Complementing Digital Logic Design with Logisim

Samir A. Al-Busaidi[1]

***Abstract*— The FOSS Logisim is a delightful tool that can be easily used to enforce a solid understanding of the theoretical concepts related to the DLD course. Unlike LogicWorks, one of the most attractive features of Logisim is with respect to its ability to include user built libraries. This resulted in the development of a library that modelled the complete set of ICs required for the course. As a consequence, numerous merits could be observed at the student's learning level of the course.**

***Index Terms*— datasheet, DLD, FOSS, IC, LogicWorks, Logisim.**

I. INTRODUCTION

THE digital logic design (DLD) course has been developed at Sultan Qaboos University (SQU) to introduce students to the fundamental theory and the basic design blocks of digital circuits. Students in the theoretical fundamentals delve into a type of algebra associated with digital circuits known as Boolean algebra. The fundamental gates, namely the AND, OR, NOT, XOR, NOR and NAND gates, are a direct product of utilizing Boolean algebra. These gates constitutes the basic units for all DLDs. To complement both the theoretical background and the design of digital circuits, the course includes experiments that progresses at an equivalent pace with the course lectures. Each experiment can be divided into two components, a prelab, that is associated with a software simulation tool, and the actual physical implementation of the experiment. The significance of the prelab is that it obligates students to practice the theory through simulation prior to conducting an 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, a suitable software package that should be utilized for simulation purposes is a core requirement for the successful delivery of the course.

This paper is aimed at sharing an experience of using the free and open source software (FOSS) tool Logisim for the DLD 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 in which Logisim has been adopted within this course. In sub-section VI(a), an account on the basic capabilities of Logisim is given while in subsection VI(b(the method of extending Logisim's library is outlined. The results of adopting this tool within the course is then given in section VII followed by a conclusion is section VIII.

[1]Department of Electrical & Computer Engineering, College of Engineering, Sultan Qaboos University, P.O. 33, Al-Khodh, Muscat-123, Sultanate of Oman (phone: 968-24141222; fax: 968-24413454; e-mail: albusaid@ squ.edu.om)

II. THE DLD COURSE

This section is devoted to account in detail the topics that are covered within the DLD course. The initial part of the course brushes 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 too. 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 follows on to include 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 underlining 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 tables. Among the functions are the well known AND, OR, NOT, XOR, NAND, NOR and 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 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 [1]. A function in this form requires only two gate levels which is essential if 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 of deriving the minimized number of literal equivalent function. Although this could be alternatively achieved through Boolean algebra manipulations, that may be a long and exhaustive procedure, however, the K-map is a graphical 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 [1]. 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 graphical method.

After accumulating this basic amount of knowledge, the course then focuses on numerous designs of digital circuits that perform useful functions. In effect, at this stage, the focus is on applications 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 adders and subtractors, two bit multiplier, decoder, encoder, multiplexer (MUX) and comparator circuits. It is worthwhile noting at this stage, the 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 the different types of FFs are introduced in the sequential circuit portion. This includes three FF types, namely, the D-FF, JK-FF and T-FF. Unlike its combinational circuit counterpart, the analysis of sequential circuits requires state tables instead of truth tables. State tables comprises 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. Firstly, 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. Secondly, the next state stage, in which the 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 graphical representation of the state table in the form of a state diagram. The course then focuses on the reverse process, 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 on the design 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 design 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.

III. LAB EXPERIMENTS

This section highlights the lab experiments that are required to be conducted at the same pace of the theoretical background. The course includes 5 experiments. 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 both 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 into 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. As a simple upgrade, this year the requirement was to design two sequential circuits that could count the number of 1's in a 8-bit sequence. For each design, the constrain 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.

From the list of experiments it is clear that students are going to be exposed to a plethora of ICs. To summarize the types of ICs, table 1 gives an extensive list which is related to both the course content and experiments.

IV. A BRIEF HISTORY AND CURRENT DEPLOYMENT OF LOGISIM

The inception of logisim occurred in 2000 by its author Carl Burch [2]. At that moment in time the author was based in the department of computer science at the college of St. Benedict and at St. John's university. His realization had been 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 developing Logisim. Another motivating factor instilled into the author was the unjustifiable cost factor that is usually tagged to 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 DLD. It had been, therefore, the logical step for the author to build the digital logic software named Logisim.

Burch´s concern for adopting an appropriate software for DLD was surely shared across other institues. This shared concern could be regarded as one of the main drivers behind outbounding the Logisim project from its local based creation. Therefore, the utilization of logisim as a teaching aid has experienced phenomenal growth during the last twelve years from its creation. Not only did it become popular in the USA but also it had crossed boundaries into other countries too. In the US it had been adopted by 75 different institutions, such as Brown University, Georgia Institute of Technology and Princeton University to a name a few. Outside the US it had been adopted in more than 30 institutes spread across 15 European countries, including Germany, France and the UK. On the other side of the Pacific, it is used in 7 institutions spread across both Australia and New Zealand. While in Latin America, 11 institutions are utilizing Logisim over 5 different countries including Brazil and Argentina. However its popularity across Asia is still lagging, with only 8 institutes having adopted Logisim, including Saudi Arabia, Singapore and recently Oman. Finally, Africa has yet to accept logisim within its boundaries as no single African institute is observed to use it. This finding has been compiled in table 2, and can be utilized as a summary of the the global spread of Logisim as per [3].

Table 1. A list of required IC used in the DLD course

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

V. INTEGRATING LOGISIM INTO THE COURSE

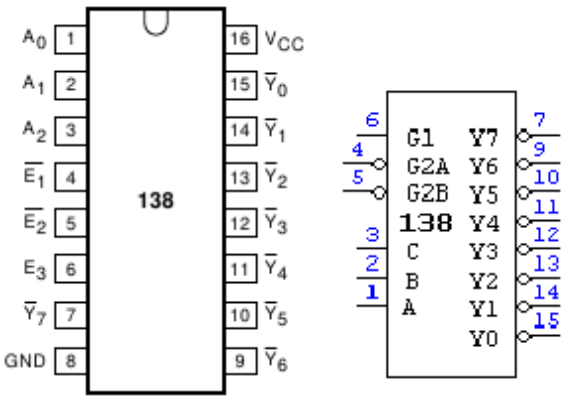
In previous years, the software package LogicWorks, from Capilano Computing Systems [4], was the main simulation tool that was used in the labs. It was a free version targeting academic institutions. Although being quite a versatile software that includes an extensive component library, however, this library was an uneccessary addition to the teaching of the course. Furthermore, being a commercial product, graduating students would have to either purchase a copy of the software to pursuit any digital design circuit simulation based on LogicWorks, or would be forced to work at a company or institute with access to the software. Both scenarios, are highly improbable given the relatively small electronics industry harbored within the Sultanate of Oman. Furthermore, while building experiments for the digital logic course around LogicWorks, two important deficiencies had been observed. First, within the course, students are designing and analysing circuits with direct

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

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. Idealistically, the simulations of all experiments must be conducted using the ICs found within the lab. This, however, could not be attained using the copy of LogicWorks at hand as it did not include all of the desired ICs within the lab environment. Secondly, 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 into the library. Figure 1 clarifies this problem using as an example the 3-to-8 decoder, IC 74138 [5]. Figure 1.a is the actual component layout while figure 1.b shows the LogicWorks model layout. As a result of this IC layout mismatch, the task of experimenting and simulating of digital circuits becomes all the more daunting and confusing to new students.

To contain the three constraining factors found in LogicWorks another software package, Logisim, had been recently adopted to fit the DLD simulation niche. For the best part it was a free and open source software (FOSS) that complied with the general public license (GPL). Under this license, students not only experience using the software as a computer aided design (CAD) tool for DLD as undergraduates, but are further encouraged to take it along with them after graduating. Accordingly, this software can be considered as a personal CAD tool that can excel graduates in pursuit of building digital logic circuits at both the professional and personal levels. Moreover, Logisim is a light weight software requiring only a small amount of computing estate, while being deployable on either Windows, Linux or the Mac operating systems. This is attributed to the fact that the software has been 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 as one of the most attractive features of this software. Consequently, it can be considered that any instructor of DLD can readily tailor the software package according to both the course requirement and lab experiments.



(a) (b)

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

VI. (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 7 built-in libraries, namely a wiring, gates, plexers, arithmetic, memory, base and input/output library. In the wiring library, the 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 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, RAM and 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 can be ignored as its components are irrelevant to the course.

VI. (B) EXTENDING LOGISIM'S LIBRARY

Although the built-in library included within Logisim can suffice to conduct most of the experiments, however, 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 figure 2. To operate this IC correctly, its corresponding function table is shown in table 3 [6].

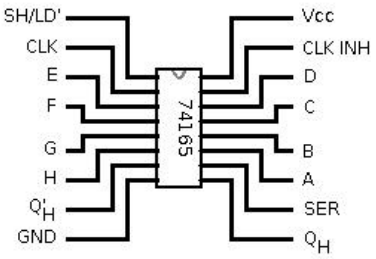


Fig 2. A Logisim representation of the in-house built parallel load serial out shift register IC 74165.

To successfully shift 8-bits from the input pins, pins 3 to 6 and 11 to 14, to output pin 9, it if 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 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 datasheet. 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.

Table 3. Function table of IC 74165

Inputs					**Internal Outputs**		**Output Q_H**
SH/LD'	CLK INH	CLK	SER	Parallel A ... H	Q'_A	Q'_B	
L	X	X	X	a ... h	a	b	h
H	L	L	X	X	Q_{A0}	Q_{B0}	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_{B0}	Q_{H0}

VII. GAINED EXPERIENCES OF USING LOGISIM

In contrast to the previous years, where the prelab 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 prelab was simulated at the IC level. By enforcing this shift a number of visible merits were clearly observed. Firstly, students realized that if their prelab 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 over their connections on the breadboard were also shortened. Secondly, it was observed that the number of damaged components could be significantly reduced. Thirdly, by exposing students to the function tables within datasheets, it was evident that students managed successfully to operate ICs that they had no prior experience with. 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 to primarily understand the operation of each individual IC from its datasheet 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, viewing each IC as a simple block achieving one specific task. Fourthly, 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 DLD students, especially at the end of semester where students would 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 DLD students will continue utilizing the software to solve real life problems through simulation prior to building a functional product.

VIII. CONCLUSION

In the FOSS world, there are applications that can suite almost every field of profession. This includes with no exception the field of CAD tools required for the DLD course. One particularly nice piece of software is Logisim. On one hand it is light weight and can run on practically every operating system, while on the other it can be tailored and expanded in accord with the DLD course requirement. Through building an extensive IC library which includes ICs that are linked to both the course and its experiments, it was possible to change the students perspective towards the importance of simulation. 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. by this students gained greater insight into how individual IC components should be treated and connected together to build a functional system. Finally, under the benefit of FOSS, students were encouraged to continue pursuing using this CAD tool to engage new problems that may be encountered in their engineering career.

REFERENCES

[1] Tocci, R. J., Widmer, N. S. and Moss, G. L., Digital Systems: Principles and Applications, Eleventh Edition, Prentice Hall, July 2010.

[2]Burch, C. "Logisim: A graphical system for logic circuit design and simulation", Journal of Educational and Resources in Computing, Vol. 2, No. 1, P.P. 5-16, March 2002.

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

[4] LogicWorks, at www.capilano.com

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

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