TJER Vol. 11, No. 1, 69-76

Complementing Digital Logic Design with Logisim

SA Al-Busaidi

Department of Electrical & Computer Engineering, College of Engineering, Sultan Qaboos University, PO Box 33, PC 123, Al-Khoud, Muscat, Sultanate of Oman

Received 30 April 2013; accepted 23 March 2014

Abstract: Classified as free and open source software (FOSS), Logisim is a delightful tool that can easily be used to reinforce a solid understanding of the theoretical concepts related to a digital logic design course. Unlike LogicWorks, one of the most attractive features of Logisim is its ability to include user built libraries. This can result in the development of a library that models the complete set of integrated circuits (ICs) required for a digital logic design course. As a consequence, numerous merits can be observed regarding a student's learning level within such a course.

Keywords: Data sheet, Digital logic design, Free and open source software (FOSS), Integrated circuit (IC), LogicWorks, Logisim.

استكمال التصميم الرقمي المنطقي باستخدام برنامج

سمير أ. البوسعيدي

الملخص: من البرمجيات المجانية والمفتوحة المصدر برنامج ,(FOSS) وهو أداة جيدة يمكن استخدامها بسهولة لتعميق المفاهيم النظرية المتعلقة بمادة التصميم المنطقي الرقمي. على عكس (ICs) واحدة من السمات اللميزة والجذابة لبرنامج هو يتعلق بقدرتها على احتواء المكتبات التي بناها المستخدم. ويؤدي ذلك إلى تطوير المكتبة على غرار نمذجة المجموعة الكاملة للدوائر المتكاملة في هذه المادة. وبالتالي يمكن ملاحظة العديد من المزايا في مستوى تعلم الطلاب لهذه المادة.

الكلمات المفتاحية: ورقة البيانات، التصميم المنطقي الرقمي، البرمجيات الحرة ومفتوحة المصدر (FOSS)، الدوائر المتكاملة، Circuits Integrated (IC).

*Corresponding author's e-mail: albusaid@squ.edu.om

1. Introduction

A digital logic design course was developed at Sultan Qaboos University (SQU) to introduce students to the fundamental theory and basic design blocks of digital circuits. Students of theoretical fundamentals delve into Boolean algebra, a subarea of algebra associated with digital circuits. The fundamental gates, namely the AND, OR, NOT, Exclusive-OR gates (XOR), NOR and a combination of AND and NOT gates (or NAND gates), are a direct product of the use of Boolean algebra. These gates constitute the basic units for all digital logic design.

To complement both the theoretical background and the design of digital circuits, the course includes experiments that progress at an equivalent pace with the course lectures. Each experiment can be divided into two components: a pre-lab that is associated with a software simulation tool and the actual physical implementation of the experiment. The significance of the pre-lab is that it obligates students to practice the theory through simulation prior to conducting the experiment. On this basis, students not only conduct their experiments in a shorter time frame but also save resources through minimizing incorrect connections that could lead to the destruction of an integrated circuit (IC). Accordingly, software packages that can be utilized for simulation purposes are a core requirement for the successful delivery of digital logic design courses (Ahmad *et al*. 2013(a); Ahmad *et al*. 2013(b); Ahmed and Ruelens 2013; Ahmed *et al*. 2012;Ahmed and Al-Abri 2012; Ahmad and Al-Abri 2010; Al-Lawati and Ahmad 2003 and Al-Busaidy 2013).

This paper aims to share an experience using the free and open source software (FOSS) tool Logisim for a digital logic design course. In section 2, a detailed account of the topics covered in the course is given. Section 3 outlines the lab experiments that are assigned to complement the course's theoretical content. This is then followed in section 4 by a brief history and the current deployment of Logisim. Section 5 outlines the method by which Logisim was adopted within this course. In sub-section 5(a), an account of the basic capabilities of Logisim is given while in subsection 5(b) the method of extending Logisim's library is outlined. The results of adopting this tool within the course are then given in section 6, followed by a conclusion in section 7.

2. Digital Logic Design Course

This section recounts in detail the topics that are covered within the digital logic design course at SQU. The initial part of the course touches upon the concepts of signed and unsigned decimal number representation in binary, binary ones complement and binary twos complement. The conversion process of numbers is then expanded to include the conversion to and from the octal and hexadecimal bases. On the other hand, the representation of the alphanumeric keys associated with the keyboard is demonstrated through the American Standard Code of Information Interchange (ASCII) codes. The concept of including the parity bit for the purpose of detecting errors linked to ASCII code transmissions is also introduced.

The course also includes arithmetic operations on pairs of numbers that are represented in binary. Of particular interest is the key concept of performing the binary subtraction operation using the binary twos complement. The outcome of this method demonstrates the underlying ingenuity of utilizing a single arithmetic operation, addition, to conduct two different operations, addition and subtraction. By representing numbers in the binary twos complement, the subtraction operation can be converted into the addition operation. Justifiably, in this case only an adder is required to manipulate the pair of binary numbers.

The next portion of the course introduces the concept of Boolean algebra along with its associated theories and propositions. Through Boolean algebra, a complete set of basic functions for two variables is derived by means of a truth table. Among the functions are the well known AND, OR, NOT, exclusively-OR (XOR), NAND, NOR and exclusively-NOR (XNOR) functions. These functions, along with their schematic gate representation, are clearly shown at this stage. This is followed by introducing the concept of transforming AND, OR and NOT gates into either a NAND or NOR-only equivalent. As a consequence of this equivalent representation, both gates are coined as universal gates.

The combination of different sets of the basic functions permits for the emergence of new and more complex functions. These new functions can be made to reflect upon the ability of solving specific real life problems in light of digital design. In deriving these functions, the truth table that relates both input variables to the output function is essential. The function can be read directly from the truth table in either the sum of standard product form or the product of standard sum form (Tocci *et al*. 2010). A function in this form requires only two gate levels, which is essential if a gate propagation delay is critical to the design. On the other hand, by mapping every possible output of the functions from the truth table, the concept of the Karnaugh (K)-map is introduced as a methodology to

derive the minimized number of literal equivalent functions. Although this could be alternatively achieved through Boolean algebra manipulations, that may be a long and exhaustive procedure; however, the K-map is a graphics tool that is both intuitive and simple. The outcome- minimized number of literal function that is derived can be represented in either the sum-of-product form or the product-of-sum form (Tocci *et al*. 2010). A function derived using this method not only requires only two gate levels but also requires a minimum number of gates. This translates into the reduction of both propagation delay and the required power consumption of the design. Furthermore, at this stage of the course, functions derived from the combination of AND, NOR and NOT gates are converted into NOR and NAND equivalents. Two methods can be used for the conversions: Boolean algebra or a graphics method.

After establishing these basics, the course then focuses on numerous designs of digital circuits that perform useful functions. In effect, at this stage the focus is on application of digital circuits. For such reasons, it is essential to differentiate between two types of digital circuits-namely combinational circuits and sequential circuits. Whereby the input signals in combinational circuits flow in only one direction and lack memory devices, the defining attributes of sequential circuits are the signal feedback and memory devices. From a different perspective, sequential circuits can be viewed as a combinational circuit with the two aforementioned attributes. As a consequence, the course delves primarily into combinational circuits before introducing the concepts of memory and sequential circuits.

A number of functional circuits are introduced in the combinational design portion of the course. This includes the half and full adder and subtractors, two bit multiplier, decoder, encoder, multiplexer (MUX) and comparator circuits. It is worthwhile noting at this stage that the students experience a transition from theory to design. As it had been earlier stated, a subtractor can be replaced by an adder through converting the binary number into its twos complement form. Henceforth, at this stage, it is clearly shown how this can be achieved using the full adder circuit with an external XOR gate to perform both addition and subtraction operations.

On the other hand, the concept of the latch and flip-flop (FF) along with its different types are introduced in the sequential circuit portion. This includes three FF types-namely the data-flip-flop (D-FF), Jack Kilby flip-flop (JK-FF) and toggle flip-flop (T-FF). Unlike its combinational circuit counterpart, the analysis of sequential circuits requires state tables instead of truth tables. State tables comprise the current state of the memory devices along with the input variable from one end, and the next state along with the output variable from the other end. In relating the next state to the current state it becomes necessary to determine the input function of each FF used within the design. Interestingly, the analysis of sequential circuits can, therefore, be conducted in two stages. First, a combination stage, in which all memory elements are virtually removed from the design and the concern, is in obtaining the input FF equations for each memory element. Second, in the next state stage, analysis is conducted only on every individual memory element; hence the complete circuit is virtually removed with only the memory devices remaining. The analysis of sequential circuits finally leads to a graphic representation of the state table in the form of a state diagram. The course then focuses on the reverse process, or the design process. Initially, a state table is derived from a specific state diagram. The state table is derived taking into consideration a specific configuration of memory elements. The choice of memory elements will dictate the design of each FF input equation according to the FF characteristic table. From the state table, the design can subsequently be obtained.

After this theoretical background, the design of sequential circuits in the form of registers and counters are demonstrated. Two designs of registers in particular are studied-namely the parallel load parallel shift register and the serial load serial shift register. For counters, two designs of counters are studied: the asynchronous ripple counter and synchronous up/down counter.

Finally, the course terminates by briefly introducing different programmable devices such as random access memory (RAM), programmable logic device (PLD), programmable logic array (PLA) and programmable array logic (PAL). At this stage, the design methodology of deriving a read only memory (ROM) device using a decoder and implementing a function using a PLA are both demonstrated.

3. Lab Experiments

This section highlights the five lab experiments conducted at the same pace as the theoretical aspect of the course. In the first experiment, the newcomer is introduced to the experimentation work area, the breadboard, and the various types of ICs within the lab environment. This can be viewed as no more than a mere familiarization exercise.

Experiment 2 includes two parts. The aim of the first part is to convert a logical expression into a working digital circuit. As for the second part, the aim is to demonstrate how Boolean algebra can be utilized to express functions in equivalent forms.

The focus in experiments 3 and 4 is on designing combinational circuits. Experiment 3 is used to design

half and full adders, while experiment 4 demonstrates how to successfully include decoders and multiplexers in a design. At this stage, the K-map is used to simplify the function to be implemented in the experiment.

Finally, the focus in experiment 5 is on the design of sequential circuits. In previous years, the experiment involved designing and implementing a 2-bit up/down counter using the D-FF. This year, as a simple upgrade, the requirement was to design two sequential circuits that could count the number of ones in an 8-bit sequence. For each design, the main constraint factor was the types of ICs that had to be included within each design. The first design required the use of a single parallel load serial shift register, a full adder and four D-FFs, while the second design required a single parallel load serial shift register and a 4-bit synchronous up/down counter.

Table 1. A list of required IC used in the digital logic design course (Datasheet, 2000).

IC Name	Function
7400	Quad NAND
7402	Quad XOR
7404	Hex NOT
7408	Quad AND
7410	Triple NAND
7411	Triple AND
7420	Dual NAND
7421	Dual AND
7430	Single NAND
7432	Quad OR
7474	Dual D-FF
7485	4-Bit Comparator
7486	Dual XOR
74138	3-to-8 Decoder
74153	Dual 4x1 MUX
74157	Quad 2x1 MUX
74165	Parallel Load Serial Out Shift Register
74169	4-Bit Up/Down Counter
74283	4-Bit Full Adder

From the list of experiments, it is clear that students are exposed to a plethora of ICs through this course. Table 1 gives an extensive list of the ICs related to both the course content and experiments.

4. A Brief History and Current Deployment of Logisim

Logisim was introduced in 2000 by Carl Burch who was, at the time, based in the Department of Computer Science at the College of St. Benedict and at St. John's University. Burch realized that students require a simulator to build logic circuits without the superfluous knowledge linked to its physical building. The outcome of his realization was a key motivating factor to his development of Logisim. Another motivating factor was the unjustifiable cost that is usually tagged onto commercial products. This is especially true in circumstances in which institutions only benefit from an extremely small portion of the complete capabilities of an extensive software product. Such a case can be directly related to the software simulation product requirement for a basic course such as digital logic design. The next logical step for the author, therefore, was to build the digital logic software Logisim.

Burch´s concern for adopting appropriate software for digital logic design was shared by other institutes. This shared concern could be regarded as one of the main drivers behind spreading the Logisim project beyond its point of origin, and the use of Logisim as a teaching aid has experienced phenomenal growth during the twelve years since its creation.

Not only has it become popular in the United States, but it also has crossed international borders. In the US, it has been adopted by 75 different institutions, including Brown University, the Georgia Institute of Technology, and Princeton University. Outside of the US, it has been adopted by more than 30 institutes. It has spread across 15 European countries, including Germany, France and the United Kingdom. On the other side of the Pacific, it is used in seven institutions in both Australia and New Zealand. In Latin America, 11 institutions use Logisim in countries including Brazil and Argentina.

However its popularity in Asia is still lagging, with only 8 Asian institutes having adopted Logisim, including those in Saudi Arabia, Singapore and, more recently, Oman. As of this publication, no institute in Africa has accepted Logisim.

These findings have been compiled in Table 2, which summarizes the global spread of Logisim.

5. Integrating Logisim Into the Course

In previous years, the software package LogicWorks from Capilano Computing Systems (Richmond,

Table 2. The global spread of Logisim.

Country	Number of Institutions
Argentina	3
Australia	4
Austria	1
Bangladesh	1
Belgium	4
Brazil	3
Bulgaria	2
Canada	6
Chile	1
China	1
Croatia	1
Colombia	3
Cyprus	1
Czech	1
Denmark	1
Finland	1
France	4
Germany	2
Greece	2
Iceland	1
India	2
Lebanon	1
Mexico	1
New Zealand	3
Portugal	1
Russia	1
Saudi Arabia	1
Serbia	1
Singapore	1
Sweden	2
Switzerland	3
Taiwan	1
Turkey	4
UK	3
USA	75
Total	143

British Columbia, Canada) was the main simulation tool used in labs. On the positive side, it has a free version targeting academic institutions. However, LogicWorks also has limitations within the university context. Although LogicWorks is a versatile piece of software, it includes an extensive component library which is unnecessary for the course. Furthermore, as LogicWorks is a commercial product, graduating students either would have to purchase a copy of the software to pursue any digital design circuit simulation based on LogicWorks, or would be forced to work at a company or institute with access to the software in order to continue using it after graduation. Both scenarios are highly improbable given the relatively small electronics industry in the Sultanate of Oman.

Furthermore, while building experiments for the digital logic design course around LogicWorks, two important deficiencies were observed. First, within the digital logic design course, students design and analyze circuits with direct connections to single gates. Unfortunately, this would rarely be the situation in the real world. Circuits are naturally built around ICs which include, in most cases, multiple gates within a single chip. Ideally, the simulations of all experiments must be conducted using the ICs found within the lab. This, however, could not be attained using LogicWorks as it did not include all of the desired ICs within the lab environment.

Second, if a particular IC did exist in the LogicWorks library, its implementation was a source of confusion to the newcomer. This can be explained by observing the equivalent IC pin assignments of both the actual and simulation components. Unfortunately, the software company did not attempt to match the pin layout of the actual component when including the IC in the library. Figure 1 clarifies this problem using as an example the 3-to-8 decoder, IC 74138 (Data Sheet 2000). Figure 1a is the actual component layout while Figure 1b shows the LogicWorks model layout. As a result of this IC layout mismatch, the task of experimenting with and the simulation of digital circuits becomes all the more daunting and confusing to new students.

To contain the three constraining factors found in LogicWorks, Logisim was recently adopted at SQU to fit the digital logic design course simulation niche. The best part is that Logisim is FOSS which complies with the general public license (GPL). Under this license, students not only experience using the software as a computer aided design (CAD) tool for the course as undergraduates, but are further encouraged to take it along with them after graduating. Accordingly, this software can be considered a personal CAD tool that can assist graduates in the pursuit of building digital logic circuits at both professional and personal levels. Moreover, Logisim is light weight software in that it requires only a small amount of computing estate and is deployable in Windows, Linux and Mac operating systems. This is attributed to the fact that the software was created using Java.

Furthermore, the built in component libraries comprises most of the basic units required for designing digital logic circuits, which is more than required to successfully deliver the course as shall be outlined in the next section. Finally, Logisim has been built with the capability of expanding its built-in library with user-defined libraries. This final feature can be considered one of the most attractive of this software. Consequently, any instructor of digital logic design can readily tailor the software package according to both the course requirement and lab experiments.

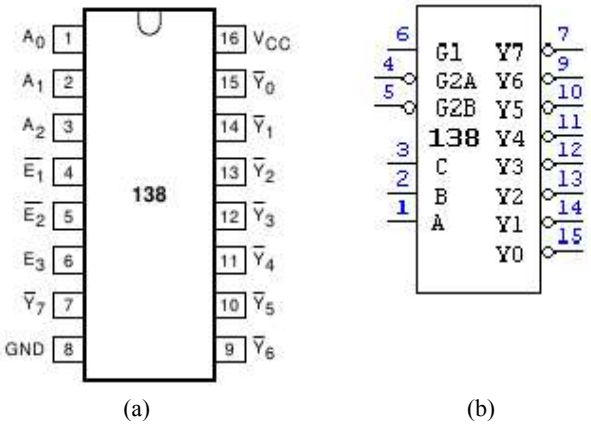
5.1 (A) Logisim's Basic Capabilities

This section is devoted to showing the various built-

in libraries of Logisim from a primer installation perspective. Logisim includes seven built-in libraries, namely a wiring, gates, plexers, arithmetic, memory, base and input/output library. In the wiring library, the most important components are the input and output probes, clock, power and ground. As for the gates library, it includes all of the basic gates. The multiplexer (MUX), demultiplexer (DEMUX) and decoder all form part of the plexer library. Included within the arithmetic library are the four basic arithmetic operators along with the comparator component. All three types of FFs, including the counter, register, shift register, random access memory (RAM) and read-only memory (ROM), form part of the memory library. While in the base library, the tools poke, edit, select, wire, text and label are readily available to the user. Finally, the input/output library contains a button, joystick, keyboard, light emitting diode (LED), and seven-segment and hexadecimal displays.

5.2 (B) Extending Logisim's Library

Although the built-in library included within Logisim can suffice to conduct most of the experiments, the built-in library components and the actual ICs have different layouts. This layout difference can be a major cause of confusion for the newcomer to the field. Furthermore, certain ICs require a particular sequence of pin configurations to operate correctly. As an example, consider the parallel load serial out shift register, IC 74165. Its package layout and pin assignment taken from an in-house built Logisim library is depicted in Fig. 2. To operate this IC correctly, its corresponding function table is shown in Table 3 (DataSheet 2002).



(a) (b)

Figure 1. The difference between the actual chip layout (a) and LogicWorks layout (b).

To successfully shift 8-bits from the input pins 3 to 6 and 11 to 14, to output pin 9, it is first necessary to load the data correctly. The parallel load operation of bits A to H into the IC requires that pin 1, indicated by

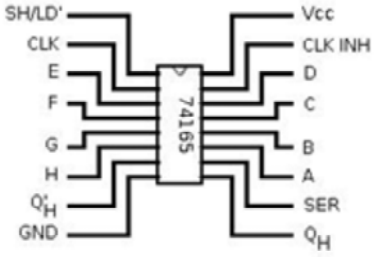


Figure 2. A Logisim representation of the in-house built parallel load serial out shift register IC 74165.

SH/LD', be initially low. This is followed by assigning pins 2 and 15 to low while subsequently changing pin 1 to high. The loaded bits can now be shifted out to pin 9. For every clock cycle, a transition from low to high will shift out one bit at a time, commencing from loaded bit H up to loaded bit A. Finally, as a bit is transferred to an adjacent pin, a new value from pin 10 is loaded at every clock cycle.

From this example, it can be stated that it is imperative that a component within Logisim operates in line with its function table from the IC's data sheet. This correspondence between the actual and simulation components has clearly not been included with the built-in library components of Logisim. However, included within Logisim is a particularly delightful feature that allows users to build their own libraries. To achieve this, a user simply designs a circuit in the usual manner and saves it. Upon opening a new project, the saved file can be included as a Logisim library. It was therefore, a logical step to build up a library inclusive of all the components found within the digital lab. As a result, the ICs listed in Table 1 were all modeled in accordance to each individual IC function table and pin layout.

6. Gained Experiences of Using Logisim

In contrast to the previous years where the pre-lab were designed at the gate level, the fall semester of 2012 has seen a shift from this routine. Given that all the necessary ICs had their equivalent model in our in-house built library IC0V2.cir, every pre-lab was simulated at the IC level.

By enforcing this shift, a number of visible merits were clearly observed. First, students realized that if their pre-lab operated as specified, then the physical work was no more than a mere copy and paste operation from the Logisim work pane onto the breadboard. From this perspective, the meaning of simulation gained a completely new dimension and students started to appreciate the concept of simulating prior to building. Moreover, the time students spent pondering

Table 3. Function table of IC 74165.

Inputs					Internal Outputs		Output
SH/LD	CLK INH	CLK	SER	Parallel A…H	Q'_A	Q'_B	Q_H
L	X	X	X	a…h	a	b	h
H	L	L	X	X	Q_{A0}	Q_{H0}	Q_{H0}
H	L	↑	H	X	H	Q_{An}	Q_{Gn}
H	L	↑	L	X	L	Q_{An}	Q_{Gn}
H	H	X	X	X	Q_{A0}	Q_{H0}	Q_{H0}

over their connections on the breadboard was also shortened.

Second, it was observed that the number of damaged components could be significantly reduced. Third, by exposing students to the function tables within data sheets, it was evident that students managed successfully to operate ICs with which they had had no prior experience. This had been achieved through the mindset that an unfamiliar IC should be treated as a black box which operates according to its function table only. This was especially true under the revised experiment 5 that required students primarily to understand the operation of each individual IC from its data sheet prior to connecting the ICs together. In this final experiment, students had to think in terms of designing a complete circuit composed of multiple ICs and viewing each IC as a simple block achieving one specific task.

Fourth, given that the components used in the simulator were identical to the real world components, groups of students could easily work together at their own leisure. This permitted the labs to be less frequented by digital logic design students, especially at the end of semester when students flock to conduct experiments in preparation for the lab exam. Finally, as Logisim follows in the footsteps of FOSS, students were encouraged to take their personal copies of the software and the in-house built IC library. With this final merit, it is hoped that all digital logic design students will continue using the software to solve real life problems through simulation prior to building a functional product.

7. Conclusions

In the FOSS world, there are applications that suit almost every field. This includes with no exception the field of CAD tools required for digital logic design courses. One particularly nice piece of software is Logisim. On the one hand it is light weight and can run on practically every operating system. On the other, it can be tailored and expanded in accordance with digital logic design course requirements.

In our course, it was possible to change students' perspectives towards the importance of simulation by having them build an extensive IC library which includes ICs that are linked to both the course and its experiments; this was only achievable given Logisim's capability to include new libraries. This permitted modeling of ICs in their actual pin configuration while realizing the IC's function table. Through this, students gained greater insight into how individual IC components should be treated and connected to build a functional system. Finally, as an added benefit of FOSS, students were encouraged to use this CAD tool to tackle new problems of the type that may be encountered in their future engineering careers.

References

Ahmad A, Ruelens D, Ahmad S (2013a), Development of verification tool for minimal boolean equation. IEEE Technology and Engineering Education, (ITEE) 8(4):29-34.

Ahmad A, Al-Busaidi SS, Al-Mushrafi MJ (2013b), On properties of PN sequences generated by LFSR - a generalized study and simulation modeling. Indian Journal of Science and Technology 10(10):5351-5358.

Ahmad A, Ruelens D (2013), Development of digital logic design teaching tool using MATLAB & SIMULINK. IEEE Technology and Engineering Education 8(1):7-11.

Ahmad A, Al-Abri D, Al-Busaidi SS (2012), Adding pseudo-random test sequence generator in the test simulator for DFT approach. Computer Technology and Applications 3(7):463-470.

Ahmad A, Al-Abri D (2012), Design of a pseudo-random binary code generator via a developed simulation model. ACEEE International Journal on Information Technology 2(1):33-36.

Ahmad A, Al-Abri D (2010), Design of an optimal test simulator for built-in self test environment. Journal of Engineering Research 7(2):69-79.

Al-Busaidi S (2013), Complementing digital logic design with Logisim. Proceedings of Free and Open Source Software Conference (FOSSC-13), held at Sultan Qaboos University, Muscat, Oman 18-19 February, 4-9.

Al-Lawati A, Ahmad A (2003), Realization of a simplified controllability computation procedure - A MATLAB-SIMULINK based tool. Journal for Scientific Research - Science and Technology 8:131-143.

Burch C (2000), Logisim: A graphical system for logic circuit design and simulation. Journal of Educational and Resources in Computing 2(1):5-16.

Data Sheet of "IC DM74LS138 Decoder / Demultiplexer", Fairchild Semiconductor, Revised March 2000.

Data Sheet of "IC 74165 Parallel-Load 8-Bit Shift Register", TI, SDLS0620, Revised February 2002.

http://maps.google.com/maps/ms?ie=UTF8&oe=UTF8&msa=0&msid=209845857912254026337.00049c67b154d0e5433a0.

Tocci RJ, Widmer NS, Moss GL (2010), Digital Systems: Principles and Applications. Eleventh Edition, Prentice Hall.