IJST EXPLORE International Journal of Science and Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-7318

International Journal of Science and Technology Volume 2 No. 8, August, 2013

Microcontroller Based Real-Time Emulator for Logic Gate and Structured Logic Devices

Uzedhe O. Godwin[1], Prof. H. C. Inyiama[2], Udeze Chidiebele C.[3] Mbonu Ekene S.
Department Electronics and Computer Engineering, Nnamdi Azikiwe University Awka, Anambra State, Nigeria.

ABSTRACT

Often a desired digital system cannot be implemented because the digital device required cannot be obtained in the local market. If such a device must be sourced from abroad, there is a lead time before the component ordered may be obtained. Such a lead time is not always tolerable because of the urgency placed on certain projects. The microcontroller based real-time emulator presented in this paper finds a way to make the microcontroller become a convenient substitute for any digital device that is readily unavailable. The microcontroller is programmed to have the capacity to emulate (or behave like) several different logic devices, and opens up a new range of possibilities. Students can perform laboratory experiment and submit lab reports on any of the emulated devices. Furthermore, when the microcontroller is adequately configured, it can be wired into any digital circuit to stand in for any logic device it is to emulate. In order to support the user, the emulator includes in its output (apart from its input/output pins) both text and graphic displays for operational information and graphical view of devices being emulated. It is therefore possible to provide a very powerful digital system laboratory by replicating the microcontroller based emulator and this will be a more virile solution to digital lab equipment provision than when actual logic devices are used. The lack of fund has always been the major reason for the under-development, and lack of educational facilities in most poor nations of the world. This novel approach to digital system lab equipment provision could be one of the best options to passing digital design knowledge to the under-developed nations through the provision of low cost laboratory equipments for effective knowledge transfer in their educational systems.

Keywords: *Emulator, Microcontrollers, Logic Devices, Experiment, Designs*

1. INTRODUCTION

A hardware design project usually goes through the following steps: project specification, design, simulation, prototyping, and production. Accurate simulations of large systems can be extremely slow and the reduced reliability of faster, more abstract simulation models cannot always be tolerated [1]. Software simulators are inexpensive and very flexible. On the other hand, prototypes are very expensive and typically inflexible. As the complexity of a system design increases, the efficiency of its software simulator decreases and the cost of its prototype increases. The gap between these two steps in the design of a complex system presents a major challenge to designers. Moving too early to prototyping may prove costly as prototypes are hard to modify. Extending the simulation phase too long delays the schedule while correct designs are not guaranteed. An attractive option is to build a prototyping test bed which is flexible enough to emulate different systems so that its cost is amortized as it is re-used.

In the lower end of a digital system experimentation and design, various types of logic gates and structured devices are usually needed. These gates and devices are however usually not available at the time of design, thereby slowing down the industry-to-market time of the end product. Another challenge is in the laboratory when students are to carry out digital design experiments. Different logic gates integrated circuits (ICs) and structured logic ICs will be required at different times for which their availability and the required number are not always guaranteed.

The solution is to model a single system whose function is to mimic the functionality of each of these devices to serve as the device itself in a system under design. Such a ubiquitous system which emulates the functions of other devices is the structured logic emulator.

Emulation is a powerful tool especially for the later stages of the development phase. It can provide the necessary environment for the system development before the device construction. It also provides verification capabilities that allow one to build a testing environment before reaching the stage of chip device prototyping. Emulators have the necessary flexibility needed for validation purposes, which chip device prototypes lack, by using reprogrammable devices as underlying hardware. Hardware emulation is a technique with which we imitate the behavior of a system under design or parts of it with another piece of hardware. The main objective of emulation technology is to provide a fast, efficient and feature rich debugging environment for the system. It can be thought of as a competitive technique to simulation, although there are fundamental differences on the approach that each technique takes and the level of abstraction and the stage of the development phase that they target [2]. We can deduce that the main difference between simulation and emulation is that while

simulation mimics the outward appearance of a system, emulation mimics the cause processes used by the real system. This is more evident by the fact that hardware emulation requires that the description of the parts under test are synthesizable whereas this is not the case with simulation. The simulator needs only a behavioral model of the system.

Logic emulation offers a compromise between the flexibility of software simulation and the speed of a hardware model. Logic emulation is distinguished from logic simulation in that the circuits being simulated are compiled directly into hardware. However, unlike the hardware model, a logic emulator can be quickly re-programmed, via software only, to emulate a new circuit. Besides the cost saving of re-use, the ability to quickly reconfigure minimizes time between design iterations. Like the hardware model, a logic emulator can be integrated directly into a target system [3].

2. THE NEED FOR A LOGIC EMULATOR

It is not always necessary to develop software directly on required hardware in every case; the traditional alternative for developing digital systems is to use simulation to debug portions of a system, [4]. The simulator can execute some of the development steps. The simulator is a software tool. Microcontroller's software is usually simulated on another microprocessor platform with an operating system. Normally it is a Personal Computer with appropriate simulating program. The gain for the designer is quality, standard and comfortable user-friendly environment that runs on the machine with many times higher computation rate. The simulator executes exactly most operations of the microcontroller, especially arithmetic and logic operations which are simulated in detail. The designer has direct control over the registers, memory or the state of some integrated peripherals as output port latches, UART transmitter, timers etc. Software breakpoints can be used very well, time consumption can be measured precisely and tracing or stepping is also usable. The states and values of function blocks as registers or memories are available in different forms. They can be watched as numbers in various number systems, character representation, back translated executable code etc. Anything can be changed without limitation during simulation. The simulator has no connection to real applications and it is the source of the restrictions for simulation at real time processes. So this is the reason for presenting the emulator.

Emulation technology is fundamental to being able to create embedded applications [5]. The emulator is an electronic circuit with microcontroller of the same type as is used in the development of applications. Minimum tolerance between required and used type is allowed. The emulator is constructed as a stand-alone device with its own user interface (display and small keypad). The best solution is a complete integrated development environment with a simulator and a higher programming interface for example a C++ compiler or a macro assembler. The other case is a simple "suit box" emulator of a single-chip microcontrollers independent of any Personal Computer. Every command has to be input through a small keypad and everything should be watched on an LCD display. The emulator runs on another circuit of the same type, as is the emulated one. The essence and efficiency are based on the fact that the developed software runs in the target microcontroller in "real" time and environment without the limits the simulator has. On the other hand if there is justified requirement for better properties or performance of the emulator, a better emulator can certainly be found or developed. The best emulators are on-chip support circuits or special programmable logic arrays. Evaluation of the emulated circuit state is done by another chip situated very close to the emulated one. It is difficult and expensive to produce these kinds of emulators. The achieved electrical (input/output capacitance and resistance, highest frequency) and thermal parameters (lead-off-lose thermal power, working temperature range) of the emulator should be equal to conventional microcontroller.

The significant use of emulation during the application development process means that improvements in emulation technology can have a direct and substantial bearing on development productivity, and ultimately time to market [5] availability of the desired product.

In the school laboratory where a number of students are required to carry out several design experiments as part of courses taken on digital systems, several experimental IC devices are usually needed. These devices in their number may be too expensive and sometimes not available as at when they are needed. Hence there is the need for system capable of emulating the functionality of the desired device in the lab and should be made ready easily when needed. The presence of this system also makes it possible for the student and the design engineer to experiment and develop his/her project outside the lab due to the emulator's mobility.

3. EMULATOR TECHNOLOGIES AND METHODOLGY

There are several technologies to process Boolean equations containing million of lines or equivalent gates [6, 7].

1. Personal computer or workstation based on Intel microprocessor can be used. Each equation will be processed sequentially using software approach, because only one processor exists although high-performance. The cost and time consumptions of this solution are very high.

2. Specialized parallel processor based on programmable logic devices (PLD). In this case high level of parallelism processing of equations compensates for relatively low clock speed. This reprogrammable solution is absolute winner in performance. But a significant drawback is absence of flexibility, which is typical for software approaches. Moreover, implementation into PLD is high

cost if the volume of future distribution will be only tens of thousands of chips.

3. The third solution involves the integration of CPU, PLD and ASIC (application specific integrated circuit) with advantages, such as [8].

   (a) Flexibility of Boolean equations programming, which allows on-the-fly editing specification in form of source code.
   (b) Minimal possible instruction set, which allows simple circuit solutions.
   (c) Parallelization of Boolean equations processing, because of PLD ideology, but with CPU elements, which means having lots of interconnected single-bit processors with simple instruction set for parallel programming.
   (d) Multiprocessor implementation into ASIC, which allows maximum clock rate and minimal cost per chip for large manufacturing volumes (more than 10 000), low power consumption.
   (e) Pipelining of Boolean equations processing is an exclusive property, which is typical to multiprocessor systems, where pipelined processing is one of the main operation mode besides parallel and sequential ones.

Systematic design methods (called *design methodologies*) are necessary for successfully designing complex digital hardware. Logic circuits are used to build computer hardware, as well as many other types of products. All such products are broadly classified as digital hardware.

The technology used to build digital hardware has evolved dramatically over the past decades. The advent of integrated circuits made it possible to place a number of transistors, and thus an entire circuit, on a single chip. In the beginning these circuits had only a few transistors, but as the technology improved they became larger. By 1970 it was possible to implement all circuitry needed to realize a microprocessor on a single chip. Although early microprocessors had modest computing capability by today's standards, they opened the door for the information processing revolution by providing the means for implementation of affordable personal computers. In the early 1990s microprocessors could be manufactured with a few million transistors, and by the late 1990s it became possible to fabricate chips more than 10 million transistors. Presently chips can have a few hundreds of millions of transistors [9].

Moore's law is expected to continue to hold true for at least the next decade. A consortium of integrated circuit manufacturers called the Semiconductor Industry Association (SIA) produces an estimate of how the technology is expected to evolve, known as the SIA Roadmap [10]. This estimate predicts the minimum size of a transistor that can be fabricated on an integrated circuit chip. The size of a transistor is measured by a parameter called its gate length. It is expected that the gate length will reduce steadily to about 35nm by the year 2012. The number of transistors per chip is expected to grow to 100 million transistors by the year 2012. The largest chip size is expected to be about 1300mm^2 at that time [10]; thus chips with up to 1.3 billion transistors will be possible.

The designer of digital hardware may be faced with designing logic circuits that can be implemented on a single chip or, more likely, designing circuits that involve a number of chips placed on a printed circuit board (PCB). Frequently, some of the logic circuits can be realized in existing chips that are readily available. This situation simplifies the design task and shortens the time needed to develop the final product.

Different design methodologies differ in their choice of number and levels of design abstractions used during the design process and the manner of constraints on the translations between the abstraction levels. These constraints are usually in the form of use of a particular structure type at the lower level of design abstraction while translating the design description that exists at a higher level of abstraction.

4. DIGITAL LOGIC EMULATOR DESIGN

In order to properly emulate several digital logic devices, the logic pattern of these devices have to be properly represented and each device pin configuration well considered during the design such that the emulator can clearly mimic the behavior of a device through its input and output pins. This logic pattern however, are to be vividly defined in accordance to the input and output relationship of each device to be emulated and mapped into the emulator.

A more virile way to mapping these patterns to yield the required outputs when inputs are applied in real time is the use of Read Only Memory (ROM) device. When the device logic (or bit pattern) are created in tables, they can be burn into a single ROM device which now seat on a board as though it is the device being emulated. It can however change its behavior in real time when input patterns are changed to produce different output logic pattern.

This ROM approach to digital emulator design as depicted in figure 4.0, can be used to emulate several number of fixed logic functions. This is due to the fact that the length of ROM to be used for a particular emulator is determined by the number of devices to be emulated and the maximum number of input/output connections that are required. Furthermore, ROMs can be cascaded to increase the memory space required. Hence, more algorithms can be added to an emulation board to increase its capability. In a 16K x 8bit ROM with 14 address lines A0 – A13 and 8 output lines, if we desire to emulate devices with maximum of 8 input lines, we can use address lines A0 – A7 as inputs. The remaining six address lines (i.e. A8 – A13) can then be used for device selection and so a total of $2^6 = 64$ devices can be emulated.

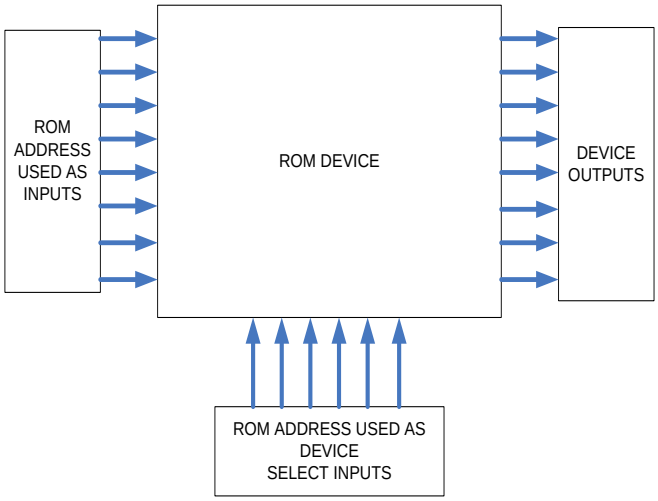

Figure 4.0: ROM Device as an Emulator

As shown in table 4.0, a single table containing the required logic pattern of all devices to be emulated by the emulator is developed and embedded in a single ROM device. Using the address line, each location containing a specified device algorithm (or logic pattern) can be accessed to produce required output pattern.

Table 4.0: Emulated Device Logic Pattern

DEVICE FUNCTION	SELECT INPUT (UPPER ROM ADDRESS) A_{13} A_{12} A_{11} A_{10} A_9 A_8	DEVICE INPUT (LOWER ROM ADDRESS) A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0	DEVICE OUTPUT (IN ROM LOCATION) Q_7 Q_6 Q_5 Q_4 Q_3 Q_2 Q_1 Q_0
NOT GATE	0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
	0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1
2-INPUT OR GATE	0 0 0 0 0 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
	0 0 0 0 0 1	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1
	0 0 0 0 0 1	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 1
	0 0 0 0 0 1	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 1
2-INPUT AND GATE	0 0 0 0 1 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
	0 0 0 0 1 0	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0
	0 0 0 0 1 0	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0
	0 0 0 0 1 0	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 1
2-INPUT NAND GATE	0 0 0 0 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1
	0 0 0 0 1 1	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1
	0 0 0 0 1 1	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 1
	0 0 0 0 1 1	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0
2-INPUT NOR GATE	0 0 0 1 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1
	0 0 0 1 0 0	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0
	0 0 0 1 0 0	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0
	0 0 0 1 0 0	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0
2-INPUT EX-OR GATE	0 0 0 1 0 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
	0 0 0 1 0 1	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1
	0 0 0 1 0 1	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 1
	0 0 0 1 0 1	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0
2-INPUT EX-NOR GATE	0 0 0 1 1 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 1
	0 0 0 1 1 0	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0
	0 0 0 1 1 0	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0
	0 0 0 1 1 0	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 1
3-INPUT OR GATE	0 0 0 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
	0 0 0 1 1 1	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1
	0 0 0 1 1 1	0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 1
	0 0 0 1 1 1	0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 1

	0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1	0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1

For example, a 2-Input OR gate function is emulated when address lines $A_{13} - A_9$, $A_7 - A_2$ are grounded and line A_8 is connected to the circuit V_{cc}. Address line A1 and A0 serve as the input while the output is at Q_0. All other functions can be selected in like manner for the emulation of a required device.

However, in order to cope with the requirements and the needs for which this emulator is being developed, the use of microcontroller which in itself possesses an internal ROM was used. This is to enable us deal with the flexibility of usage required in this emulator device. The use of a microcontroller also provides us with increased capability in a single chip. Text and graphic user interfaces are also made available alongside access control keys and multiplicity of input/output pins achievable by multiplexing through the high speed provided by the microcontroller chip.

5. THE DESIGN PROCESS

Any design process comprises a basic sequence of tasks that are performed in various situations. This sequence is presented in Figure 5.1. Assuming that we have an initial concept about what should be achieved in the design process, the first step is to generate an initial design. This step often requires a lot of manual effort because most designs have some specific goals that can be reached only through the designer's knowledge, skill, and intuition. The next step is the simulation of the design at hand.

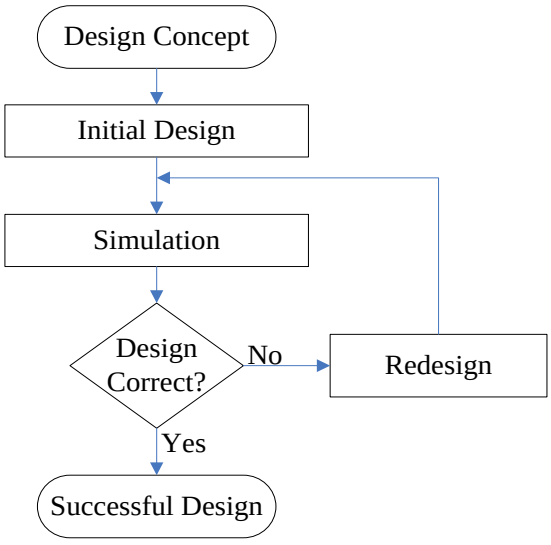

Figure 5.1: The Design Loop

To carry out the simulation successfully, it would be necessary to have adequate input conditions that can be applied to the design that is being simulated and later to the final product that has to be tested. Applying these input conditions, the simulator tries to verify that the designed product will perform as required under the original product specifications. If the simulation reveals some errors, then the design must be changed to overcome the problems. The redesigned version is again simulated to determine whether the errors have disappeared. This loop is repeated until the simulation indicates a successful design. A prudent designer expends considerable effort to remedy errors during simulation because errors are typically much harder to fix if they are discovered late in the design process. Even so, some errors may not be detected during simulation, in which case they have to be dealt with in later stages of the development cycle.

5.1 Analysis

The design of any new electronic system or device equipped with microcontroller needs premeditation at the beginning and the subsequent design of the electric circuit, printed copper board and the development of the microcontroller's software of course. Discussion is necessary in a designer team. Even if everything seems to have been done perfectly the design may not be successful. Different prototypes, experiments and another laboratory investigation help to eliminate possible development errors. When hardware is constructed or activated it is often used. There is very low probability that the first version of the software is definitive. It is recommended to use different tools, aids and methods to increase the probability and shorten the development time. A designer of digital systems is faced with two basic issues. For an existing logic network, it must be possible to determine the function performed by the network. This task is referred to as the analysis process. The reverse task of designing a new network that implements a desired functional behavior is referred to as the synthesis process. In this paper we consider the analysis process.

In this emulator design, several existing logic devices with predefined logic networks are to be emulated. Hence the following analysis is done:

- The emulator block should consist of an input subsystem through which these logic devices can be selected for emulation. A matrix numeric/alphanumeric keypad is proposed for such an input as it gives room for multiple selections.
- The user of the emulator needs to know what device was selected, its configuration, and its physical

connectivity. This forms the output subsystem and requires that an output display capable of handling alphanumeric characters and/or graphic be used.

- Peculiar to such an emulator is the need for an Input/Output (I/O) port through which data enter and leave the emulator. This port subsystem need be dedicated only for signal flow to and from the control subsystem, and consist of several lines of data that cater for the largest device (in terms of I/O pins) to be emulated. However, the envisaged control subsystem may not have such number of I/O pins. Due to insufficient routing resources, time-multiplexed virtual-wiring [3] may be necessary for emulation systems, at the expense of further reducing the overall system speed [11]. A Programmable Peripheral Interface (PPI) device may be used to expand the I/O pins of the control subsystem to meet with the design demand.
- As a micro-code ROM-based emulation system, the control subsystem requires an efficient microprocessor/microcontroller device with large enough code and data memories for storage of a large database and efficient data manipulation. This will give room for flexibility and also allow future updates. Such device should also be of high speed so the emulator can meet up with the throughput of several devices when used in replacement.
- Lastly, since the emulator's control system will bear an embedded microprocessor/microcontroller, embedded driving software program needs to be written for it. Such software program will requires any of the embedded software languages understandable to the processor device of choice. Typically, assembly language and/or C language which converts to hex or binary code using an assembler or any good compiler will be of good choice.

5.2 System Specifications

ITEM	CONTENT
Controller IC	Microcontroller
Interface IC	Bus Expander
Display Type	LCD (Graphic & Text)
Dedicated Input Lines	8
Dedicated Output Lines	16
Input/Output Lines	36
Control Lines	17
Input Mode	Digital (3-5V)
Output Mode	Digital (3-5V)
Text Format	Alphanumeric
Operating Temperature	$20 – 70^0C$
Storage Temperature	$30 – 80^0C$
Power Supply	11-20V d.c/220V a.c
Power Extension	Yes, 3

5.3 Hardware Subsystem Design

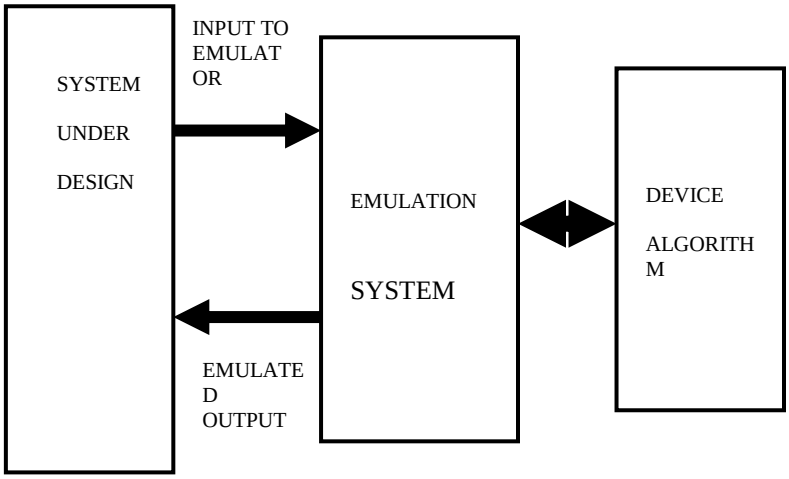

Figure 5.2: Block Diagram

Our focus here is the emulation system itself. Putting out this section from the diagram above, we have:

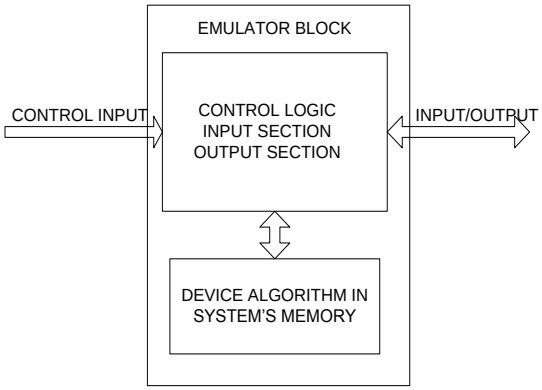

Figure 5.3: Emulator Block

As illustrated in figure 5.3, the emulation block is made up of two basic parts: the device algorithm which is, in reality, embedded in the emulator's memory, the processing unit which contains the control logic, the input section, and the output section. Below is the expansion of how the emulator is configured to include the various parts.

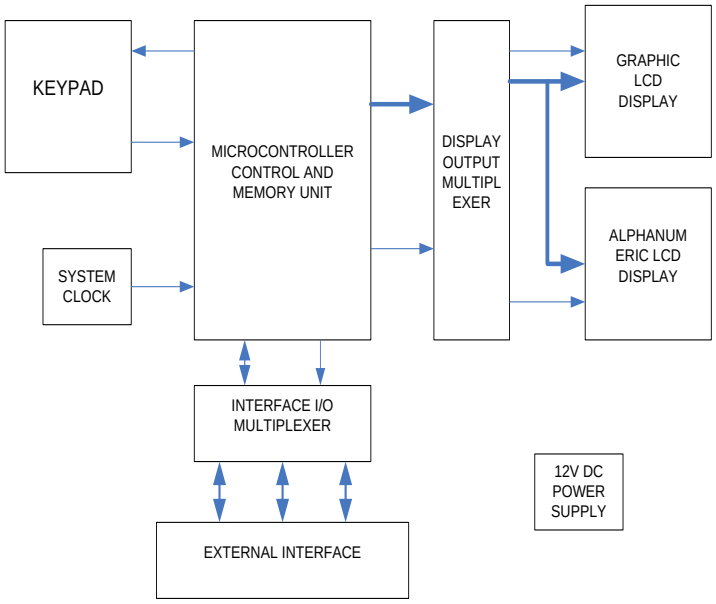

Figure 5.4: The Expanded Emulator lock

The Input Subsystem

The input subsystem is used to enter control commands into the system. The commands required are:

- The "function code" of a desired device.
- Enter command. And
- Quit command.

The function code will be represented by three decimal digit numbers (xxx). This three digit number is allowed a value of 1 to 255. Hence the emulator can mimic a total number of 255 different logic functions of different devices. The code 001 will correspond to function1, 002 will correspond to function2, and so on. The ENTER and QUIT commands are single actions and can each be carried out with a press of a button. Therefore, considering the number from 0—9, and the two buttons required for enter and quit actions, a total of 12 buttons is then required for the input subsystem. These buttons can be provided by a 4x3 matrix keypad.

The Control Subsystem

The control subsystem is the emulator's brain. It requires a high processing capability with large enough dynamic memory for data manipulation, and a high capacity ROM for its functional algorithm database. This unit also needs to have input/output connections through which it can communicate with the rest of the systems units as well as the external interface.

We may choose any microprocessor as the central processing unit (CPU), a random access memory (RAM) as dynamic memory, any ROM (read-only memory) for the database implementation, and finally add input/output connections. However, there are microcontroller chip available which already implement the entire scheme in a single package. Therefore the use of a microcontroller-based design is highly recommended.

The *Central Processing Unit (CPU)* oversees everything that the controller does. The CPU is a processor with associated circuits that controls the running of the controller software program. Basically, the CPU obtains (fetches) each program instruction from memory and carries out (executes) the instruction. After completing one instruction, the CPU moves on to the next one and in most cases can operate on more than one instruction at the same time. This "fetch and execute" process is repeated until all of the instructions in a specific program have been executed.

The Output Subsystem

The output subsystem of the emulation system is a display unit on which the status and the information required for the operation of the system are to be viewed. There are several display devices that can be used. These displays however, are either LED (in the form of 7-segment displays, etc), liquid crystal displays LCD, or cathode ray tube (CRT). In order to meet up with the requirement of this type of emulator, a graphic LCD and a character LCD has both been used. The graphic LCD is used to pictorially show configuration and parts of a selected device as well as its labeling. The high volume of text required to communicate with the user demands that an alphanumeric LCD be used. This character LCD provides us with a more flexible way of manipulating text as compared to the graphic LCD.

Each of these display devices has several input/output pins that utilize approximately two I/O ports from the controller. When directly connected to the controller, there will be no I/O lines left for other connections. Hence, it is required that the two display devices be multiplexed together to approximately one port utilizing few lines of control from other ports.

The Input/Output (I/O) Subsystem

The I/O subsystem is an interface system through which the emulator communicates with external circuitry. It serves to provide all pin connections that replace the input and/or output pins of a device being emulated. In order to cater for all possible devices to be emulated, 36 I/O pins has been specified. This number of I/O lines is more than what one 8051 based microcontroller can support. The 8051 controllers can only provide a total 32 I/Os. From the result of our input subsystem, 8 I/O lines will be required; and the data lines of the output subsystem also require 8 I/O lines. If we dedicate one port (i.e. 8 I/O lines) for signal control, then we are left with only 8 I/O lines which do not meet the 36 I/O lines needed for the interface subsystem. The only alternative to solving this problem is

multiplexing. Hence, utilizing some control signal lines from the control subsystem and with the addition of a multiplexing circuit, we can multiply these 8 I/O lines from the controller to any desired number of I/O lines.

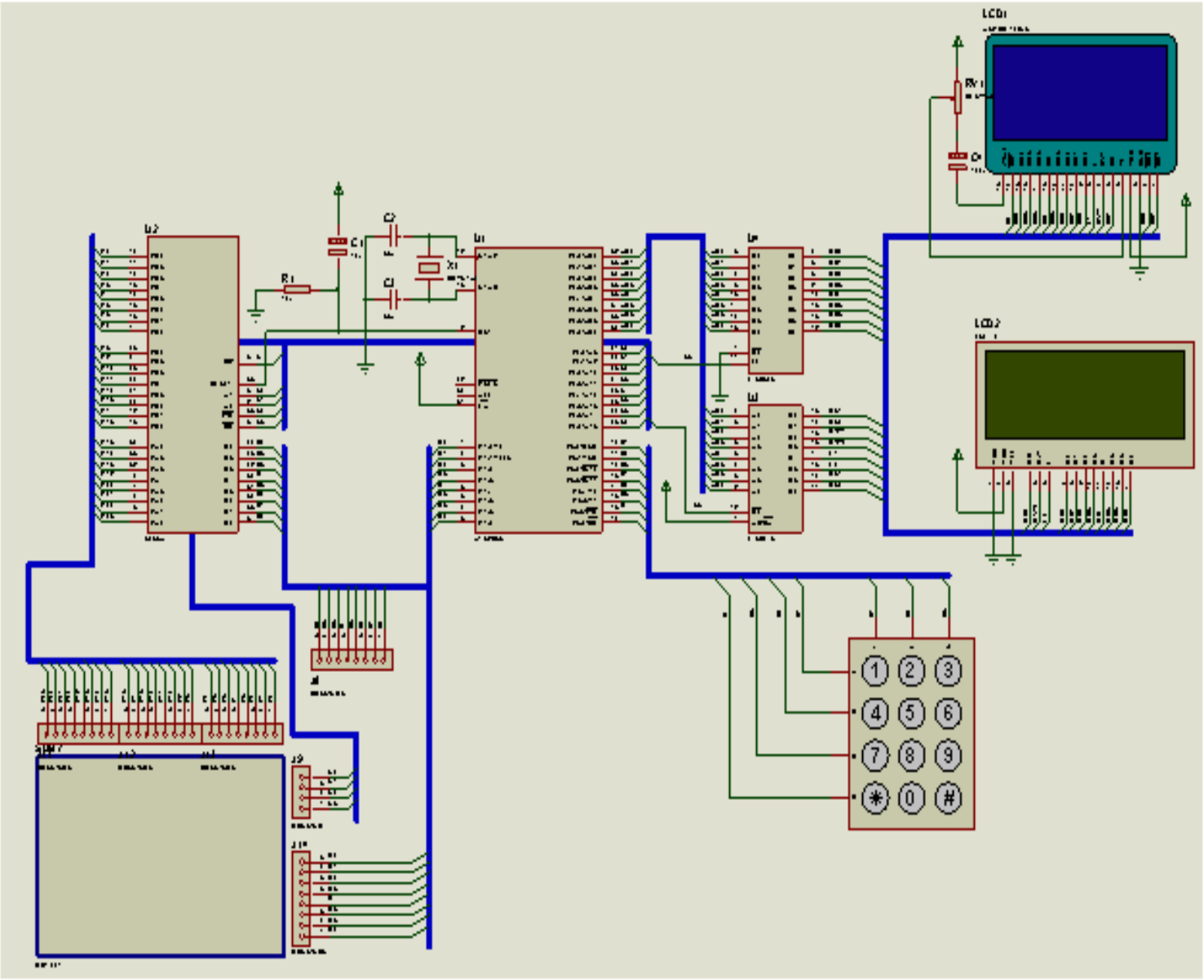

Figure 5.5: The Emulator's Circuit Diagram

The Control Algorithm

The various functions to be emulated by the emulator are carried out by a general system software program embedded in the memory of the microcontroller. This software program acts as the emulators operating system (OS) coordinating the various operations to perform any required function needed. This general coordinating software system is described in the algorithmic flow chart of figure 5.6.

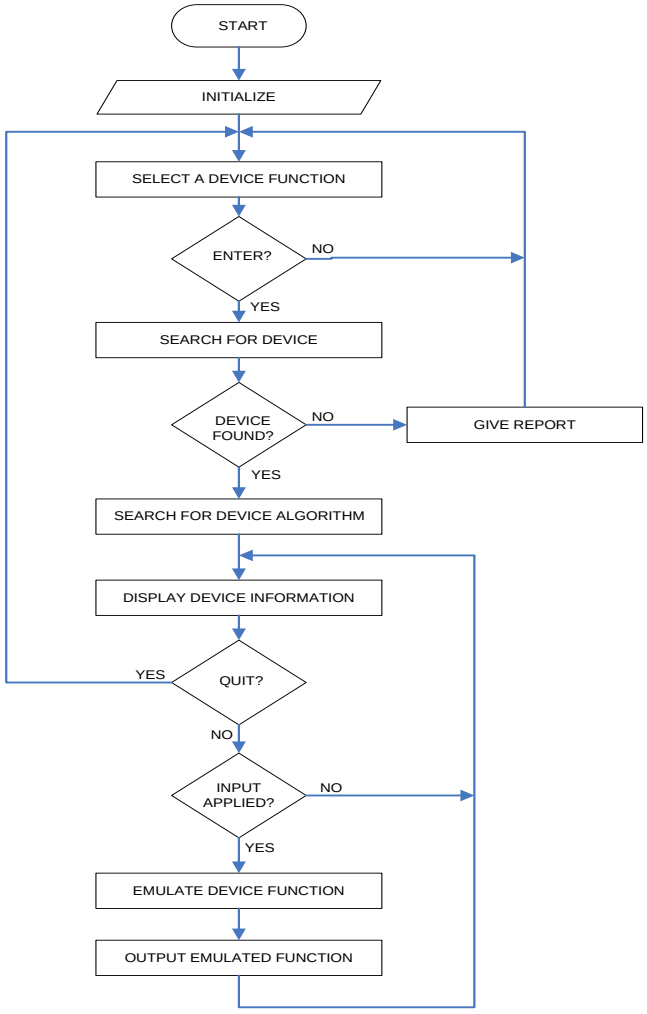

Figure 5.6: The Emulators Control Algorithm

6. DISCUSSION

Generally, several logic gates and structured logic devices may be required to carry out experimental designs in the lab, and in their numbers to support and serve the increasing population of students flooding our technological schools today. There will always be a time lag before supply is made, and insufficiency of the required devices may result. Cost of lab equipment provision and maintenance will also be on the high side. The most virile solution to this is to use logic emulators, such as have been described in this paper, to provide faster and low cost solutions.

7. CONCLUSION

Logic emulators as presented in this paper, provide a very durable, robust and flexible solution to just-in-time real-time development platform for digital systems developers as well as state-of-the art provision of laboratory equipments for technology transfer especially in developing countries.

REFERENCES

[1]. R. Turner, (1999).“System-level verification—a comparison of approaches,” in Proc. 10th International Workshop on Rapid System Prototyping (RSP ’99), pp. 154–159, Clearwater, Fla, USA.

[2]. Nikolaos Mitas. (2008). Design Partitioning for Custom Hardware Emulation. MSc. Thesis,Computer Engineering, Mekelweg 4, 2628 CD Delft, The Netherlands.

[3]. A Agarwal, J. Babb, and R. Tessier. (1993). “Virtual wires: overcoming pin limitations in FPGA- based logic emulators,” Proc. IEEE Workshop on FPGAs for Custom Computing Machines, pp. 142-151.

[4]. Scott Hauck, Gaetano Borriello, Carl Ebeling Springbok. (1994). A Rapid-Prototyping System for Board-Level Designs Department of Computer Science and Engineering University of Washington Seattle, WA 98195.

[5]. Bill Novak. (2002). XDS560 Emulation Technology Brings Real-Time Debugging Visibility to Next-Generation High-Speed Systems. Texas Instruments Post Office Box 655303 Dallas, Texas 75265

[6]. Baneres D., Cortadella J., Kishinevsky M. A Recursive Paradigm to Solve Boolean Relations // Proceedings of Design Automation Conference, P. 416 - 421.

[7]. Richard J. (2001). Discrete Mathematics, Prentice Hall 2001, 621 p.

[8]. Voros N.S., Sanchez L., Alonso A., Birbas A.N., Birb M., Jerraya A. (2003). Hardware-Software Co-Design of Complex Embedded Systems. Design Automation for Embedded Systems.– Boston: Kluwer Academic Publishers. P.5-34.

[9]. Stephen Brown, Zvonko Vranesic. (2009) Fundamentals of Digital Logic with VHDL Design, 2nd Edition, McGraw Hill.

[10]. Semiconductor Industry Association. (2009). National Technology Roadmap for Semiconductors. Retrieved from http://www.semichips.org/

[11]. Chen Chang, Kimmo Kuusilinna, Brian Richards, Robert W. Brodersen. Implementation of BEE: a Real-time Large-scale Hardware Emulation EngineUniversity of California, Berkeley Wireless Research Center 2108 Allston Way, Berkeley, CA 94704