making the comparison, some subjects believed that the 16s place in hexadecimal has the same weight as the 10s place in decimal. Subject 8 revealed this misconception by treating the digit 2 in $(2B)_{16}$ as 20.

> Subject 8: "*B* is 11, so a one carries over and two plus one is three [20 + 10 = 30], so, 31. Which is the same as base ten 31."

A little later, Subject 8 confirmed this misconception when comparing $(11010)_2$ and $(2C)_{16}$ (note that $(11010)_2 = (26)_{10}$ and $(2C)_{16} = (44)_{10}$). The subject considered the digit *C* to be 12 and again treated the digit 2 as 20 (i.e. 20 + 12 = 32).

> Subject 8: "$(2C)_{16}$ so that's twelve, one two three, 32, $[(11010)_2]$ is two four, zero one two three four, two to the fourth, is less than 32. Since $[(11010)_2]$ is always less than 2^5, it's less than $[(2C)_{16}]$."

On the DLCI, two questions test whether students believe that "changing the base does not change the weights." About 15% of students chose both answers that reflect this misconception (Herman, Zilles, & Loui, 2011).

4.1.2. Bigger bases are always bigger (bigger-is-bigger)

Subjects demonstrated a misconception that numbers represented with a greater base are always greater. Subjects 9 and 10 revealed this misconception when asked to order the numbers $(1.1)_2$, $(1.4)_{10}$, and $(1.5)_{16}$ from smallest to largest. Subject 10's response shows that this misconception is held for both integers and fractions.

> Subject 9: "Obviously binary is small, so it's smallest, and then decimal is smaller than hex [Subject writes $(11)_2 < (14)_{10} < (15)_{16}$]."

> Subject 10: "A one in base 16 is obviously bigger than a one in base 2, no matter what's after [the radix point]."

On the DLCI, two questions test whether students believe that "bigger bases are always bigger." About 25% of students chose both answers that reflect this misconception (Herman, Zilles, & Loui, 2011).

4.1.3. Fractional position misconceptions

The *unchanging-weights* and *bigger-is-bigger* misconceptions manifested more often when subjects needed to compare the fractional component of numbers in different bases. Subjects used proper positional interpretations when manipulating fractions less often than when manipulating integers.

Subject 15 demonstrated the *unchanging-weights* misconception when ordering the three numbers $(1.1)_2$, $(1.4)_{10}$, and $(1.5)_{16}$.

> Subject 15: "1.5 in base 16 would be 1.5 in base 10 . . . . 1.4 in base 10 is obviously less than 1.5."
>
> Interviewer: "Can you explain to me how you know 1.5 in base 16 is the same as 1.5 in base10?"
>
> Subject 15: "Basically [hexadecimal is] the same as base 10 except for A, B, C, and D."

Subject 1 also demonstrated the *unchanging-weights* misconception when comparing the two numbers $(1.0101)_2$ and $(1.5)_{10}$.

> Subject 1: "I think [$(1.0101)_2$ and $(1.5)_{10}$] are equal . . . ."
>
> Interviewer: "Why?"
>
> Subject 1: "Because [0101] is 5 in binary."

Subject 1 manifested this misconception a second time when asked to compare $(1.101)_2$ and $(1.5)_{10}$, and even thought that she was solving the same problem as the $(1.0101)_2$ and $(1.5)_{10}$ comparison. For this subject, the 101 symbol pattern seemed to be more important than the position of the symbols.

This weakness with fractions is widespread because fewer than 50% of students could correctly compare the two numbers $(2.7)_{16}$ and $(2.7)_{10}$ on the DLCI.

4.1.4. Improper positional weights

Some subjects tried to use positional notation to interpret the different numbers, but they failed to correctly calculate the weights of the various positions. A common mistake was to interpret the ones place as having

weight r^1 rather than r^0 for radix r. For example, when interpreting $(2B)_{16}$, Subject 7 interpreted the B as, "11 times 16, whatever that is [pause] 161." Similarly, Subject 8 interpreted $(1.0101)_2$ as $1 \cdot 2^1 = 2$ plus a fractional component.

4.1.5. Hexadecimal difficulties

Subjects frequently interpreted hexadecimal numbers with techniques different from those they used to interpret binary or decimal numbers. As seen in Section 4.1.1, some subjects could interpret binary and decimal numbers correctly, but they faltered when interpreting hexadecimal numbers.

Subject 16 demonstrated a unique and surprising difficulty with hexadecimal numbers.

> Subject 16: "$[(1.4)_{10}]$ was 1.4, $[(1.100)_2$ was] 1.5, and $[(1.5)_{16}$ was] 24."
>
> Interviewer: "How did you come up with 24?"
>
> Subject 16: "1.5 times 16."

Subjects rarely interpreted hexadecimal numbers by using a positional notation interpretation. Subjects preferred to convert hexadecimal numbers to binary first or they used one of the misconceptions described earlier to interpret hexadecimal numbers. This reliance on converting to binary first is not a misconception; but the subjects' over-reliance on this one strategy is indicative of novice behavior that conceals the subjects' misconceptions.

4.1.6. Borrowing during subtraction misconceptions

Subjects' responses indicated that they did not understand positional weighting when they were asked to subtract two numbers. A few subjects expressed dislike for performing the subtraction operation in binary, and several refused to perform the task. Subjects 3, 6, and 20 all made mistakes while performing the borrowing operation in subtraction. They should have borrowed $(10)_2$ when borrowing from a higher order position. As can be seen in Figure 1, Subject 6 borrowed $(1)_2$ instead and

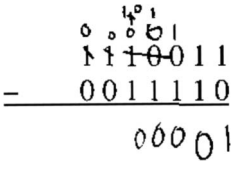

Figure 1. Subtraction operation where a subject borrowed 1 rather than 10.

found a difference of $(0000001)_2$ rather than $(1010101)_2$. Subject 20 was asked about why he borrowed 1:

> Interviewer: ''So what's this 1 that you put above the numbers?''
>
> Subject 20: ''Let's see, I tried looking at this addition, and how carry works here, and I did it backwards here.''

4.2. Two's complement representation misconceptions

Almost every subject could recall the two's complement operation of ''complement the bits and add one.'' A few subjects forgot the details of this operation and simply complemented the bits, added two instead of one, or subtracted one instead of adding one. This recollection of the operation was often isolated from an understanding of how to interpret numbers in two's complement representation and the reason why two's complement representation is used in computers.

4.2.1. Difficulties with interpreting two's complement representation

Students rarely demonstrated a deep understanding of how to interpret numbers in two's complement representation. For example, few subjects understood that two's complement representation could be interpreted as a weighted positional code. Some subjects discarded the most significant bit when adding numbers in two's complement representation because it was only a sign bit. These subjects determined the appropriate sign bit for the sum through inferences (i.e. two positive numbers should sum to a positive number) rather than through simply adding the numbers together with the most significant bit included in the addition.

This lack of understanding was further revealed as Subjects 5, 6, and 7 struggled to differentiate between the ''two's complement representation'' and the ''two's complement operation.'' These subjects thought that the two's complement operation was the method for interpreting numbers in two's complement representation. For example, Subject 6 interpreted a number 0100101 in 7-bit two's complement representation as $(1011011)_2$ or $(91)_{10}$ rather than as $(37)_{10}$:

> Subject 6: ''I think you invert all the bits, and then add one, or maybe you add two. I think it's one. [Subject writes 1011010 (the complement of the bits) and then 1011011 (the complement + 1).]''

On the DLCI, two questions ask students to interpret the sum from an addition operation performed on integers in two's complement representation. Only 36% of students could choose the correct interpretation on the first of these problems and 46% have correctly chosen the correct interpretation on the other (Herman, Zilles, & Loui, 2011).

4.2.2. Confusion about the purpose of two's complement representation

When asked why they might use two's complement representation in a computer, subjects rarely cited the primary reasons given in Section 3.4. Subjects often explained that they would use two's complement representation because (1) it has only one representation for zero and (2) it can represent more numbers than any other representation.

Subjects' emphasis on purpose (1) was accompanied by other conceptual deficiencies. For example, Subject 5 did not believe that numbers in two's complement representation could be added directly. When adding numbers in two's complement representation, this subject first converted the numbers to unsigned binary and then added or subtracted the numbers depending on their relative signs.

Subjects 12 and 16 both claimed purpose (2) and said that two's complement could represent more numbers than even unsigned binary.

> Subject 16: "I know you can represent one more number [with two's complement] than either one's complement or signed magnitude. And then I think it's one more than unsigned."

Subject 1 claimed that two's complement could represent the same quantity of numbers as signed magnitude with one fewer bit.

> Interviewer: "Why do we use two's complement?"
>
> Subject 1: "It saves an extra bit. If you're doing signed [magnitude] of -7 then you have to have a 1 out in front of $(111)_2$ [1111] to represent -7, but if you do it in two's complement, you only have three bits for two's complement [writes 111 then $000 + 1 = 001$]."

When asked on the DLCI to identify the primary reason for why two's complement representation is used, only 60% of students could do so (Herman, Zilles, & Loui, 2011).

4.2.3. Aliasing of the different representations of negative numbers

Several subjects failed to properly differentiate the different properties of the three representations of negative numbers they had learned. For example, some subjects performed addition and subtraction in two's complement representation as if the numbers had been in signed magnitude numbers or they interpreted numbers in two's complement representation as though they were in signed magnitude representation.

> Subject 3: "I'm having trouble distinguishing in my memory signed magnitude from two's complement and one's complement. But mostly the addition we did was with two's complement."

4.3. Fixed length representation misconceptions

Subjects struggled most with understanding the implications of the fixed lengths of registers in a computer. Along with this conceptual difficulty, they struggled to understand the concept of overflow.

When asked to give an example of overflow, Subject 1 gave an example of adding two numbers represented with three bits, but she evaluated the sum as a number represented with four bits:

> Subject 1: ''If you add two 7's [writes $111 + 111$] together and [writes 1110 then pauses] actually that would still be [pause]. Let's try something else [writes $111 + 101 = 01$ then pauses] I'm trying to think of something that overflows.''

Other subjects similarly struggled to use the same number of bits when interpreting the addends but a different number of bits when interpreting the sum.

Subjects' misconceptions about, or lack of conception of, the fixed length of number representations in computing systems were further revealed in other overflow questions. Only a couple of subjects stated that overflow happens because of the fixed length of registers. Most subjects used two imprecise and non-generalizable definitions of overflow: (1) overflow occurs when there is a carry-out of 1 from the most significant bit in an addition, or (2) overflow occurs when the addition of two positive numbers results in a negative number or when the addition of two negative numbers results in a positive number. The first definition is true specifically for fixed-length unsigned binary addition; the second definition is true specifically for two's complement representation.

When subjects were asked to decide whether a two's complement addition resulted in overflow, they often failed to synthesize these two definitions into one cohesive definition and instead pitted these two definitions against each other. Some subjects like Subject 17 were able to choose one definition over the other; others simply became more confused as they solved more problems.

> Subject 17: ''Overflow could be defined in one of two ways. You can either define overflow as when the operation, the addition operation overflows – there is a carry bit that can't fit into the result. I did not think that was the likely answer because, it says two's complement. Here this is not technically an adder overflow, however this is a positive number, and a positive number, and if you're representing [the answer] with two's complement, this is a negative number, which is not the correct functionality of an addition operation.''

> Subject 3: ''So that's a positive and a negative number being added together. I think there's overflow when there's this extra bit [circles carry-out from most significant bits]. Let's do a different one, maybe I'll do better. I remember you're adding two positives and you get a negative

value, so that just means, I think that means overflow occurs, because that's just impossible. Wish I knew what that extra bit meant!''

Subjects like Subject 3 often insisted that the system must do something with the additional carry-out of 1. On the contrary, they easily believed that a carry-out of 0 could be ignored. This inconsistent treatment of the carry-out bit reveals an understanding of numbers in computers that is not rooted in the architecture of the computer.

When asked on the DLCI to identify whether two different two's complement additions result in overflow, only 44% of students could properly identify which addition resulted in overflow and which one did not (Herman, Zilles, & Loui, 2011).

5. Discussion

Although the subjects in the study were taught in different departments by different instructors, the two courses presented number representations in similar fashion. A survey of textbooks revealed that these topics are commonly presented in similar fashion (Brown & Vranesic, 2009; Givone, 2003; Hwang, 2006; Irwin & Jr., 1995; Mano & Kime, 2008, Vahid, 2006; Wakerly, 2006). Based on this survey, we suspect that the presentation of number representations is similar at many institutions. After reviewing how these topics were taught in the classroom, we offer some hypotheses for how subjects developed the misconceptions documented in Section 4.

5.1. Presentation of number representations

In both courses, instructors began by teaching students about binary numbers and why computers use the binary number system. Instructors taught the general principles of positional notation, and they showed methods for converting from decimal to other radices – particularly between binary and decimal. Instructors also taught ''tricks'' such as chunking bits into 3-bit chunks or 4-bit nibbles to convert from binary to octal or hexadecimal, respectively. For example, $(10110010)_2$ could be chunked as (010 110 010) to yield $(2\ 6\ 2)_8$ or chunked as (1011 0010) to yield $(B\ 2)_{16}$. In both courses, instructors emphasized that hexadecimal and octal are convenient bases for representing long strings of bits.

After instructors covered the basics of positional notation, they taught binary addition and subtraction, and they showed how to implement a binary adder in digital logic. Instructors introduced the concept of overflow as a carry-out of 1 from the addition of the most significant bits. Instructors then taught signed magnitude representation, one's complement representation, and two's complement representation. They demonstrated how two's complement representation simplifies

subtraction. They created several examples to show that two's complement representation did not have two encodings for 0, and they highlighted some historical examples where the two encodings for 0 proved problematic. Finally, instructors reintroduced the concept of overflow and how to detect it in two's complement additions.

After several weeks of learning about combinational circuits and sequential circuits, students were taught about memory and registers.

5.2. Connections between presentation and misconceptions

5.2.1. Hexadecimal misconceptions

Subjects generally demonstrated a weak conceptual understanding of hexadecimal representation. For example, subjects made more mistakes when comparing numbers in hexadecimal and decimal representations than when comparing numbers in binary or decimal representations. These mistakes were often paired with *unchanging-weights, bigger-is-bigger*, or *improper positional weights* misconceptions. Instead of using positional-notation reasoning, subjects opted to convert hexadecimal numbers to binary first and then convert from binary to decimal (by using a memorized conversion method). In other words, subjects used procedural knowledge rather than conceptual knowledge. Teaching both the conceptual basis and the convenient procedure at the same time did not foster a solid conceptual understanding of hexadecimal representation.

5.2.2. Two's complement misconceptions

Subjects struggled to understand the purpose of two's complement representation (Section 4.2.1), and they confused the different representations of negative numbers with each other (Section 4.2.3). These difficulties may have arisen because the three representations of negative numbers were introduced at the same time. Because the subjects learned the three representations at the same time, they likely built their understandings of these representations with reference to each other. For instance, subjects learned two's complement representation as a way to represent negative numbers that was better than signed magnitude or one's complement rather than as a distinct number representation system akin to the positional notations. Consequently, subjects struggled to differentiate their knowledge of two's complement representation from the other representations. This lack of distinction may have caused our subjects to treat numbers in two's complement representation as though they were in signed magnitude or one's complement representation. In addition, this method of instruction may have caused students to over-value a single representation of 0. Since the subjects' understanding

of two's complement representation was developed through a comparison of multiple representations, they focused on the simplest distinctions between two's complement representation, one's complement representation, and signed magnitude. Because they learned this simple distinction, many subjects did not learn the primary advantage and purpose of two's complement representation.

5.2.3. Overflow misconceptions

There are two possible reasons for how instruction supported the development of students' difficulties with overflow: (1) a *long time elapsed* between when students learned about overflow and when they learned about memory storage; and (2) students learned about addition and the possibility of overflow *before* they learned about fixed-length registers. The length of time between instruction on overflow and fixed-length registers makes it more difficult for subjects to build an appropriate association between these two concepts. Similarly, because subjects learned about overflow before they learned about fixed-length registers, they could not build their concept of overflow on an appropriate conceptual framework. For example, when subjects were asked to "add two 4-bit numbers," they did not fully understand that the sum must also be only four bits, and they easily dismissed the constraint of four bits when solving the addition.

Since a conceptual understanding of overflow requires an underlying conceptual understanding of the structure of a register in a computer, many subjects could not appropriately solve overflow problems. Instead, subjects often resorted to operational/situational definitions of overflow such as "overflow is when the most significant bit in an addition results in a carry-out" or "overflow happens when the addition of two positive numbers results in a negative number." Furthermore, the subjects struggled to understand that when determining the sum of two integers, computers do not store the carry-out from the most significant bit position with the sum, regardless of whether it is 0 or 1.

If the fixed length of registers were taught first, the instructor would not need to introduce two different types of overflow: one for addition in unsigned binary addition and one for addition in two's complement representation. This order of instruction may also reinforce the concept of fixed length registers in students' minds as they use the concept in multiple contexts.

5.3. Procedural learning versus conceptual learning

As can be seen in the previous section, many of our subjects' difficulties and misconceptions were caused by their emphasis on learning procedures

rather than concepts. When asked, subjects could easily convert from one radix to another, but when asked to compare two or three numbers with different radices, they struggled to choose appropriate procedures or concepts for comparing these numbers. When subjects were not given a familiar task or told which procedure to use, their misconceptions appeared.

Our subjects' emphasis on procedural knowledge was also demonstrated by the way they used borrows and carry-outs. Subjects borrowed the wrong value during subtraction, and they did not know what to do with the carry-out from the addition of the most significant bits. The subjects who made this mistake knew approximately what they needed to do, but they did not know how to interpret their procedures.

5.4. Integer and fraction misconceptions

Our subjects manifested misconceptions about number representations more frequently when asked to manipulate fractions. Certainly, some of the subjects' difficulties can be attributed to less experience with fractions, but there are patterns in the students' misconceptions that suggest that there is also a more deeply rooted conceptual problem. Our subjects' misconceptions with fractions are the same as their misconceptions about integers, and their misconceptions are similar to the misconceptions of students in grades K-10 (Sackur-Grisvard & Leonard, 1985; Steinle, 2004).

Although students revealed misconceptions less frequently when manipulating integers, they still occasionally revealed the same misconceptions about integers as they did about fractions. This lower frequency of misconceptions can be explained by two factors. First, the misconceptions that are detrimental to the interpretation of fractions are not as detrimental to the interpretation of most integers. For example, a student may correctly complete the task of comparing $(27)_{10}$ and $(27)_{16}$ while reasoning with the *bigger-is-bigger* misconception. Second, students' procedural knowledge for manipulating integers is likely more rehearsed than their procedural knowledge for manipulating fractions. Consequently, students have a wider array of procedures to correctly compare two integers, and this array of procedures can mask their misconceptions.

The misconceptions demonstrated by our students are similar to those of K-10 students, even though our students have received additional training on how to understand number representations. K-10 students had two primary misconceptions about fractions: *longer-is-larger* and *shorter-is-larger* (see Section 2). Both misconceptions are overgeneralizations of types of reasoning and procedures that work when comparing integers, but not when comparing fractions. When students use *longer-is-larger* reasoning, their reasoning focuses on the symbols (quantity or which characters are used) rather than the positions of the

symbols. Their conception of numbers emphasizes the symbols that are used regardless of the positions that are used. Our subjects demonstrated similar reasoning fallacies when using the *unchanging-weights* misconception. When subjects claimed that $(1.4)_{10}$ is less than $(1.5)_{16}$, their reasoning similarly focused on what symbols were used and not the weighting of the symbols.

When students use *shorter-is-larger* reasoning, they reason that the size of one or two weights can be used to compare two numbers (note that the name of this misconception is a little misleading because the actual length of the number is less important in the students' reasoning than the weights of the positions in the numbers). This reasoning works for integers: when one number has weights in the thousandths (e.g. 1234) while the other has weights only in the hundredths (e.g., 789), one can determine that the first number is larger by looking solely at the weights that are present in the two numbers. This reasoning fails for fractions: when the representation of a number has weights in the thousandths (0.651) while the other has weights in the hundredths but not the thousandths (0.65), one cannot determine the relative size of the two numbers by looking solely at the weights that are present in the two numbers. On the contrary, students who possess the *shorter-is-larger* misconception would incorrectly claim that 0.651 is smaller than 0.65 solely because the first representation is composed of thousandths while the second is composed of hundredths (i.e. thousandths $<$ hundredths therefore $0.651 < 0.65$). *Shorter-is-larger* is a reasoning scheme that leads students to believe that the relative magnitude of two numbers can be determined by comparing the weights that are present in each number. The *bigger-is-bigger* misconception similarly inappropriately compares two numbers by comparing only the weights that are present in the different bases. For example, students with the *bigger-is-bigger* misconception believe that a number represented by hexadecimal (e.g. $(2.7)_{16}$) is larger than a number represented in decimal (e.g. $(2.7)_{10}$) because the weighting for the hexadecimal number is based on 16 rather than 10. Because 16 is greater than 10, these students reason that the hexadecimal number must be greater than the decimal number.

Because our subjects' struggles with fractions resemble the misconceptions of younger students, we cannot conclude whether these misconceptions are a result of instruction or of students' preconceptions or patterns of reasoning that they possessed prior to instruction.

5.5. Limitations

The observations and conclusions of this study were created through a qualitative research study. As a result, the study intended to primarily provide detailed descriptions of our subjects' mistakes and to provide a

rich theoretical foundation that will guide future quantitative or mixed-methods studies. These descriptions can validate and add meaning to these future quantitative studies. This study is not intended to compare how different teaching methods affect the prevalence of misconceptions. Future quantitative and mixed methods studies can allow us to assess the prevalence of these misconceptions in the general student population and to measure the effect that different teaching methods have upon the generation and persistence of the documented misconceptions.

Although the interviews were conducted on only one campus, we believe that these results can be reasonably generalized to other campuses. Administrations of the DLCI have revealed that students at other institutions possess similar misconceptions at comparable rates to the interviewed students.

6. Conclusions

This study revealed students' misconceptions about positional notation, two's complement representation, and overflow. This study also revealed that students can mask their misconceptions through their reliance on procedural knowledge. These results are critically important, because they defy the expectations of many instructors. During our Delphi survey of expert instructors and textbook authors, the experts rated number representations as the easiest of the concepts that students need to learn in digital logic and computer organization (Goldman et al., 2010). Some experts even asserted that their students had mastered this topic before coming to the university. Results from the DLCI provide strong evidence that this common perception among instructors might be devastatingly incorrect. About 40% of students do not understand why two's complement is used in computers. Over 40% of students still exhibit the *bigger-is-bigger* or *unchanging-weights* misconceptions about positional notations. Over 50% of students cannot reliably determine whether overflow occurs during an addition operation in two's complement representation. Finally, over 50% of students cannot reliably interpret numbers in two's complement representation. These results indicate that typical computer engineering and computer science students can progress through their first few years without an accurate, consistent conceptual understanding of how numbers are used and represented in computers.

Because these misconceptions are so prevalent, we suggest several implications for instruction and research.

6.1. Implications for instruction

Unsurprisingly, students tend to think of numbers according to the procedures and computations that they perform on them. Instructors

should find creative ways to help students use and practice deeper conceptual knowledge. Instructors may be able to help students avoid this over-reliance on procedures by asking students to solve problems that are simple, but do not resemble a common, well-rehearsed procedure. For example, instructors should ask students to generate an example of addition that overflows rather than ask them to complete a pre-written addition problem and determine whether overflow occurs. If instructors ask students to perform ranking tasks such as ''order these four numbers from least to greatest,'' they can help students practice a variety of procedures. By being forced to use a variety of procedures, students must use their conceptual knowledge to choose the best procedures. As students practice using their conceptual knowledge more, they will develop strong conceptual understandings rather than just facility with a memorized procedure.

To strengthen students' understandings of hexadecimal representation, instructors should teach hexadecimal as a positional notation and perhaps introduce the 4-bit nibble conversion trick at a later time.

We suggest that students should be taught extensively about fixed-length registers and how numbers are stored in computers before they are taught about binary arithmetic/overflow or two's complement representation. Students have a weak grasp of the fixed-length of registers in a computer and therefore struggle to fully comprehend that number representations in computers likewise have fixed lengths. Teaching binary number systems and then teaching the constraints that computer architectures place on these representations naively seems like a good way to build on students' prior knowledge, but we recommend that this order of instruction should be inverted, because students' prior knowledge seems to be a hindrance without the proper context.

Instructors must also be careful to provide consistent, precise presentation of number representation concepts. Through a survey of several textbooks (Brown & Vranesic, 2009; Hellerman, 1967; Hwang, 2006; Givone, 2003; Irwin & Jr., 1995; Mano & Kime, 2008; Marcovitz, 2008; Patt & Patel, 2003; Vahid, 2006; Wakerly, 2006), we found several examples where textbooks fail to maintain appropriate levels of rigor. For example, in one textbook, students are asked to do the following:

> Add the following 2's complement binary numbers. Also express the answer in decimal.
>
> (1) $01 + 1011$
> (2) $11 + 01010101$
> (3) $0101 + 110$
> (4) $01 + 10$.

We suppose that this question was intended to test students' understanding of sign extension, but this problem is still ambiguous at best. A two's complement representation requires a specific number of bits, but the authors' failure to specify a number of bits may reinforce the typical students' weak grasp of fixed-length representations. Furthermore, the conversion to decimal also requires knowledge of the number of bits in the representation. We have also observed similar mistakes as authors use the term *two's complement* to mean either the *two's complement operation* or the *two's complement representation*.

Because students struggle to properly understand two's complement representation, we suggest that instructors teach only the two's complement representation when introducing representations for negative numbers. Although understanding the history of how computers developed and the alternative designs of computers are important to understand, this knowledge seems to cause more confusion than deep, accurate conceptual understanding – at least in a first course on digital logic or on computer organization.

6.2. Implications for research

The interviews revealed that, despite additional training in computer organization, students' misconceptions about numbers are similar to younger students' misconceptions about numbers. Testing students prior to instruction or conducting interviews before instruction could reveal whether students' misconceptions are a result of pre-existing misconceptions or a result of instruction.

Future research could investigate whether the suggested changes to instruction help remedy students' misconceptions about numbers. The DLCI tests these misconceptions and provides a way to measure the effectiveness of new teaching methods (Herman et al., 2010a). Detailed research on students' response patterns on the DLCI may reveal even more insight into the formation of students' misconceptions. For example, researchers could compare students who were taught about fixed-length registers first with students who were taught about representations and binary arithmetic before they were taught about registers. Differences in students' scores may reveal that the order of instruction affects how students develop their conceptual frameworks. Furthermore, students' response patterns could reveal whether these misconceptions are isolated to only poorly performing students or whether they are widespread among all students regardless of overall performance. Requests to access the DLCI should be directed to Herman.

Acknowledgments

The authors thank Joe Handzik for transcribing the interviews and Lisa Kaczmarczyk for helping to develop analysis protocols. This work was supported by the National Science Foundation under grants DUE-0618589. The opinions, findings, and conclusions do not necessarily reflect the views of the National Science Foundation or the authors' institution.

References

Ben-Ari, M. (2005, February). *The Concorde doesn't fly anymore*. Keynote address, presented at the technical symposium on computer science education (SIGCSE). Retrieved from http://stwww.weizmann.ac.il/g-cs/benari/articles/concorde.pdf

Brown, S., & Vranesic, Z. (2009). *Fundamentals of digital logic with VHDL design*. New York, NY: McGraw Hill Higher Education.

Clement, J. (1982). Algebra word problem solutions: Thought processes underlying a common misconception. *Journal for Research in Mathematics Education*, *33*(1), 16–30.

Clement, J. (2004). A call for action (research): Applying science education research to computer science instruction. *Computer Science Education*, *14*, 343–364.

Ericsson, K.A., & Simon, H.A. (1984). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.

Evans, D.L., Gray, G.L., Krause, S., Martin, J., Midkiff, C., Notaros, B.M., ... Wage, K. (2003). Progress on concept inventory assessment tools. In *Proceedings of the 33rd ASEE/IEEE frontiers in education conference* (pp. T4G1–T4G8). New York, NY: IEEE.

Fischbein, E., Deri, M., Nello, M., & Marino, M. (1985). The role of implicit models in solving problems in multiplication and division. *Journal for Research in Mathematics Education*, *16*, 3–17.

Givone, D.D. (2003). *Digital principles and design*. New York, NY: McGraw Hill.

Goldman, K., Gross, P., Heeren, C., Herman, G., Kaczmarczyk, L., Loui, M.C. & Zilles, C.B. (2010). Setting the scope of concept inventories for introductory computing subjects. *ACM Transactions on Computing Education*, *10*, 1–29.

Hake, R. (1998). Interactive-engagement vs traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, *66*, 64–74.

Halloun, I.A., & Hestenes, D. (1985). The initial knowledge state of college physics students. *American Journal of Physics*, *53*, 1043–1055.

Hellerman, H. (1967). *Digital computer system principles*. New York, NY: McGraw Hill.

Herman, G.L., Kaczmarczyk, L., Loui, M.C., & Zilles, C. (2008). Proof by incomplete enumeration and other logical misconceptions. In *ICER '08: Proceedings of the fourth international workshop on computing education research* (pp. 59–70). Sydney, Australia: ACM.

Herman, G.L., Kaczmarczyk, L., Loui, M.C., & Zilles, C. (2011). Describing the what and why of students' difficulties in Boolean logic. *Transactions on Computing Education* (in press).

Herman, G.L., Loui, M.C., & Zilles, C. (2010a). Creating the digital logic concept inventory. In *Proceedings of the 41st annual ACM technical symposium on computer science education* (pp. 102–106). Dallas, TX: ACM.

Herman, G.L., Loui, M.C., & Zilles, C. (2010b). Work in progress: How do engineering students misunderstand number representations? In *Proceedings of the 40th ASEE/IEEE frontiers in education conference* (pp. T3G1–T3G2). Arlington, VA: IEEE.

Herman, G.L., Loui, M.C., & Zilles, C. (2011a). Flip-Flops in students' conceptions of state. *IEEE Transactions on Education* (in press). DOI: 10.1109/TE.2011.2140372

Herman, G.L., Loui, M.C., & Zilles, C. (2011b). Students' misconceptions about medium-scale integrated circuits. *IEEE Transactions on Education* (in press). DOI: 10.1109/TE.2011.2104361

Herman, G.L., Zilles, C., & Loui, M.C. (2009). Work in progress: Students' misconceptions about state in digital systems. In *Proceedings of the 39th ASEE/IEEE frontiers in education conference* (pp. T4D1–T4D2). San Antonio, TX: IEEE.

Herman, G.L., Zilles, C., & Loui, M.C. (2011). Administering a digital logic concept inventory at multiple institutions. In *Proceedings of the 2011 American Society for Engineering Education Annual Conference and Exposition* (Paper #: AC 2011–1800). Washington, DC: ASEE.

Hestenes, D., & Halloun, I.A. (1995). Interpreting the force concept inventory: A response to March 1995 critique by Huffman and Heller. *The Physics Teacher*, *33*, 502–506.

Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher*, *30*, 141–158.

Hwang, E.O. (2006). *Digital logic and microprocessor design with VHDL*. Toronto: Thomson.

Irwin, J.D. & Jr., D.V.K. (1995). *Introduction to electrical engineering*. Upper Saddle River, NJ: Prentice Hall.

Khoury, H.A., & Zazkis, R. (1994). On fractions and non-standard representations: Preservice teachers' concepts. *Educational Studies in Mathematics*, *27*, 191–204.

Kvale, S. (1996). *Interviews: An introduction to qualitative research inquiry*. Thousand Oaks, CA: Sage.

Mano, M.M., & Kime, C. (2008). *Logic and computer design fundamentals* (4th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Marcovitz, A.B. (2008). *Introduction to logic and computer design*. New York, NY: McGraw Hill Higher Education.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., ... Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *ITiCSE-Working Group Reports* (WGR) (pp. 125–180). New York, NY: ACM.

Merenluoto, K., & Lehtinen, E. (2004). Number concept and conceptual change: Outlines for new teaching strategies. *Learning and Instruction*, *14*, 519–534.

Mestre, J.P. (2005). Facts and myths about pedagogies of engagement in science learning. *Peer Review*, *7*, 24–27.

Miles, M.B., & Huberman, M. (1984). *Qualitative data analysis: A sourcebook of new methods*. Thousand Oaks, CA: Sage Publications.

Moloney, K., & Stacey, K. (1997). Changes with age in students' conceptions of decimal notation. *Mathematics Education Research Journal*, *9*(1), 25–38.

Moskal, B.M., & Magone, M.E. (2000). Making sense of what students know: Examining the referents, relationships, and modes students displayed in response to a decimal task. *Educational Studies in Mathematics*, *43*, 313–335.

Ni, Y., & Zhou, Y.D. (2005). Teaching and learning fraction and rational numbers: the origins and implications the whole number bias. *Educational Psychologist*, *40*(1), 27–52.

O'Connor, M.C. (2001). ''Can any fraction be turned into a decimal?'' A case study of a mathematical group discussion. *Educational Studies in Mathematics*, *46*, 143–185.

Patt, Y.N., & Patel, S.J. (2003). *Introduction to computing systems: From bits and gates to C and beyond*. New York, NY: McGraw Hill.

Sackur-Grisvard, C., & Leonard, F. (1985). Intermediate cognitive organization in the process of learning a mathematical concept: The order of positive decimal numbers. *Cognition and Instruction*, *2*, 157–174.

Stafylidou, S., & Vosniadou, S. (2004). Students' understanding of the numerical value of fractions: A conceptual change approach. *Learning and Instruction*, *14*, 503–518.

Steinle, V. (2004). *Changes with age in students misconception of decimal numbers*. Melbourne, Australia: Department of Science and Mathematics Education, University of Melbourne. Retrieved from http://eprints.unimelb.edu.au/archive/00001531/

Steinle, V., & Stacey, K. (1998). The incidence of misconceptions of decimal notation amongst students in Grades 5 to 10. In C. Kanes, M. Goos, & E. Warren (Eds), *Teaching mathematics in new times. Proceedings of the 21st annual conference of the Mathematics Education Research Group of Australasia* (Vol. 2, pp. 548–555). Brisbane: MERGA.

Strauss, A., & Corbin, J. (1998). *Basics of qualitative research*. Thousand Oaks, CA: Sage.

Vahid, F. (2006). *Digital design*. Hoboken, NJ: John Wiley & Sons.

Vamvakoussi, X., & Vosniadou, S. (2007). *How many numbers are there in a rational numbers interval? Constraints, synthetic models and the effect of the number line*. In S. Vosniadou, A. Baltas, & X. Vamvakoussi (Eds.), *Reframing the conceptual change approach in learning and instruction* (pp. 265–282). Oxford: Elsevier.

Wakerly, J.F. (2006). *Digital design: Principles and practices*. Upper Saddle River, NJ: Pearson Prentice Hall.

Wollman, W. (1983). Determining the sources of error in a translation from sentence to equation. *Journal for Research in Mathematics Education*, *14*, 169–181.