MagicBlocks: A Game Kit for Exploring Digital Logic

Nawwaf Kharma, Leon Caro,
and Vivek Venkatesh*

Electrical and Computer Engineering and *Education Departments,
Concordia University,
1455 de Maisonneuve Blvd. West,
Montreal, Quebec, Canada - H3G 1M8
kharma@ece.concordia.ca

Abstract: The paper introduces MagicBlocks, a LEGO™-like game that allows learners to build increasingly more elaborate functioning digital logic circuits from well-defined logic blocks. These logic blocks represent some fundamental digital logic (and computing) concepts such as: input, repetition, arithmetic and logical operations.

Keywords: games and simulations, engineering education, digital logic.

1. Introduction

This paper presents MagicBlocks, a game kit that can be used to introduce pre-university learners to fundamental concepts of Digital Logic. It is our belief, based on theoretical arguments as well as first-hand instructional, and life experiences, that games, both virtual and embodied, hold great potential, especially in the delivery of advanced concepts and skills to a pre-university audience.

Many engineering subjects have been characterized as *theoretical*, thereby lending themselves to didactic lecture-based instruction accompanied by rigorous problem-solving exercises [1]. We do not believe that there will be a complete alternative to such an instructional methodology. However, the fact that the *amount* of information in engineering, is increasing at an exponential rate entails that all the relevant the material cannot be covered in lectures. Furthermore, the level of skill required by a practicing engineer is so high, that universities find it difficult to effectively deliver a comprehensive curriculum, in about four years of undergraduate study. Students, therefore, must be well equipped to acquire an understanding of the expanding field of engineering outside of their scholastic environments. There is also an onus on universities to produce creative thinkers among their engineering graduates. In our view, therefore, fundamental concepts of core subjects, such as computer architecture and programming, should be introduced, via appropriate educational vehicles, prior to the undergraduate level of education. While there will always be a need for intelligent interactive tuition, the role of the tutor is becoming more that of a guide who presents and mediates the

acquisition of advanced concepts and ideas. We contend that games are effective vehicles for guiding students' understanding of concepts and for providing students with the opportunity to abstract a holistic working knowledge of the subject at hand, in our case, that of digital logic.

2. Background

2.1 Example of Current Educational Practice

Digital logic is taught through lectures and labs. The lectures are presented by an instructor (usually a professor), and focus on theoretical aspects of the topics listed in the outline of the course. The instructor's main activity in these lectures is to explain, as clearly as possible certain concepts or tools, such as K-maps, and then provide one or more examples of how such concepts are used for fulfilling some objective or solving a problem, such as minimizing logic circuits. There is a measure of interaction between students and the instructor. This interaction usually takes the form of questions and comments that seek clarifications, elaborations, and additional examples. The instructor attempts to answer as many of those requests as possible, but is also expected to cover a number of pre-determined subjects in each lecture. Lectures are accompanied by laboratory-based activities (labs). In each lab the students, individually or in groups, are required to carry out certain experiments with real components and instruments, as well as design, build and test, their own simple or complex circuits. The labs allow the students not only to experience concrete realizations of the concepts presented in the lecture (and the text book), but also demand that students utilize these ideas in design projects, which require creative thought, organization, decision-making, team work, practical skills, and an appreciation of the value of testing and verification.

Assessment of students learning include pen and paper exams, pen and paper take-home assignments, practical lab assignments and projects, as well as pen and paper midterm and final exams, held under controlled conditions. These methods aim to measure (a) students' understanding of the theoretical concepts, (b) students' ability to use these concepts to solve problems, (c) students' ability to carry out circuit analysis and synthesis, individually and in groups, in a lab as well as under controlled exam conditions.

2.2 Related Work

There have been good attempts at building simulations to aid in the teaching of digital logic. These, however, are almost exclusively software simulators, mainly due to the special suitability of computer languages to the implementation of such simulators. This fact does not affect the amount of work required to design, build and test usable and stable simulators. Following are some examples.

Corsini & Rizzo [2] introduced a software package that can be used, by students with no background knowledge of professional simulation packages, to a) describe and simulate four classes of digital circuits by using a special language, as well as b) derive final, optimized Boolean equations that describe the proofed circuits. The software provides the user with a user-friendly menu-driven interface with context-sensitive help. It is our belief that this program is a good tool to introduce students quickly and effectively to the formal design process used in packages such as OrCAD.

El-Hajj & Kabalan [3] describe an innovative method to simulate digital circuits. The main innovation is the use of a spreadsheet program to implement these simulations. This can prove beneficial for schools that lack the resources to purchase professional or educational

simulators. However, their approach is rather limited because: a) it requires that the user be quite proficient in spreadsheet use, and b) it ".. does not allow [users] to study timing problems."[3].

The above examples are simulations, while MagicBlocks is a gaming environment: the functionality that is derived from one or any construction of blocks reflects actual performance of underlying hardware circuitry. Further, according to the first-hand experience of Singh [4], students who use simulations place too much confidence in the precision of the results, "not realizing that they are only as accurate as the models used in the simulator". In addition, Singh states "the greatest danger in the use of simulators is the development of 'hardware phobia' in students."

In addition, the immediate usability and the self-sufficiency of the blocks make them accessible to pre-university learners, as well as those potential learners who may not have sufficient access to a computer or the licenses necessary to run a useful software simulation package.

2.3 Theoretical Foundations of Current Practice

Current instructional practice (as briefly outlined in section 2.1) embodies a number of conceptions about learning. Some of these are:

- The **primary sources of knowledge** about the subject are: the **instructor** and the various **printed documents** (e.g. text book, printed notes, handouts). These are presented to the students as the authoritative and wholly accurate sources of all the declarative, and some of the procedural knowledge, required for the course.
- **Students activate and enhance their mental understanding** of the concepts explained by the instructor and in the book **through practice**. Practice takes the form of solving pen and paper problems and carrying out hands-on lab assignments.
- **Cooperation between students on group projects is necessary** if the relatively complex projects are to be completed in the time allotted to them. However, students are not particularly encouraged to collaborate in the discovery of knowledge or tools that enable the extension of this knowledge; their collaboration is oriented towards the completion of the project.
- Assessment is carried out in various ways; however all **answers are measured against an optimal** or (in the case of design) a preferred solution. Assessment items are designed to test a student performance against a set of explicit learning objectives designated before the start of the course.

Given the above, it is obvious that the learning theory of behaviorism has the strongest influence on the design of many digital logic courses. Its influence is apparent in the way learning objectives and assessment methods are devised. Emphasis is placed on specific measurable learning outcomes (e.g. the ability to design efficient and reliable combinational circuits that satisfy a prescribed functionality)[6].

Students are mostly assessed via individual assignments and tests. Except in some design cases, there exists a model answer (response) to each problem (stimulus), for which the learner is regularly reinforced, with good marks and a measure of social recognition. Failure to produce correct answers requires repetitive study of the same (or similar) material until the learner can display that he/she has mastered it.

Also, and in line with most instructional prescriptions of behaviorist theories, the instructor is the authoritarian centre of the learner's universe. The instructor is not only the

source of true knowledge, but also the controller of the course, and the ultimate evaluator of students' performance.

Empiricism, a fundamental epistemological underpinning of behaviorism, expresses itself in the laboratory-based exercises. Empiricism emphasizes that the physical realization (i.e. actual circuit) of a concept should behave in line with certain pre-stated requirements. Besides behaviorism, cognitive information processing (CIP) [6] theories have the second biggest influence on instructional design. For example, all scientific instructional material (e.g. books) and courses (including digital logic) are carefully organized in a manner that provides a) a clear logical order, b) a progression of ideas and exercises from simple to complex, as well as from well-grounded (in older material) to substantially new. Further, problem-solving, as opposed to just rote memorization, is the main goal of most (advanced) exercises and assignments.

3. Rationale for Development of MagicBlocks

3. 1 Problems with Current Instructional Practices

The development of MagicBlocks was necessitated after a critique of the existing manner in which digital logic is taught in many university courses. This revealed certain deficiencies. The main problems with current mainstream instructional strategies for teaching digital logic to university students are:

- **Lack of prior knowledge**: Most university students encounter digital logic topics for the first time at the threshold of their university career. They possess no foundation of prior information (i.e., mental schemas) that can be used to encode new information and create new schemas of understanding. This places an onus on the instructor to successfully relate the new digital logic concepts (e.g., the binary nature of data) to existing out-of-discipline concepts (such as black & white printing and images.) Such a situation requires that the instructor or course designer (a) seek information about his/her students' academic and professional background, as well as (b) allocate enough time, at the start of the course, so that students are able to effectively absorb/construct the seemingly simple but foundational concepts of the discipline.
- **Lack of time to discuss advanced concepts**: Due to the time needed to introduce a good amount of new ideas to the students, the constraints of time allocated to a university course, and the dependency of students on the lecturer as the main source of knowledge, students are usually unable to take more than one or perhaps two courses in digital logic. Hence, advanced topics, which are valued by industry (e.g. validation & verification) and emerging areas of R&D, which are becoming increasingly important (e.g. parallel processing)[1], are unlikely to be studied at a serious level of depth in the context of an undergraduate degree, unless the student specializes in digital/computer design or a similar area.
- **Exposure to an abstract language**: In addition to the lack of prior knowledge and the resulting lack of time for discussion of advanced topics, most instructors have no option but to teach concepts of digital logic and introduce the language of the discipline simultaneously. The learner is allowed little time, to construct his/her own understanding of concepts about the discipline, via the manipulation of, first actual objects (e.g. simple components), then of image-based and mixed representations (e.g. block diagrams), and finally, of a highly-abstract and efficient symbolic formal language (e.g. Boolean logic). It is totally natural that this is not currently done, because adopting such a Brunerian [7] discovery-learning type of

approach to instruction requires the development of new educational technology, as well as early introduction, of some basic ideas of computer architecture and programming.

3.2 Theoretical Foundations of Games

For the purposes of this paper, we will define a game as a conceptual, goal-based, learner-initiated [8], and stimulating set of situated activities [9], which, through the act of discovery and play, enables a constructivist form of learning.

Childrens' acts of play, from its freest to most structured form, have been theorized to enable learning [10-12]. The philosophies of children's' play have long informed the development of adult simulation and gaming [8, 11, 13, 14]. Researchers have contended that the act of playing a game propels learners through three critical phases: (a) experience, (b) reflection, and ultimately, (c) learning [11, 15]. Reflection, or reflective activity is a constructive process wherein learners interpret the experience of participating in the game and make sense (or meaning) of it. This is in stark contrast to learners receiving didactic instruction form a lecture-based instructional strategy. Games, in our opinion, lean heavily towards the radical constructivist paradigm [16], focusing both on reflective discussion as well as on extensive 'hands-on' participation in authentic and situated activities [9,17]

In keeping with a constructivist tradition, MagicBlocks, is developed to satisfy the following broad criteria (adapted from Corbeil [8]):

- Goal-directed activities **stimulate exploration** and evoke challenge for the learner.
- Learners are not restricted in their approach(es) to playing the game and hence, can apply their **individual learning styles.**
- Activities are **learner-centered**, initiated and performed by learners, and learners set their own intermediary goals to help attain the ultimate goal of the activity.
- The instructor plays a **guiding and mediating role**: modeling, and scaffolding instruction [18, 19].

MagicBlocks, the gaming environment described in this paper will help alleviate some of the problems identified in the instruction of digital logic in the following ways:

- MagicBlocks is innovative and challenging, and hence will spike the interest of the learner. This will affect the intrinsic **motivational** traits positively [20] and increase the probability that students will learn how digital blocks function.
- MagicBlocks introduces abstract concepts to the students in an authentic setting [9,17] thereby promoting the chances that the knowledge system developed will be **relevant and transferable** to activities outside of the gaming environment [21].
- The Brunerian discovery learning approach [7] is adopted in the use of MagicBlocks as a learning tool. Discovery learning will help learners **construct mental models** of digital design through their active participation in creating digital blocks. MagicBlocks allows learners to observe the necessary internal workings of digital design. Learners are also required to engage with digital blocks, and abstract a symbolic understanding of the manner in which these digital blocks function. MagicBlocks supports the construction of meaning through the actual manipulation of objects to the creation of a symbolic mental model representing the digital design created.

4. MagicBlocks: Description & Use

MagicBlocks consists of a) Building Board, on which blocks would be placed, b) Logic Blocks, and c) User Manual.

4.1 The *Board*

The board is large in size, and provides both the physical and electrical foundation for the logic blocks**,** which are placed on it. The blocks fit in any location on the board similar to the way a LEGO block fits in place. However, once a block is placed on the board, it is automatically grounded and provided with power. The blocks that may run synchronously are also provided with a universal clock pulse from the board. The board is, otherwise, a passive component.

4.2 The *Blocks*

The logic blocks are seven in number. They are:

- An **Input Block** (fig. 1): this block is **a** user configurable generic package of input lines**.** It has 8 input lines. The input lines have no input pins, but provide only driven 1/0 output pins. The user, via a set of 8 small switches decides the logic levels of the individual pins. Each of the output pins is able to drive the input pin of any other block in the kit.

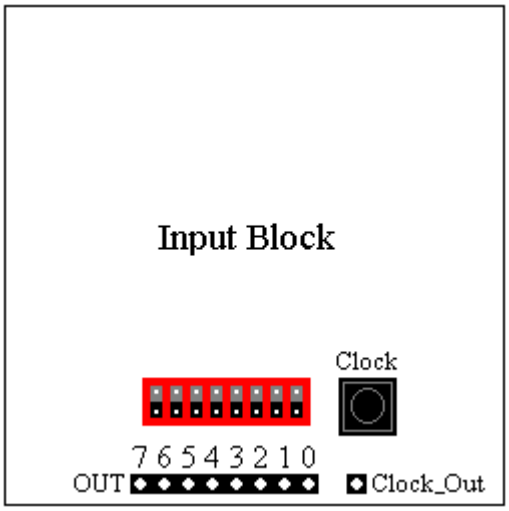

Figure 1: Input Block

- A **Monitor (or output) Block** (fig. 2): this block displays logical value of the input signal in binary, and in either Decimal, Hex, or ASCII. The input of each bit is connected directly to the corresponding output bit, as well as to a corresponding LED and to a pin on the chip. This gives the user the ability to use the monitor block, as either a terminal output block, or as an intermediate monitor, which sits in between two other blocks, expressing visually the nature of signals passing between them. Each of the 8 LEDs connected to the input signal indicate the logic level of the bit it is connected to. Together they show the value of the input signal in binary. Green indicates that the logic level is 1, while red indicates logic level 0. The monitor block also displays the value of the input signal in either Decimal, Hex, or ASCII. The user can toggle between these modes by repeatedly pressing and releasing the button. Each time the button is pressed, the mode will change from Decimal to Hex to ASCII, then back to

Decimal. The current mode will be displayed on an 8-segment display as “d, H, or A”. The current input signal value can be observed on a 16-segment display and two 8-segment displays. Another LED is only used in ASCII mode. Green indicates that the current ASCII character is upper case (otherwise it is lower case), while red indicates that it is not a keyboard symbol, and thus Ctrl must be depressed at the same time as the character that is displayed, to represent this character.

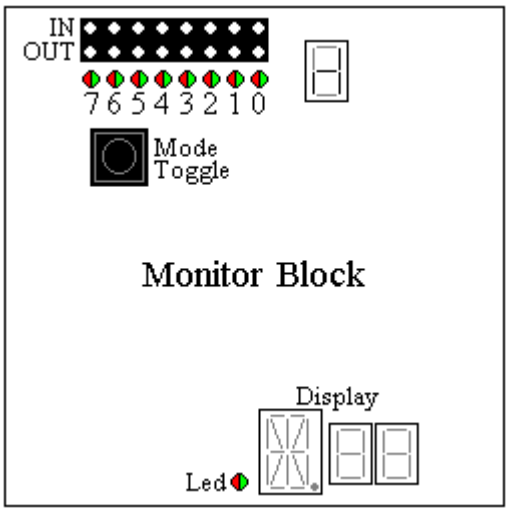

Figure 2: Monitor Block

- A **Logic Function Block** (fig. 3): this block accepts two 4-bit inputs, and produces a single 1-bit output. It can either perform a sum of minterms or a product of maxterms. A switch is used to select the mode. Logic 0 produces a sum of minterms while logic 1 produces a product of maxterms. The current mode is displayed on a 16-segment display, using the AND symbol (^) for the sum of minterms or the OR symbol (v) for the product of maxterms. Green-red LEDs are connected to each input and output bit so that the current input and output signal values can be visually observed.

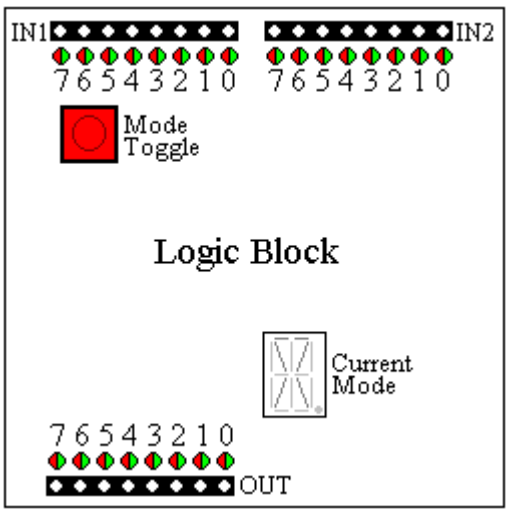

Figure 3: Logic Function Block

- An **Arithmetic Block** (fig. 4)**:** this block is basically a configurable simple calculator, with two sets of 4-bit input lines, and one set of 8-bit output lines. The user can toggle between the modes by pressing and releasing the button. Each time the button is pressed, the mode changes from addition to subtraction to multiplication to division, then back to addition. The current mode will be displayed on a 16-segment display as "+, -, *, or /". In addition mode, a 1-bit carry input is provided in case the user wishes to use 2 or more arithmetic blocks to add two 8-bit (or higher) numbers. If no signal is connected to the carry input, it will be assigned logic 0 (no carry). In subtraction mode, the absolute value of the result will be outputted, and a LED is used to indicate the sign of the result. Green indicates that the result is negative; otherwise, the LED will be off. In division mode, the 4 most significant bits of the result represent the integer part, while the 4 least significant bits represent the fractional part. The LED is also used to tell the user whether the result is an integer, contains a fraction, or a divide by 0 operation was attempted. If the LED is green, this indicates that the result contains a fraction (the 4 least significant bits are not 0), while if the LED is off, this indicates that the result is an integer. If a divide by 0 operation is attempted, the LED will be Red and all the output bits will be set to 1. Green-red LEDs are also connected to each input and output bit so that the current input and output signal values can be visually observed.

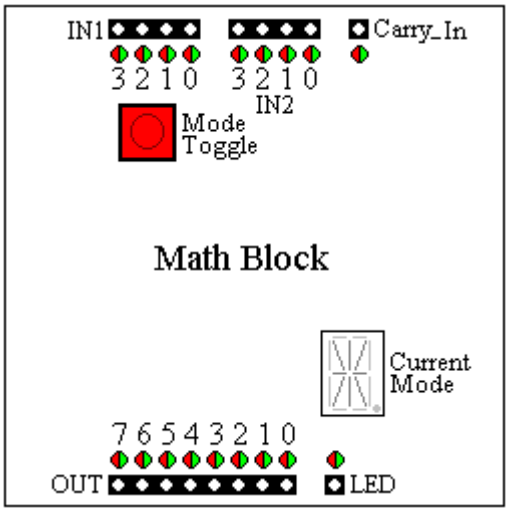

Figure 4: Arithmetic Block

- A **Counter Block** (fig. 5): this block is an 8-bit up/down counter that counts from 0 to 255 then rolls over back to 0. Each time a clock pulse is received, the counter is either incremented or decremented by one, depending on the value of the up/down switch. By setting a second switch, the user can use either the system clock (provided by the board), or an external clock. The counter block also gives the user the option to load an 8-bit number into the counter (from the 8 input lines), instead of incrementing/ decrementing it. A third switch controls whether the counter will be incremented/ decremented or loaded when a clock pulse is received. The current value of the counter is provided on 8 output pins. An asynchronous reset switch is also provided, which sets the value of the counter to 0 when it is set to 1. Green-red LEDs are connected to each input and output bit so that the current input and output signal values can be visually observed.

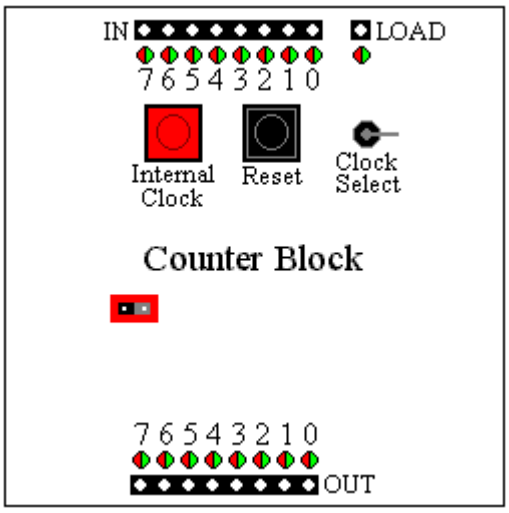

Figure 5: Counter Block

- A **Memory Block** (fig. 6): this block is basically a simple RAM. It can save up to eight 8-bit numbers (in 8-bit registers). It has 8 data inputs, 3 address inputs, and 8 outputs. The address lines are used to select which register (0 to 7) is selected. If no signal is connected to the address lines, then register 0 is selected. The number of the currently selected register is shown on an 8-segment display. The 8-bit number stored in the selected register can always be read from the output pins. A new number can also be stored in the selected register, by setting the read/write switch to 1. As soon as the next clock pulse is received, the old value in the selected register will be replaced with the value on the 8 input lines. By setting a switch, the user can use either the system clock (provided by the board), or an external clock. Green-red LEDs are connected to each input and output bit so that the current input and output signal values can be visually observed.

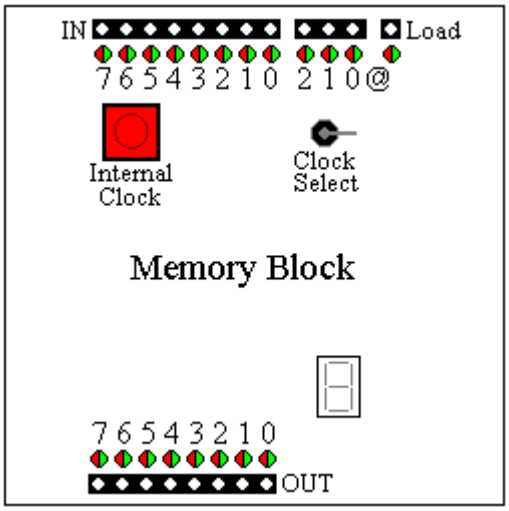

Figure 6: Memory Block

- A **Capsule (or cover) Block**: this is simply a resizable cover that allows the user to place a number of interconnected blocks of type 1-5 within a single package. Once a number of blocks are encapsulated, all that one sees is a blank block with a) input and output pins, and b) a set of (user-authored) notes on its top specifying: the overall functionality of the block, and the names of the input & output pins. The capsule block's input and output pins can be connected to any number of input and output pins of the encapsulated blocks (inside). The cover block essentially implements information hiding. This allows the user to break the whole project into a number of packages, each of which encompassing its own (potentially elaborate) set of interconnected blocks. The capsule block has 16 input and 16 output pins.

4.3 The Learner's Manual

The learner's manual contains two step-by-step sequences that explain in detail, using figures and text, how a total novice can use MagicBlocks to construct interesting digital circuits. The board, as well as all of the blocks, is required for the realization of the two examples. The board and each of the blocks are introduced upon first time. This provides a situated explanation of what the blocks do (as opposed to a detached abstract one).

In addition the learner's manual contains one solved and two unsolved challenges, covering a wide range digital circuit subjects. The subjects covered include subjects such as a) NAND, NOR, and XOR circuits; b) code conversion; and c) multipliers.

4.4 Examples of Use

Following are two examples of varying difficulty of digital logic projects built with only blocks from MagicBlocks. The purpose of these examples is to provide concrete evidence that: a) MagicBlocks can be used to build digital circuits without the need for any additional components (e.g. buffers or resistors), b) Digital circuits can built that are both relevant to digital logic, as well as meaningful to the young learner, and c) that the number of blocks required is not so larger as to render the whole idea practically infeasible.

- **Project #1** (Figure 7): Differentiates between **odd and even numbers**, by looking at the least significant bit of the counter block (Out0). The user can use either the system clock or the internal clock on the counter block.
- **Project #2** (Figure 8): Finds all **multiples of a 4-bit number** (entered using bits 7-4 of the input block). At each clock pulse, the counter will be incremented; it will roll over to zero after it reaches 15. Storing the most recent multiple in memory is optional. (If the storage of the most recent multiple is not desired, simply omit the memory block in the scenario).

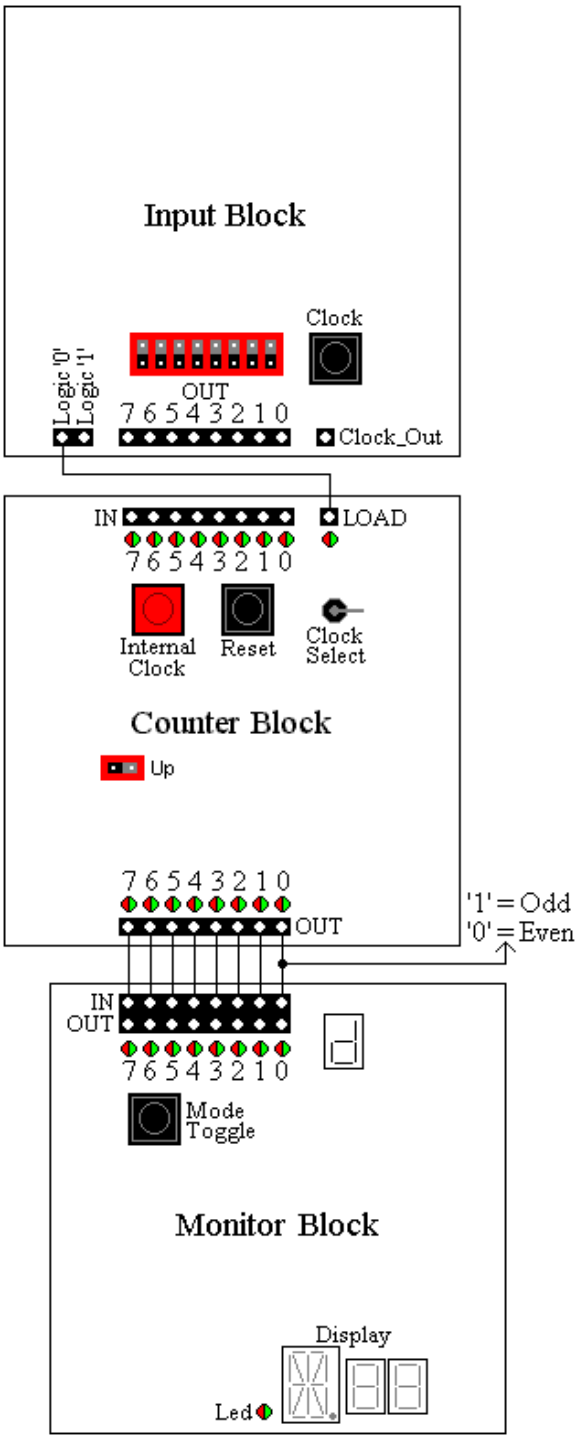

Figure7: Odd-Even Number Classifier

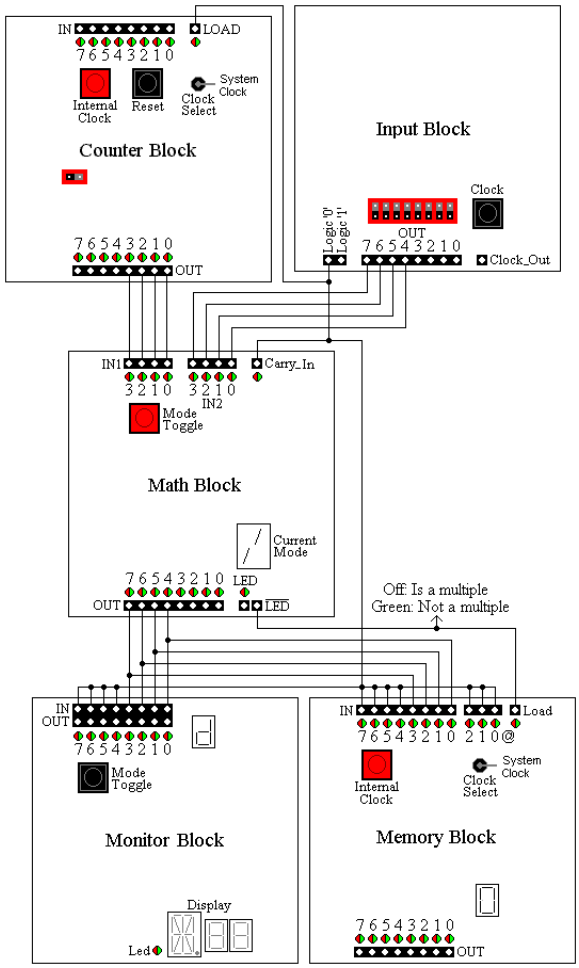

Figure 8: Number-Multiples Calculator

4.4 Educational Settings

MagicBlocks can be incorporated into a variety of educational settings, from instructor-centred to student-based. MagicBlocks can be used as a complement to an instructor-led lecture to demonstrate relevant digital circuits. Short, laboratory-based activities can incorporate the use of MagicBlocks. Project-based activities, which can last as long as a semester, can realize their potential with the use of a MagicBlocks digital design activity. Finally, students can play with MagicBlocks on their own time, and at their own pace, either at their homes or as a school-based extra-curricular activity.

5. Summary & Conclusions

First, the current mainstream methodology for teaching digital logic is outlined. Then, a quick survey of a number of related simulations is presented. None of them is hardware-based, though all have their educational uses. The advantages that MagicBlocks offers over its most closely related commercial kits were presented. This was followed by an outline of the (implicit) theoretical foundations of the mainstream's approach to the teaching of digital logic. What the authors see as the major deficiencies in current teaching practice is discussed. And this, in turn, forms the core of the rationale for the development of MagicBlocks. A somewhat detailed

description of the various components of MagicBlocks is presented next. This description covers: a) the board, b) the blocks, and c) the manuals. A couple of example projects of varying difficulty, all utilizing the blocks, are presented. Finally, examples of possible educational settings are put forward.

The most valuable contribution of this research (and development) effort is the presentation of a complete working example of a physical block (or embodied object) based approach to the learning/teaching of digital logic design. The other main alternatives to the approach we are proposing here are: software simulation and traditional logic labs. The first approach, despite its many advantages, suffers from the curse of disembodiment, and the other, despite its total realism, offers the learner too many unnecessary distractions (e.g. making the right pin connections). What we aim at is providing an embodied object-oriented approach to digital/hardware design, in the hope that it will remedy some of the deficiencies associated with simulations and traditional hardware labs, while simultaneously spiking the learners interest and hence motivation. It is hoped that, among other advantages, this will allow educators to introduce some of the most fundamental concepts of digital logic, early in the educational curricula and hence allow university instructors to focus more on teaching advanced topics, and supervising even more realistic lab-based design projects.

Bibliography

[1] C. Baillie and Fitzgerald, G., "Motivation and Attrition in Engineering Students", *European Journal of Engineering Education*, vol. 25, no. 2, pp. 145-155, June 2000.

[2] P. Corsini and L. Rizzo, "SSCSSC: A Tool for the Teaching of Digital Circuits", *IEEE Transactions on Education*, vol. 34, no. 1, pp. 70-75, February 1991.

[3] A. El-Hajj and K. Y. Kabalan, "A Spreadsheet Simulation of Logic Networks", *IEEE Transactions on Education*, vol. 34, no. 1, pp. 43-46, February 1991.

[4] M. Singh, "Role of Circuit and Logic Simulators in EE Curriculum", *IEEE Transactions on Education*, vol. 32, no. 3, pp. 411-413, August 1989.

[5] Lego Website: http://mindstorms.lego.com/

[6] M. P. Driscoll, *Psychology of Learning for Instruction*, 1st ed. Boston, MA: Allyn and Bacon, 1994.

[7] J. S. Bruner, "The Act of Discovery", *Harvard Educational Review*, vol. 31, pp. 21-32., 1961.

[8] P. Corbeil, "Learning From the Children: Practical and Theoretical Reflections on Playing and Learning", *Simulation and Gaming*, vol. 30, no. 2, pp. 163-180, June 1999.

[9] J. S. Brown, A. Collins and P. Duguid, "Situated Cognition and the Culture of Learning", *Educational Researcher*, vol. 18, no. 1, pp. 32-42, February 1989.

[10] B. Beatty, *Preschool Education in America: The Culture of Young Children from the Colonial Era to the Present*, New Haven, CT: Yale University Press, 1995.

[11] G. Brougère, " Some Elements Relating to Children's Play and Adult Simulation/Gaming", *Simulation and Gaming*, vol. 30, no. 2, June 1999.

[12] D. Varga, "The Historical Ordering of Children's Play as a Developmental Taks", *Play and Culture*, vol. 4, no. 4, pp. 322-333, November 1991.

[13] J. Piaget, *Biologies and Knowledge: Essays on the Relations Between Organic Regulations and Cognitive Processes*, Paris: Gillemard, 1967.

[14] J. Piaget, *Psychology and Pedagogy*, Paris: Denoel-Gönthier, 1969.

[15] S. Thiagarajan, "How to Maximize Transfer from Simulation Games Through Systematic Debriefing", in F. Percival, S. Lodge, and D. Saunders (Eds.), *The Simulation and Gaming Yearbook 1993: Developing Transferable Skills in Education and Training*, pp. 47-52, London: Kogan Page, 1993.

[16] E. von Glasersfeld, "An Exposition of Constructivism: Why Some Like it Radical", *Journal of Research in Mathematics Education*, Monograh No .4, Reston, VA: National Council of Teachers in Mathematics, 1990.
[17] J. Lave, "Situating Learning in Communities of Practice", in L. Resnick, J. Levine and S. Teasley (eds.), *Perspectives on Socially shared Cognition*, pp. 63-82, Washington, DC: APA Press, 1991.
[18] L. Vygotsky, *Mind In Society: The Development of Higher-Psychological Processes* (M. Cole, V. John-Steiner, S. Scribner, and E. Souberman, Eds. And Trans.), Cambridge, MA: Harvard University Press, 1978.
[19] A. Collins, J. S. Brown, and S. E. Newman, "Cognitive Apprenticeship: Teaching the Craft of Reading, Writing and Mathematics", in L. Resnick (Ed.), *Knowing, Learning and Instruction: Essays in Honor of Robert Glaser*, pp. 453-494, Hillsdale, NJ, 1989.
[20] D. Bergin, "Influences on Classroom Interest", *Educational Psychologist*, vol. 34, no. 2, pp. 87-98. Spring 1999.
[21] S. J. Derry, J. R. Levin, H. P. Osana and M. S. Jones, "Developing Middle-School Students' Statistical Reasoning Abilities Through Simulation Gaming", in Lajoie, S. P. (Ed.), *Reflections on Statistics: Learning, Teaching, and Assessment in Grades K-12*, Mahwah, NJ: Lawrence Erlbaum Associates, 1998.

DR. NAWWAF KHARMA is Assistant Professor in the Electrical and Computer Engineering Department at Concordia University. His main areas of research are character/pattern recognition, artificial intelligence (AI) and developing innovative educational methods of engineering education.

LEON CARO is a student in the Electrical and Computer Engineering Department at Concordia University whose interest lies in the development of games for teaching digital logic.

VIVEK VENKATESH is a graduate student in the Educational Technology program at Concordia University's Department of Education. His interests lie in the investigation of technology as a a teaching tool, and in developing mathematics curriculum material.