

Hardware-Software Interaction: Preliminary Observations

Neil Steiner and Peter Athanas
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
neil.steiner@vt.edu, athanas@vt.edu

Abstract

As computational devices continue to advance, there are reasons to examine their foundations a little more deeply, and to ask whether there may not be something more to be found. The fundamental manner in which hardware and software interact is poorly understood, and yet there is little indication in the literature that this is being discussed or explored. In spite of our technological achievements, we are at a loss to precisely define the boundaries between hardware and software, and to describe the nature of their interface. This paper aims to raise some of the major issues and questions, to propose a hardware-information duality, and to suggest directions in which further research might be pursued.

1. Introduction

We suppose information to be non-physical, and software to be a form of persistent information. We further suppose the interface between hardware and software to be non-passive, but to instead exhibit something analogous to forces, though these forces are presumably not physical forces, since they span the boundary between physical hardware and non-physical software. A better understanding of this interface would recast our view of computation, shed light on unforeseen computational properties and resources, and begin to bridge the gaps between physics, information theory, and the theory of computation.

The desire to facilitate calculations with the help of physical devices dates back thousands of years to the introduction of the abacus and its precursors, and extends to more recent inventions such as Napier's rods and Oughtred's slide rule. These *computational aids* eventually made way for complex but rudimentary *computing machines* such as Schickard's calculating clock, and Pascal's and Leibnitz's mechanical calculators, which did not simply assist humans in performing computations, but actually per-

formed the computations themselves. The computing machines soon made way for *programmable machines*, starting with Jacquard's loom, which could be configured with wooden punch cards, which in turn significantly influenced Babbage's Analytical Machine, arguably the first general purpose computer.

While the early computing machines relied only on physical hardware to perform their function, the programmable machines placed hardware under the control of patterns or instructions. The notion of instruction-based control is very familiar to us, and with the benefit of Turing's concept of *algorithms*, we know that problems of almost any complexity can be broken down into sequences of supported instructions. But some machines are controlled by data rather than explicit instructions, as was the case with Jacquard's loom, so it may not always be meaningful to describe the operation of a programmable machine in terms of instructions alone. Instead one may choose to think of the operation of the loom as being controlled by a single extended and implicit instruction, acting on the provided pattern data.

The modern notion of *software* can encompass both instructions and data, so one might reasonably choose to view computations as being the result of interaction between hardware and software. While hardware is commonly accepted as being entirely physical in nature, the fundamental nature of software is not as clear. That fact will be brought more clearly into focus with an attempt to describe the boundary between hardware and software in a simple circuit. The ambiguities that we encounter in describing something so basic may serve to underscore that there is still much that we do not know about the field of computation, and a number of compounding questions arise from there.

The fact that software and its boundary with hardware are not fully understood, leads one to wonder whether there may be unexpected or unforeseen behavior or properties to be found, particularly at the interface between hardware and software. The admirable success of the computer industry may have led to the belief that all of the foundations are known, and that may have stifled further exploration in

some of these areas. But as long as the underpinnings of computation remain in question, exploration should go on.

This paper aims to raise questions about the nature of the interface and interactions between hardware and software, or more properly between hardware and information, as will be clarified further on. The authors find reason to suspect that there may be novel behaviors and properties that can be identified and eventually put to use. However, this effort is still in a very forward-looking and speculative stage, and should be seen more as a preliminary inquiry into the configurable systems of the future, than as something that promises near-term commercial or engineering benefits. The present aim is to dig very deeply, in the hopes of uncovering a foundational substrate upon which future systems may be built. The reader is invited to participate in this search, because the foundations of our field are of common importance to all of us.

The organization of this paper loosely follows the chronology of the research. Section 2 studies a simple circuit, in order to develop some of the questions raised here, and Section 3 more clearly states the proposed hardware-information duality. Sections 4 and 5 delve still deeper into abstraction, with discussions of information, structure, randomness, boundaries, topology, and symmetries. Section 6 concludes by summarizing outstanding questions and future research directions.

2. Analysis

Because the interaction between any two entities typically involves their boundaries, it is important to be able to resolve those boundaries. A very familiar circuit is enough to illustrate the difficulty that we find in resolving the boundaries between hardware and software. Figure 1 shows a two-input look-up table, consisting of two inputs, one output, a few logic gates, a few memory cells, and some wires to tie everything together. This type of circuit is commonly used in configurable logic, and can generate an arbitrary function of two inputs, given the proper configuration settings. The basic operation decodes the inputs in order to address one of the four memory cells, and then propagates the selected bit to the output. We note in passing that this is a basic computational circuit whose memory contents might just as readily be used as data or as instructions.

So what constitutes the hardware and what constitutes the software in this circuit? The straightforward answer is that the memory cells, highlighted in Figure 2, constitute the software, and everything else, highlighted in Figure 3, constitutes the hardware.

But that isn't really correct, is it? Although we did not discuss the implementation of the memory cells, they must necessarily be implemented as flip-flops or DRAM or toggle switches or some other suitable form of storage, and

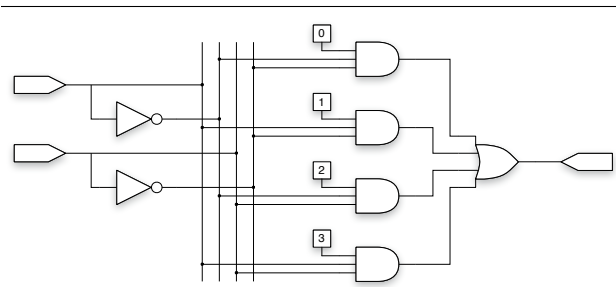


Figure 1. 2-input look-up table

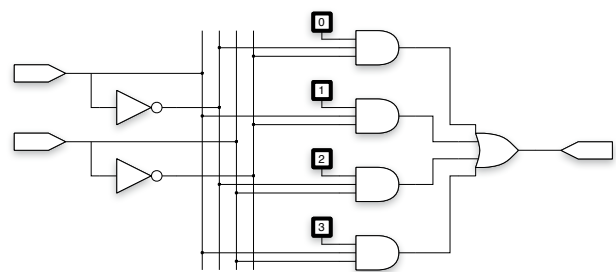


Figure 2. 2-input look-up table (software)

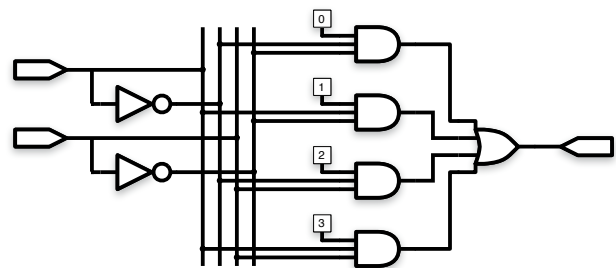


Figure 3. 2-input look-up table (hardware)

they are therefore part of the circuit's physical hardware. The software settings contained in those cells, however, are not intrinsically part of the hardware, and so we are led to view the memory bits as being software logically overlaid on top of the hardware. By convention, we will use the term *storage* to denote the implementation of the memory cells, and the term *software* to denote the settings or values stored in those cells, where the software may consist of data or instructions without loss of generality.

The software then exists in the memory cells, or at least in conjunction with the memory cells, but we are still no closer to pinpointing its location. Does it cease to be software as soon as it leaves a memory cell and enters a wire?

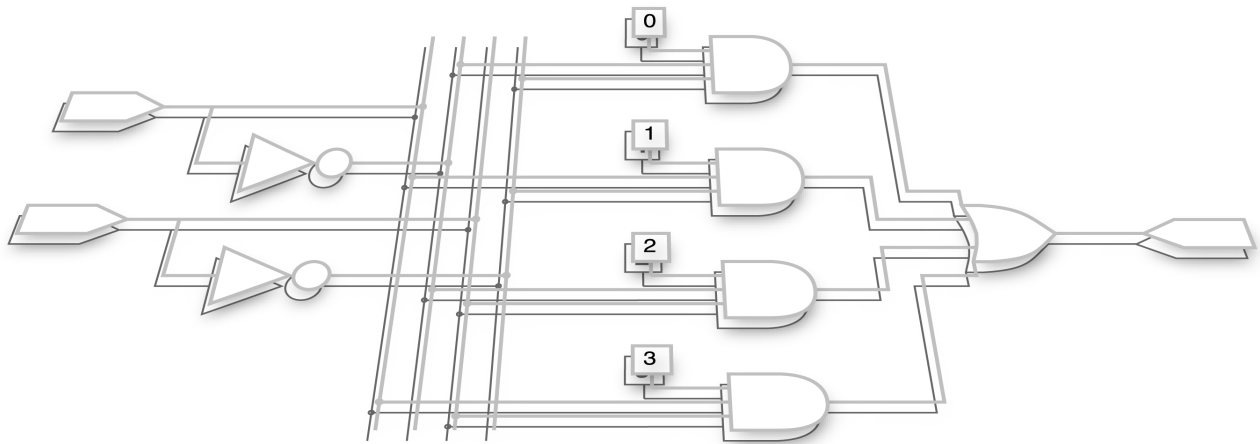


Figure 4. 2-input look-up table (hardware and information overlaid)

How would we justify that, considering that the wire carries the same logical value as the memory cell, or that we could route the value to another memory cell and again have “software”? It seems as though we may have to imagine a signal overlaid on top of the physical wire. We may not be able to call that signal “software,” but it does clearly have informational content related to the software.

At this point we have software and a signal, both having the same logical value, and both very difficult to pinpoint in physical coordinates. From a more general perspective, we may view both the software and the signal as forms of information. So then we have information in the memory cells and information in the wires. Does the information disappear when it enters a logic gate? Not really, because the gate simply performs an operation on the information. The information leaving the gate may be different from the information entering it, but there is a direct relationship between the two, and therefore we may have to treat the operation as a form of information overlaid on top of the logic gate.

We are left with the two parallel structures of hardware and information in relationship to one another. Figure 4 provides a more appropriate view of hardware and information in that it depicts them as paired components.

Although this example began by asking questions about hardware and software, it quickly led to a discussion of hardware and information. We now postulate what we alleged earlier: That software is simply a form of information.

3. Duality

The close correspondence between the hardware components and the information components suggests a dual-

ity, but before delving into that, we first define a handful of terms and the manner in which they will be used.

3.1. Hardware components

It is easiest to begin by discussing hardware, since *everyone* agrees that hardware is purely physical. Some readers may counter that configurable devices make it possible to swap hardware in or out of a device [12], but it is generally understood that the underlying physical hardware is not changed when such swapping takes place, so it seems more sensible to speak of swapping *functionality* or *design* in or out of a device. It may be best to refer to virtual hardware as *virtual computation*, or perhaps *virtual circuitry* in Brebner’s terminology [4], in order to avoid confusing the physical with the non-physical.

Hardware will be broken down into three categories: Storage, connection, and logic. The reader is advised that the hardware is not necessarily assumed to constitute a digital computer. It could just as easily be an analog or biological or quantum computer for the purposes of this discussion.

Storage: The term *storage* will denote anything that can physically retain information. The storage might be implemented through a flip-flop, a capacitor, a magnetic polarization on a hard disk, an electron spin, a bump or pit on an optical disc, and so on. The storage basically holds a value that can be read, whether destructively or non-destructively.

Connection: The term *connection* will denote a physical information carrier. The connection might be implemented as a wire, as an optical fiber, as a radio wave, as entangled photons [3], or as any number of other

forms. A connection is basically anything that allows information to propagate from one spatial location to another.

Logic: The term *logic* will denote anything that processes information in a physical manner. *We note that the term logic is not very satisfying because it is typically only used in the context of digital computers. Perhaps the term function would be more general, but that term doesn't necessarily suggest something physical. Until a better term is found, this paper will use logic to indicate a physical—and not necessarily digital—information processing stage.* The logic could be implemented by a digital gate, an op amp, a quantum molecular gate [16], an optical polarizer [2], and so on. The logic essentially performs some operation on one or more inputs and generates one or more outputs.

3.2. Information components

Information will represent the dual of hardware, and will be considered to be purely non-physical. As an aside, we do allow that information may frequently be expressed or encoded in physical form, as is elegantly required by Landauer [10], but the information itself will still be considered to be non-physical. Information will be broken down into three categories that correspond to the hardware categories: Software, signal, and operation.

Software: The term *software* will denote any persistent information. Software persistence is implemented through some form of physical storage, but the software itself will be considered non-physical. For convenience, software may be further divided into instruction and data, although it is usually possible to view all data as instructions, or all instructions as data. There are times when a piece of data might be used as an instruction, and by semantic convention it would at that time be considered to *be* an instruction.

Signal: The term *signal* will denote non-persistent information, that can move from one logical point to another. The signal may be constant or time-varying. A signal in a computational machine is carried by a physical connection. The signal may be carried as an electromagnetic field or as a set of photons, but that will be understood to be a physical representation of the signal, while the signal itself will be considered to be non-physical.

Operation: The term *operation* will denote any process that can be applied to information. An operation may be implemented by physical logic, though the reader is reminded that that term may include analog or other

processes. Regardless of implementation, the operation itself will be considered non-physical (which is not quite the same as calling it *abstract*).

3.3. Hardware-information duality

With the requisite terms now defined, we propose a duality between hardware and information:

Hardware	\iff	Information
Storage	\iff	Software
Connection	\iff	Signal
Logic	\iff	Operation

The reader who is so inclined may note that hardware is itself a product of design, and therefore that hardware implicitly carries information. Instead of immediately suggesting that our duality is really between information and information, we simply note for now that the coupling between hardware and information may consist of many strands, and we defer further observation until a later time.

4. Information

Having conveniently borrowed the concept of information to make our unorthodox proposal, we must sooner or later consider precisely what information is, and this is where the situation becomes a little more complicated. The search for a useful definition of information lends the impression that everyone knows what information is, but nobody is quite sure how to define it. This is an especially surprising condition for the fields of electrical and computer engineering, which deal almost exclusively with information, and yet lack a working definition of what it *is*.

One may counter that *information theory*, sparked by Claude Shannon's seminal paper, has been an established and thriving field for over fifty years. And it is indeed difficult to overstate the importance and contributions of information theory. But information theory makes no attempt to consider the *nature* of information, or as Shannon explains it: "Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem." [13] *Italics in original.*

Although the contents of this section are addressed more fully in a separate paper the reader may find that a summary is helpful, so we briefly discuss information and structure.

4.1. Perspectives on information

Some very interesting work on the nature of information has arisen more recently from the fields of physics [10, 11, 17, 1] and philosophy [6], and to a lesser extent

semiotics and cybernetics. Although much of that work is applicable to the current discussion, it is still very much an ongoing effort: no conclusive answers have surfaced yet.

For the purposes of the present discussion, we borrow Chmielecki's basic definition of information as "any detectable difference of physical states" or simply "any detected difference" [6]. This is also reminiscent of the *distinction* in Spencer-Brown's *Laws of Form* [14]. Although Chmielecki relates information to differences in physical states, he doesn't see information itself as physical, but instead says: "It has no mass, energy, or spatial extension, it cannot be seen, touched, or smelled. Nevertheless, it is a distinct, objective reality."

4.2. Structure and randomness

Structure may be loosely defined as an ordering of parts, and as such, it constitutes relational information. The relation may be spatial, temporal, logical, or any combination of these. Structure takes building blocks and assembles or connects the blocks into some larger grouping. Examples of this include molecules assembled out of atoms, atoms assembled out of subatomic particles, serial protocols assembled out of sequences of symbols and data, algebraic groups assembled out of sets and operations, and so on. In a mathematical context, two constructs that share identical structures are said to be isomorphic, and isomorphism turns out to be a powerful analytical tool.

Structure is something that has to be designed and built, and the building process requires energy. This is in agreement with the Second Law of Thermodynamics, which states that in a closed system, the total amount of available energy may either remain constant or decrease, but may not increase. We note that structure sometimes corresponds to a lower energy state, as is true of various crystalline forms, in which case the energy of formation may come from the material itself.

The energy requirements of the Second Law of Thermodynamics dictate that it is easier to destroy than to build, which attributes a certain measure of fragility or transience to structure. In the same way, anything that has been assembled or built will tend to erode and deteriorate over time, and experience suggests that without intelligent intervention, the process will not be reversed. Surprisingly, while nature seems adept at removing order at no cost, human efforts to do so require energy unless one does not care how long the process takes [10]. Conversely, design is difficult precisely because it involves creating structure, and preferably robust structure that will function in a wide range of situations and resist deterioration for some measure of time.

The fact that structure has to be designed and built means that structure carries design, and by extension, that structure is an information carrier. Furthermore, structure depends

upon finer grained interaction or forces, because without forces nothing can hold together. For example, molecules can exist only because of the electrostatic forces that govern the interaction of atoms. Similarly, atoms can exist and hold together only because of the electrostatic and nuclear forces that govern the interaction of subatomic particles. We may generalize this by saying that structure can only exist where there are underlying forces that govern the interactions of constituent particles. Perhaps physicists are pointing us in the right direction when they state that symmetries (in the quantum mechanical formalism) give rise to forces. In the present context, we also note that symmetry naturally depends upon structure, since it consists of a replicated sub-structure. If we take this to a logical end, we may have to conclude that structure depends upon force, and that force depends upon structure. Whether or not this will hold up to scrutiny remains to be seen.

On the subject of structure and forces, we point out that behaviors that exist at certain scales do not necessarily hold at other scales. For example, life is usually thought to exist at scales ranging from $10^{-7} m$ (a nano-bacteria) to $10^2 m$ (a whale). Quantum behavior is generally thought to be significant only from particle scales to small molecular scales. In the realm of physics, we know that different forces dominate at different scales, with the strong and weak nuclear forces dominating at particle scales, the electrostatic force dominating at intermediate scales, and the gravitational force dominating at very large scales. If something analogous to physical forces governs the structure and interaction of hardware and information, we should be prepared to observe a variety of forces operating on different scales.

Randomness may be seen as the absence of structure (according to Ceperley, as quoted in [7]), and it is interesting to note that true randomness is usually difficult to find. It may not be very difficult to identify something that appears to be random, but we often have to work very hard to rigorously prove its randomness. When seeking true randomness, people generally turn to sources that are unpredictable, such as radioactive decay or atmospheric noise. But while the issue may be philosophical, there is arguably a difference between a source in which we can discern no structure, and one that truly has no structure. Is radioactive decay truly random? In a statistical sense it certainly is, because we lack the ability to predict when the next decay will occur. But if there are deterministic processes underlying particle behavior, as suggested by 't Hooft [15] and Fredkin [8], then what appears random to us may still have structure.

It is interesting to note that mathematical complexity increases with true randomness, which means that for a given number of building blocks, a structured arrangement of the blocks will obey certain rules and will therefore be express-

ible in a more compact form than a “random arrangement.” Another way to state this is to say that structure is generally compressible, while randomness is not, because structure can be decomposed into relationships [5]. Conversely, although structure is fragile, randomness is not, in that we don’t risk creating structure by perturbing randomness.

5. Boundaries

At any interface we necessarily have boundaries, and in some cases those boundaries can tell us a lot. Consider electromagnetics, where we rely heavily on boundary conditions to solve problems. Or consider differential equations, where either boundary conditions or initial conditions allow us to take a general relationship and apply it to a specific problem.

What then do we know about boundaries at the hardware-information interface? Are the boundaries rigid? Fixed? Flexible? Can we bend them to alter their behavior? Are there continuity properties or surface-normal properties? Do reflections occur at the boundaries when there are mismatches, as happens with electromagnetic waves? Are there forces that define or affect the boundaries?

Although very little is understood about the hardware-information interface or its boundaries, there are some related areas that may help bring us a little closer to the issues of interest. For now we have to build on what little foundation is available, and we begin with topology and distinctions.

5.1. Topology and symmetries

Matzke suggests that information laws might be understood as topological constraints [11]. This is an interesting insight because in a certain sense hardware and software depend upon each other’s structure or topology in order to function.

We understand that software cannot function on its own and requires an underlying substrate, but that substrate need not necessarily be hardware. Emulators are frequently used in place of real hardware, and as long as the expected underlying structure is provided, the software can function exactly as it would on actual hardware. The same could be said for software running on a soft core inside an FPGA. This suggests that hardware-software boundaries might be usefully analyzed in terms of their topologies. (The term software was used here in place of information, because we do not typically think of information being able to *function*, but in fact the topological perspective would apply to information in general and not just software.)

Even on a very simplistic level, it is easy to see that hardware and software must be structurally or topologically

matched with each other, and that any mismatches can be expected to result in unexpected behavior or crashes. From that viewpoint, part of the difficulty of writing robust software might be attributed to the difficulty of matching the software to the structure of its underlying substrate, which is usually a complexity issue, and sometimes a completeness issue. Or returning to the quantum mechanical assertion that symmetries give rise to forces, one might think of a computer as a way of molding the information to the hardware, so that the topologies match and the symmetries are enforced.

Structure has already been presented as relational information, and thus something non-physical, but structure clearly also pervades the physical world. The structure itself may thus be thought of as information, whether it pertains to physical or non-physical components. In some ways this is in agreement with the underlying assumption that all information stems directly or indirectly from design.

5.2. Hardware-software distinctions

Even though hardware and software are intended to work together, there are obvious differences between them. (It is easier to think of *software* working with hardware, because we do not normally think of *information* working or acting in any sense at all. But what is said here about software should be generally applied to information as well.) Hardware is visible and tangible. Software is invisible and is generally understood to be intangible, even when it is provided on tangible media. However, the effects of software on hardware are clearly visible. It seems intuitively correct to say that the hardware is already capable of anything that the software might cause it to do. But wouldn’t it be interesting if this were not true? That is, if the interaction of hardware and software yielded more than the sum of the parts?

Conventional thinking tends to approach hardware and software as completely separate topics. To be sure, hardware development is not done in the same way as software development, and a very different mind-set is usually necessary for the one versus the other. And yet the two are intricately related. The situation brings to mind a quote by Einstein regarding electrons and fields:

“I feel that it is a delusion to think of the electrons and the fields as two physically different, independent entities. Since neither can exist without the other, there is only *one* reality to be described, which happens to have two different aspects; and the theory ought to recognize this from the start instead of doing things twice.” *Quoted in [9]. Italics in original.*

At least one difference between what Einstein was considering and the case of hardware and software, is that there

are no known laws for hardware and software corresponding to the physical laws that keep the electron and its field in perpetual relation to each other. Hardware and software are easily mismatched or thrown out of sync, and solid design takes a great deal of understanding of both components, followed by substantial testing to avoid functional errors, all of which may begin to explain the current emphasis on co-design. However, co-design and the concepts proposed here are mutually independent. Co-design addresses a very important complexity issue, but does not give much additional insight into the fundamental issues that the present effort seeks to investigate.

Returning to the electron/field analogy, the underlying laws seem to apply primarily at very small scales, which is to say that the structure is defined at the particle level or below. Nature does not seem to build physical laws above the atomic scale: the interactions and behaviors become more complex, but no additional laws are introduced. Do hardware and software benefit from a similarly fine-grained basis? Large hardware-software systems require constraints up to the very highest levels, and it is perhaps appropriate to ask whether those constraints really belong to the underlying behavior, or whether they are simply structural arrangements based on the more fundamental behavior.

6. Conclusions

Our present understanding of software and its boundaries and interaction with hardware appears to be very incomplete, and there is little evidence that anybody has considered or is considering these issues. We suggest that these questions are important for the fields of electrical and computer engineering, and furthermore that there is reason to expect new and unforeseen interactions and properties at the hardware-information interface.

A cursory and high-level analysis of a simple circuit is sufficient to reveal sizeable gaps in our understanding, and to show that while we may be accustomed to dealing with hardware and software, it is sometimes necessary to tackle the very elusive information question as well. But the topic of information also enhances our insights about structure and topology as they relate to hardware and software.

Although the long-term motivation of this effort is to gain a comprehensive understanding of hardware-software interaction, many of the puzzle pieces still defy placement, and we find ourselves working from known edges in the hope of eventually reaching the area of interest. Many questions remain outstanding:

- How do we define hardware-information boundaries?
- What properties exist at those boundaries?
- What “forces” exist at those boundaries?
- Can we employ those forces in computational devices?

- What “energy states” exist at those boundaries?
- Do the energy states favor certain structural arrangements?
- What is the fundamental nature of information?
- What is the fundamental nature of software?
- How do hardware and information interact?
- Can we formulate more comprehensive hardware-software models?

This list is clearly not exhaustive, but it is presented in the hope of sparking interest, and generating discussion and further research into these matters. With answers to these questions, we might hope to gain deeper understanding and insight into the nature of computation, and though the initial outcome might only be a renewed perspective, the longer-term benefits might extend to engineering goals and system design. The argument is that by broadening our foundation, we may find new ways and perhaps better ways to do computation. Since reconfigurable computing is so deeply rooted in both hardware and software, if there are indeed unforeseen properties at the hardware-software interface, then reconfigurable computing, more than any other discipline, stands to benefit from them. The resulting systems might prove to be more fluid and more efficient, giving way to more responsive and life-like behavior.

As may be expected at the beginning of any new exploration, the current work is strong on speculation and weak on substance. If there is indeed substance to be found, then time will begin to redress that situation. Additional information, including a conceptual animation, is available at <http://www.ccm.ece.vt.edu/hsi>.

7. Acknowledgments

This work is made possible through a Bradley Fellowship given by the Bradley Department of Electrical and Computer Engineering.

References

- [1] J. D. Bekenstein. Black holes and information theory. *Journal of Contemporary Physics*, 45:31–43, 2003.
- [2] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin. Experimental quantum cryptography. *Journal of Cryptography*, 5(1):3–28, 1992.
- [3] D. Bouwmeester, J.-W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger. Experimental quantum teleportation. *Nature*, 390:575–579, 1997.
- [4] G. Brebner. Circlets: Circuits as applets. Napa, California, April 1998. Proceedings of the 6th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 1998.

- [5] G. J. Chaitin. In *Information, Randomness & Incompleteness*. World Scientific Publishing Co. Pte. Ltd., Singapore, 1987.
- [6] A. Chmielecki. What is information? Boston, Massachusetts, August 1998. Proceedings of the Twentieth World Congress of Philosophy.
<http://www.bu.edu/wcp/Papers/Cogn/CognChmi.htm>.
- [7] L. Foley. Analysis of an on-line random number generator. Project report, The Distributed Systems Group, Computer Science Department, Trinity College Dublin, April 2001.
- [8] E. Fredkin. Digital mechanics. an informational process based on reversible universal cellular automata. *Physica D*, 45(1–3):254–270, September 1990.
- [9] E. T. Jaynes. Probability in quantum theory. In W. H. Zurek, editor, *Complexity, Entropy and the Physics of Information*, pages 381–403. Addison Wesley Publishing Company, Redwood City, California, 1990. *Proceedings of the 1988 Workshop on Complexity, Entropy, and the Physics of Information held May–June, 1989, in Santa Fe, New Mexico*.
- [10] R. Landauer. Information is physical. *Physics Today*, 44:23–29, May 1991.
- [11] D. J. Matzke. Information is protophysical. In *Proceedings of the Workshop on Physics and Computation: PhysComp'96*, pages 223–225, New England Complex Systems Institute, Boston, MA, November 1996.
- [12] C. Plessl and M. Platzner. Virtualization of hardware — introduction and survey. Las Vegas, Nevada, June 2004. Proceedings of the 2004 International Conference on Engineering of Reconfigurable Systems and Algorithms, ERSA 2004.
- [13] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.
- [14] G. Spencer Brown. *The Laws of Form*. The Julian Press, Inc., New York, 1972.
- [15] G. 't Hooft. Quantum gravity as a dissipative deterministic system. *Classical and Quantum Gravity*, 16:3263–3279, 1999.
- [16] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–887, 2001.
- [17] J. A. Wheeler. Information, physics, quantum: The search for links. In W. H. Zurek, editor, *Complexity, Entropy and the Physics of Information*, pages 3–28. Addison Wesley Publishing Company, Redwood City, California, 1990. *Proceedings of the 1988 Workshop on Complexity, Entropy, and the Physics of Information held May–June, 1989, in Santa Fe, New Mexico*.