NIPES

Journal of Science and Technology Research

Journal homepage: www.nipesjournals.org.ng

Emulation Technique in Digital Systems Design

Ifeyinwa Obiora-Dimson[1], Hyacinth Inyiama[2], John Adejumo[3*]
[1]Electronic and Computer Engineering Department, Nnamdi Azikiwe University, Awka, Anambra State, Nigeria.
[2]Enugu State University of Technology, Enugu, Enugu State, Nigeria.
[3]Electronic and Computer Engineering Depertment, Nnamdi Azikiwe University, Awka, Anambra State Nigeria.
Email: ifeyinwaodimson@gmail.com, drhcinyiama@gmail.com, *Johnadejumo93@gmail.com

Article Info

***Keywords*:**
Emulation Software, Emulator, Digital system design, logic table

Received 9 February 2023
Revised 2 March 2023
Accepted 2 March 2023
Available online 22 March 2023

https://doi.org/10.5281/zenodo.7760471

Abstract

The need for emulation as a technique in digital systems design was established in this paper. Three scenarios where emulation can be applied were stated. To successfully emulate, an emulator and emulation software is of the essence. The emulators identified for use include a microcontroller and Read Only Memory (ROM) while the emulation software is crafted from the State Transition Table (STT) /the fully expanded STT or the logic table of the control system or device being emulated. For a device or control system to be emulated its characteristics must be converted to its resultant STT, logic table, or truth table which is used to craft the emulation software. The strength, drawback, and when more appropriate to deploy any of the emulators and their corresponding emulation software was also discussed.

1.0 Introduction

Tr In digital systems design, emulation technique is of great importance. As the name implies it simply means imitating the function of a device. In digital systems design, a device or software or both can be used to emulate another device. The emulation device is often referred to as an emulator [1] while the software is usually called emulation software [2]. Emulation is of necessity in control systems design because it can be used where there are: Non-availability of relevant components, Need to reduce the design complexity and Need to reduce component count.
In this paper, ROM or microcontroller shall be used and referred to as an emulator while the emulation software can be crafted from an appropriate State Transition Table (STT)/logic table or truth table as the case may require.
Also, three scenarios where emulation can be applied are presented and they are: Microcontroller with STT Based Software emulation, ROM with fully expanded STT Based Software emulation, ROM or Microcontroller with truth table-based software emulation (for logic gates).
For the three scenarios mentioned it should be noted that the system to be emulated must have its design requirement tailored to a STT or its truth table (for logic gates) before it can be emulated. Details of each scenario shall be discussed in the following sections.

1.1 Literature review

Inyiama et al described the use of ASM chart and its corresponding STT in the design of digital control systems [3]. Also, how these were implemented using ROM [4] and Micros [5] was also

developed. It was established that any digital control system whose design can be tailored to an ASM can be realized following well-defined design steps.
The trend in digital systems is the production of robust but miniature devices, some for embedded purposes.[6] To achieve this some components are factory manufactured to given specifications to have them match the design space. This could be costly as it is a customized component. But where off-the-shelf components which are readily available in the market can be used to emulate the custom-made device, then that path should be followed. There is need also to reduce the complexity in design and also component count thus the need for emulation. The three scenarios where emulation can be applied in digital systems design are discussed hereunder.

2. Scenario 1: Microcontroller with STT-Based Software emulation.

A control system can be emulated using a microcontroller as an emulator and adapting the STT for use to craft the emulation software. This design involves developing the specifications of the envisaged system using an ASM chart, converting it to its equivalent STT, and then to software implementation of the STT in a microcontroller. The emulation software is crafted directly from the state transition table by writing software statements that match rows of the STT which is programmed into the memory of the microcontroller. Other processing/interfacing required by the system being emulated is also carried out. Therefore, when the specifications of a control system have been successfully tailored to an ASM chart, and its equivalent STT crafted into the software, which is then programmed into a microcontroller and the system is working as expected, we say that the control system has been successfully emulated.

2.1 Microcontroller with STT-Based Software emulation Design Procedure

The following steps are prescribed to be followed to achieve a STT-based control emulation software design for use in a microcontroller. They include:

a) Put down the specifications for the envisaged system,
b) Develop the corresponding Algorithmic State Machine (ASM) chart,
c) Derive the STT corresponding to the ASM chart.
d) Reduce input/output line demand of the application using input multiplexing and output decoding if need be.[10]
e) Modify STT by placing the State Code column before the Qualifiers' column.
f) Write software to implement rows of the STT.
g) Fit the emulation software into the microcontroller.
h) Apply necessary interface devices where applicable.

3. Scenario 2:ROM with fully expanded STT Based emulation

This scenario presents an emulation of control systems designs using a ROM device as the emulator and an emulation software crafted from a fully expanded STT. The design, therefore, stems from ASM chart of the envisaged control system to STT to Fully expanded STT and implementation using a ROM. In [7, 3], the external and internal ROM architectures have been looked into. Of great importance for the emulation is the internal architecture of the ROM which is obtained from a fully expanded State Transition Table. [3] The STT comprises the input qualifiers, the present state codes, the next state code, and the state outputs. [8, 9] To effectively embed the details of an STT into a ROM, one must first fully expand the STT. Next is to concatenate the input qualifiers and the present state codes to serve as the address in the ROM. The next state code and the state outputs both combined, form the content of that ROM location. Therefore, the input address and the memory

content are both converted to HEX on the flat table before burning into the ROM as shown in Table 1. Thus, this hex file programmed into the ROM mimics the behavior of the control system and is therefore referred to as ROM-based emulation.

Table 1: Fully expanded State transition table of a typical control system [11]

Link path	Address in Hex	Qualifier (inputs) CDS	CPS	CFS	Present state name	Present state code B	A	Next state name	Next state code B'	A'	State outputs		ROM Input in Hex
L1	00	0	0	0	ST0	0	0	ST0	0	0	1	0	02
	04	0	0	1	ST0	0	0	ST0	0	0	1	0	02
	08	0	1	0	ST0	0	0	ST0	0	0	1	0	02
	0C	0	1	1	ST0	0	0	ST0	0	0	1	0	02
L2	10	1	0	0	ST0	0	0	ST1	0	1	1	0	06
	14	1	0	1	ST0	0	0	ST1	0	1	1	0	06
	18	1	1	0	ST0	0	0	ST1	0	1	1	0	06
	1C	1	1	1	ST0	0	0	ST1	0	1	1	0	06
L3	01	0	0	0	ST1	0	1	ST1	0	1	0	0	04
	05	0	0	1	ST1	0	1	ST1	0	1	0	0	04
	11	1	0	0	ST1	0	1	ST1	0	1	0	0	04
	15	1	0	1	ST1	0	1	ST1	0	1	0	0	04
L4	09	0	1	0	ST1	0	1	ST2	1	1	0	0	1C
	0D	0	1	1	ST1	0	1	ST2	1	1	0	0	1C
	19	1	1	0	ST1	0	1	ST2	1	1	0	0	1C
	1D	1	1	1	ST1	0	1	ST2	1	1	0	0	1C
L5	03	0	0	0	ST2	1	1	ST2	1	1	0	1	3D
	17	0	0	1	ST2	1	1	ST2	1	1	0	1	3D
	13	1	0	0	ST2	1	1	ST2	1	1	0	1	3D
	17	1	0	1	ST2	1	1	ST2	1	1	0	1	3D
L6	07	0	0	1	ST2	1	1	ST3	1	0	0	1	39
	08	0	1	1	ST2	1	1	ST3	1	0	0	1	39
	17	1	0	1	ST2	1	1	ST3	1	0	0	1	39
	1F	1	1	1	ST2	1	1	ST3	1	0	0	1	39
L7	1A	0	1	0	ST3	1	0	ST3	1	0	0	0	28
	0E	0	1	1	ST3	1	0	ST3	1	0	0	0	28
	1A	1	1	0	ST3	1	0	ST3	1	0	0	0	28
	1D	1	1	1	ST3	1	0	ST3	1	0	0	0	28
L8	02	0	0	0	ST3	1	0	ST0	0	0	0	0	20
	06	0	0	1	ST3	1	0	ST0	0	0	0	0	20
	12	1	0	0	ST3	1	0	ST0	0	0	0	0	20
	16	1	0	1	ST3	1	0	ST0	0	0	0	0	20

This simply implies that when an address (input qualifiers +present state codes) is called up, the content (next state code +the state output) of that memory location is released. Thus, a control system whose design specifications can be translated into an ASM chart and then STT and also the Fully Expanded STT (FESTT) derived can be emulated using a ROM as the emulator.

3.1 ROM emulator with FESTT design procedure

ROM-based emulation is most important where inputs only are required to generate an output. Thus to emulate successfully using a ROM, the following steps must be followed.

a) Translate the characteristics of the control system to be emulated into an ASM chart
b) Derive its STT from the ASM chart.
c) Fully expand the table by adding logic zeros or ones where appropriate, to obtain the FESTT.

d) The input variables serve as address into the ROM and is converted to Hex.
e) The output variables serves as the ROM content and is converted to Hex
f) Burn these hex files into a ROM.
g) Attach other peripheral devices needed for the system to function well.

4. Scenario 3: Truth Table-Based Emulation (for logic gates).

Many logic gates can be emulated either with the use of a ROM or a Microcontroller as the emulator, while the truth table of those logic gates is used to craft the emulation software. Where the logic gates to be emulated simply involves input and output, the ROM is sufficient for use as an emulator. But where the emulation would require a keypad or other processing such as decoded output for some of the gates to be emulated, or the use of an LCD, then a microcontroller is a better choice.

The emulation software for the logic gates to be emulated is crafted from the truth tables of the gates. This is achieved by converting the expected input into hex file and the expected output also into hex file. These are programed into the ROM or memory of the microcontroller and serves as the emulation software. Other peripheral devices and programming required (for the microcontroller) are completed for the emulator to function as required.

5.1 Design steps of emulation for logic gates

The following steps are suggested to guide a developer to successfully emulate logic gates using a ROM or a microcontroller.

a) Map out the logic gates to be emulated
b) Write out their truth tables
c) Collate the truth tables into a single table
d) Convert the input patterns per row into its hex equivalent
e) Convert the output patterns per row into their hex equivalent
f) Program the hex patterns into ROM. For ROM, the input patterns form the address of ROM locations while the output patterns become the content of those locations. Alternatively, program the hex patterns into the memory of the microcontroller if it is the emulator of choice.
g) Carry out other necessary programming required (for the microcontroller)
h) Attached are all peripheral devices needed for the emulator to work
i) Test the system

5. Evaluation and discussion

To realize an emulation software from an STT, a fully expanded STT may be necessary so as to eliminate the dashes (-), replacing them with binary combinations whenever a ROM is to be used as an emulator. A dash entry in an STT cannot be programmed into a ROM, each combination of dashes would have to be replaced by the full binary combinations that are possible. Thus, each row in an STT with two dashes (- -) needs to be replicated four times in each row where the two dashes are represented either as 00 or 01 or 10 or 11. Similarly, any row with 3 dashes leads to 8 replications in each of which the 3 dashes are represented as either 000 or 001 or 010 or 011 or 100 or 101 or 110 or 111, and so on. Thus, for a fully expanded STT needed for a ROM-based emulation more rows than the rows in the original STT would be programmed. This is called combinatorial explosion, which is a drawback for ROM-based designs, where unwanted combinations that contribute nothing to the logic would have to be represented in an exhaustive decoding process associated with ROMs.

Contrast this with microcontroller-based emulation where there are software constructs that can be used to represent the rows in a STT without decoding the dashes. Thus, the microcontroller program height would be much shorter than for the corresponding ROM height.

An area of application of emulation for logic gates is in the development of logic trainers. With the steps outlined in section 5.1, several logic gates can be emulated using a single ROM chip or a microcontroller with the necessary interfacing devices to produce a logic trainer.
Also a single ROM or microcontroller can be used to emulate several control systems. Once the necessary input is applied, the control system is executed and the output released. Thus, for control systems requiring input to generate an output, this is achievable in ROM or microcontroller-based emulation.

6. Conclusion

Three scenarios for emulation in digital systems design has been presented in the foregoing. Emulation is possible with an emulator, emulation software or both. The use of Microcontrollers or ROM as emulators with STT based emulation software or truth table-based emulation software has been discussed. The design steps necessary for developing a digital system design using any of the scenarios have also been presented. The scenarios presented were appraised to show clearly where each is best suited and applied.

References

[1] Gowda, Megha & T G, Namratha Shetty & Kumar e, Pavan. (2020). Design Optimization in Digital System Design. 7. 1884-1889.
[2] Roy, Shirshendu. (2021). Advanced Digital System Design - A Practical Guide to Verilog Based FPGA and ASIC Implementation.
[3] Ifeyinwa C. Obiora-Dimson, Hight C. Inyiama, Omijeh Bourdillon Omijeh. “Re-Engineering Complex Process Control Systems Using Sub-Process Agents.” International journal of engineering research and applications (IJERA) 2017 vol 7 issue 4 pp 53-61
[4] Hyacinth C. Inyiama, Ifeyinwa Obiora-Dimson, Christiana C. Okezie. “Designing Multi-Agent Based Linked State Machine”. International journal of research in engineering and technology (IJRET). Vol 2 issue 7 July 2013. http://www.ijret.org
[5] Hyacinth C. Inyiama, Ifeyinwa Obiora-Dimson, Christiana C. Okezie. “State Transition Table Based Control Software Engineering For Micros”. International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 3, March – 2013.
[6] Khoirom Johnson Singh, Tripurari Sharom, Huirem Tarunkumar. “High speed and low-power basic digital logic gates, a half adder and full adder using modified gate diffusion input technology”. Journal of VLSI design tools and technology (JoVDIT) 2018 vol 8 issue 1, 34-42
[7] Uzedhe, G. & Inyiama, C & Chidiebele, Udeze & Ekene, Mbonu. (2013). “Microcontroller Based Real-Time Emulator for Logic Gate and Structured Logic Devices”. International Journal of Science and Technology. 2. 639-647.
[8] Winograd, T., Shenoy, G., Salmani, H., Mahmoodi, H., Rafatirad, S., & Homayoun, H. (2018). “Programmable Gates Using Hybrid CMOS-STT Design to Prevent IC Reverse Engineering”. ACM Transactions on Design Automation of Electronic Systems (TODAES), 23, 1 - 21.
[9] Salah, Wael & Zneid, Basem. (2019). “Evolution of Microcontroller-based Remote Monitoring System Applications.”International Journal of Electrical and Computer Engineering. 9. 2354-2364. 10.11591/ijece.v9i4.pp2354-2364.
[10] Inyiama Hyacinth C.; Okezie Christiana C.; Okafo Ifeyinwa C, (2013). “Complexity Reduction in Rom-Based Process Control Systems via Input Multiplexing And Output Decoding”. International Journal of Research and Advancement in Engineering Science, Vol 3 No 1. 2013.
[11] H.C. Inyiama and I.C. Obiora-Dimson, Designing Real-Time Digital Control Systems. Salvation press Enugu, Nigeria. 2022. pp 87.