iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VII **Month of publication:** July 2022

DOI: https://doi.org/10.22214/ijraset.2022.44711

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

A Software System for A Finite State Machine (FSM)

Mr. Rupak Kumar Gogoi[1], Mr. Abinash Borah[2], Ms Chandrani Borah[3]
Asstt. Professor, Department of Computer Application, Jorhat Engineering College.

Abstract: ***A finite-state machine (FSM) is a computational mathematical model. For design and analysis, circuits and system operations can be represented in a variety of ways. FSM is one of the methods for using a drawing to represent the operations of many circuits and systems in electronic engineering, computer engineering, and so on. In terms of design, the finite state machine is a very simple machine. It is made up of a set of input symbols, output symbols, and states that must be designed. Furthermore, a function of input and output symbols with the current state to give the next state must be present. To simulate the Finite State machine, a software simulator is implemented in this paper using the Visual Basic programming language (in terms of design and operation).Several general examples are represented in this model. However, the software can be used to teach students about FSM and how it works.***

I. INTRODUCTION

The concept of Finite State Machine is one application of directed graph that is more useful than non directed graph. The directed graph, denoted by "D," consists of two things: .

1) A 'V'-shaped set whose elements are known as vertices, points, or nodes.
2) Arcs are a set 'E' of ordered pairs of vertices.

A digital computer can be viewed as a machine that is in a "internal state" at any given time.
The computer "reads" an input symbol before "printing" an output symbol and changing its "state."
The output symbol is solely determined by the input symbol and the internal state of the machine, and the internal state of the machine is solely determined by the previous state of the machine and the preceding input symbol.It is assumed that the number of states, input symbols, and output symbols is finite.
A finite state machine can be defined and used to describe the

II. THEORY OF FSM

FSM is made up of states and state-in-actions. Researchers(3) classified FSM models into two types based on the number of output states and predictability of output. These are referred to as non-deterministic FSM and deterministic FSM, respectively. There are also other classifications, such as output state dependence. A FSM's direction of state-transition routes is another distinguishing feature. FSM optimization to determine the smallest number of states with the same function is a current research topic.
A finite state machine (FSM) is a model or behaviour that is made up of a limited number of states, transitions between those states, and actions. A finite state machine is an abstract machine model with primitive internal memory. The concept of FSM is illustrated with an example of a logic function that depends on its input history to determine its output, so the entire process of transforming a specific function is viewed as a variety of equivalent representation, resulting in an actual implementation of gates and flip-flops.
Generally An FSM is denoted as M and is 5 tuple'

1) A finite set of input symbols.
2) A finite set S of internal state.
3) A finite set Z of output symbols.
4) A next state function F from S X A—>S.
5) An output function g from S X A—> Z.

The technique for designing a finite state machine can now be addressed as follows:

a) *Step1:* Because FSM is frequently stated in terms of English-language particular to its behaviour in order to comprehend the problem, it is critical that you interpret this explanation clearly. Try some input sequences for FSM to ensure that you understand the conditions under which the various outputs are produced.

b) *Step2:* FSM's abstract representation is obtained. Once you've grasped the problem, you'll need to put it into a format that the FSM processes can easily handle.A state diagram is a specification in hardware description languages that represents an algorithmic state machine.State diagrams are a graphical representation of an FSM's design.

c) Step3: It performs a state minimization step before deriving the abstract representation, which frequently results in a description with too many states paths through the state machine that can be eliminated because their input/output behaviours are duplicated by other functionally equivalent paths, and this is a new step that isn't required in the simpler counter design process.

d) Step4: It conducts state assignment in the opposite direction of the state, and the output was identical, so there was no need to worry about encoding a specific state output that is derived from the bit stored in state flip-flops (plus the input).

e) Step5: To choose the types of flip-flops to implement the FSM state, which is the same as the decision made during the counter design procedure.

In a nutshell, a Finite State Machine (FSM) is a model of behaviour made up of a finite number of states, transitions between them, and optional actions. The transition function, which is dependent on the input symbol, manages the transitions between states (event).

III. THE SOFTWARE'S MAIN WINDOWS

Fig. 1 shows the FSM software's main window (1). It is divided into two sections: "Enter data portion" and "display part":

1) *Part 1*: As illustrated in the diagram above, the input FSM table includes of edit boxes for entering data and processing buttons. These are the following:

- *Edit Boxes*

a) "Enter number of states in FSM": this edit box allows you to enter the number of states, for example (1,2,3,...). The number of states in this software is limited to one to five, so if the user enters 3, a massage box for correct entry will appear, as shown in fig. 2. A massage box error for erroneous entry will occur if the user enters a character or zero, as seen in fig.3.

b) "present state" edit box: this edit box is for entering the current state, such as q_0,q_1,...etc.

c) "input": this edit box is for entering input symbols like a,b, and so on.

d) "output": this edit box is for entering the output symbol, such as x,y, and so forth.

e) "next state" edit box: this edit box is for entering the next state, such as q_0,q_1,..., and so on.

- *Buttons:*

f) "Save current state": this option is for saving the raw data entered in the edit boxes described above. When the user presses this button, the data is saved as raw in the matrix within the software.

g) "enter next state": this button is for entering the above-mentioned raw data.

h) "see FSM": this button displays the data that has been input and saved in the matrix. (See Fig. 4 for further information.)

i) "draw FSM": this button displays the FSM digraph based on the information supplied. (As seen in Fig. 5)

j) "draw all FSM": this button is used to show some examples that have been saved in the software for one, two, three, and five states.

k) "*Exit*": this button is designed to exit from the software when the user wants to exit.

l) "Clear All": this button is deigned to clear all edit boxes and screens.

2) *Part 2:* "display part": this part is divided into two screens:

a) The first screen is used to show the FSM state table that has been saved in the matrix.

b) The FSM state digraph is displayed on the second screen. The process flowchart, on the other hand, is depicted in the fig.6.

IV. EXPERIMENTS AND RESULTS

1) *Example 1:* (7) If the number of states is set to 1, the input symbols are a, b, and the output symbols are x, y. Table 1 shows the state table for the function of the following state (1). Fig.1 shows a state diagram for one state (7). If the user runs the preceding example through the software, the results will look like this (8).

2) *Example 2:* (7)If the user enters the number of states = 5, the input symbols = a, b, and the output symbols = x, y. Table 1 shows the state table for the function of the next state (2) Figure shows a state diagram for one state (9).If the user runs the preceding example on the software, the results will be displayed as shown in fig.10.However, the user can apply the 2,3,4 states in the software and the results will be shown in figures 11, 12, 13 respectively.

V. CONCLUSION

1) In general, the five factors must be valid in order for FSM to represent any circuit or system. However, the input and/or output may be absent in some circuits and systems.
2) Any type of symbol can be used for input/output and state; binary numbers can also be used.
3) The graphic tools (functions) in the programming language are used to implement software. These tools will have an immediate impact on implementation and modification.On the other hand, the programmer should be able to use the tools with caution.
4) The number of (input, output) symbols and states is finite, but by tweaking the programme, it is possible to make it endless.
5) The programme that has been built can be utilised to teach students about FSM concepts and principles.
6) This software is easy to use and is regarded as a primary version of FSM-related integrated software.

Table (1): State table of example one.

Present State	Input	Output	Next State
q_0	a	x	q_0
q_0	b	y	q_0

Table (2): State table of example two.

present State	Input	Output	next State
q_0	a	y	q_1
q_1	b	x	q_0
q_3	a	x	q_1
q_2	b	x	q_2
q_0	a	y	q_2
q_3	a	x	q_4
q_3	b	y	q_2

Fig.1: The main window.

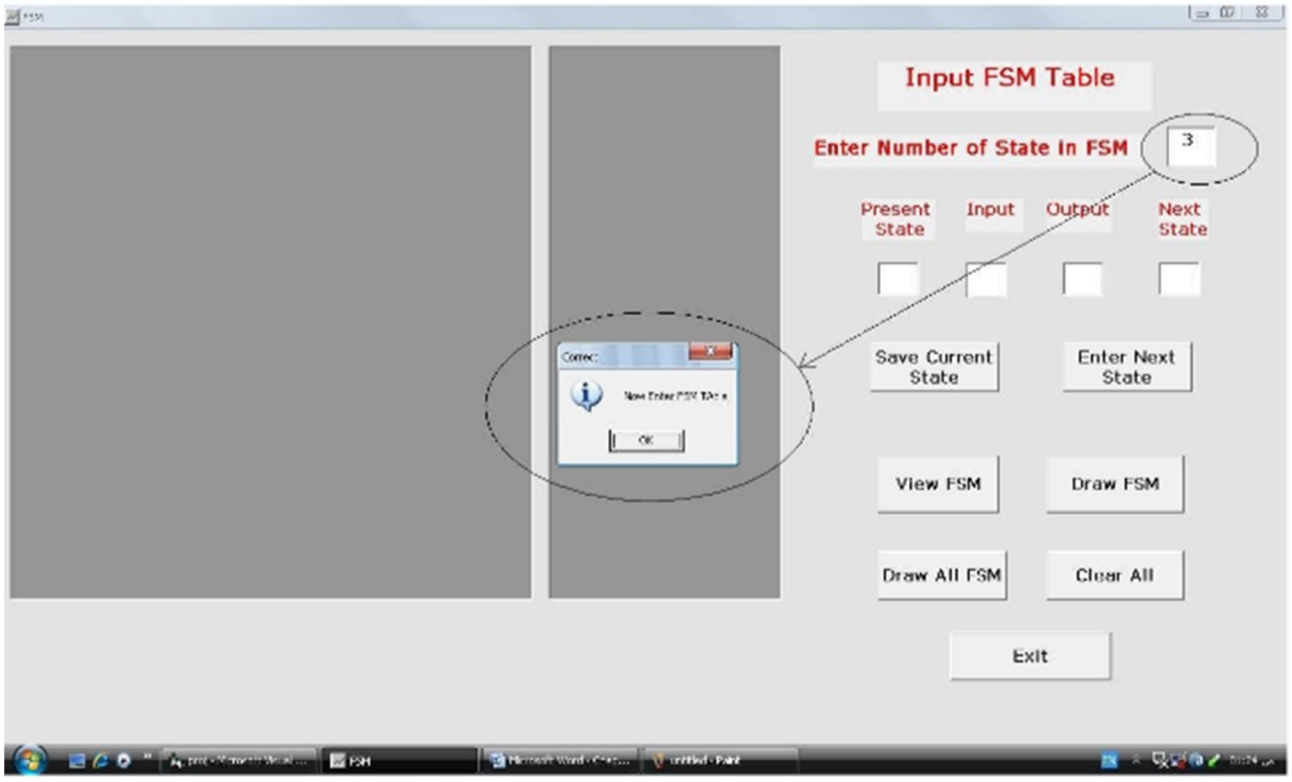


Fig.2: A massage box for correct enter.

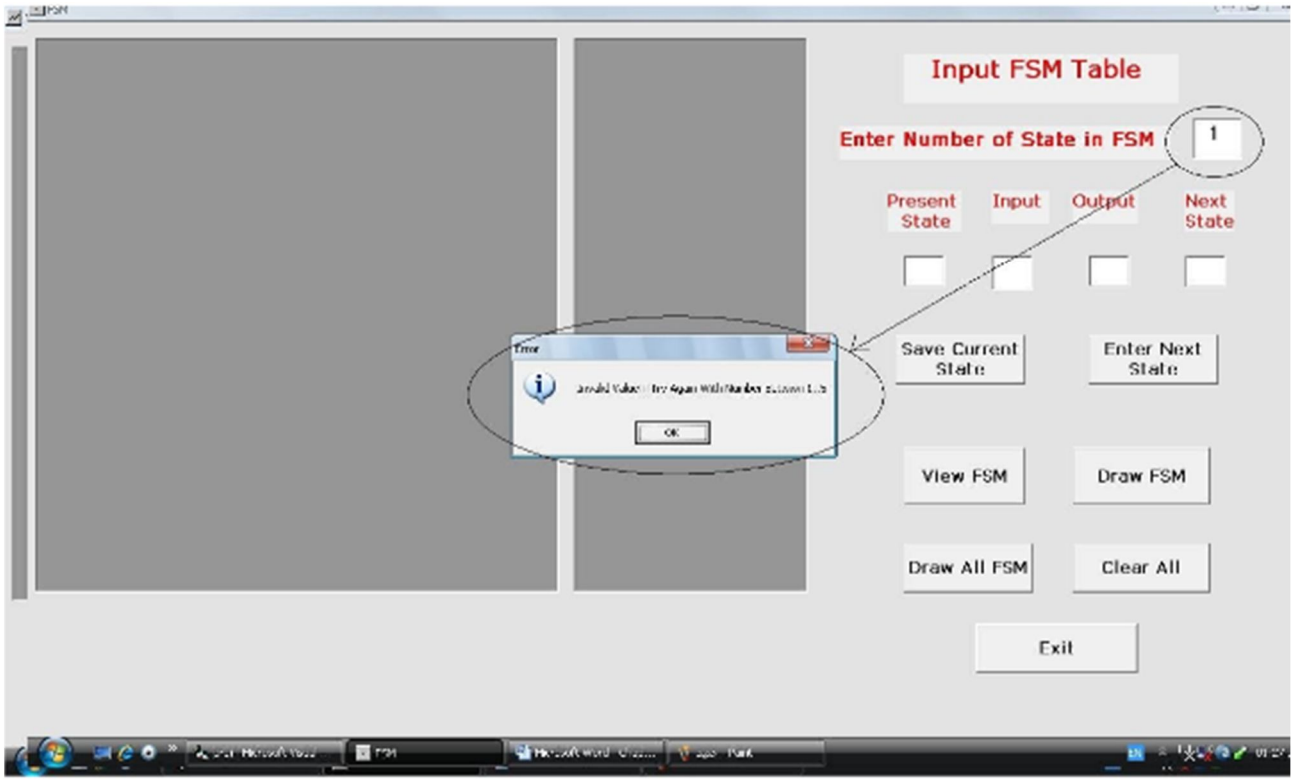


Fig.3: Error massage box for wrong enter.

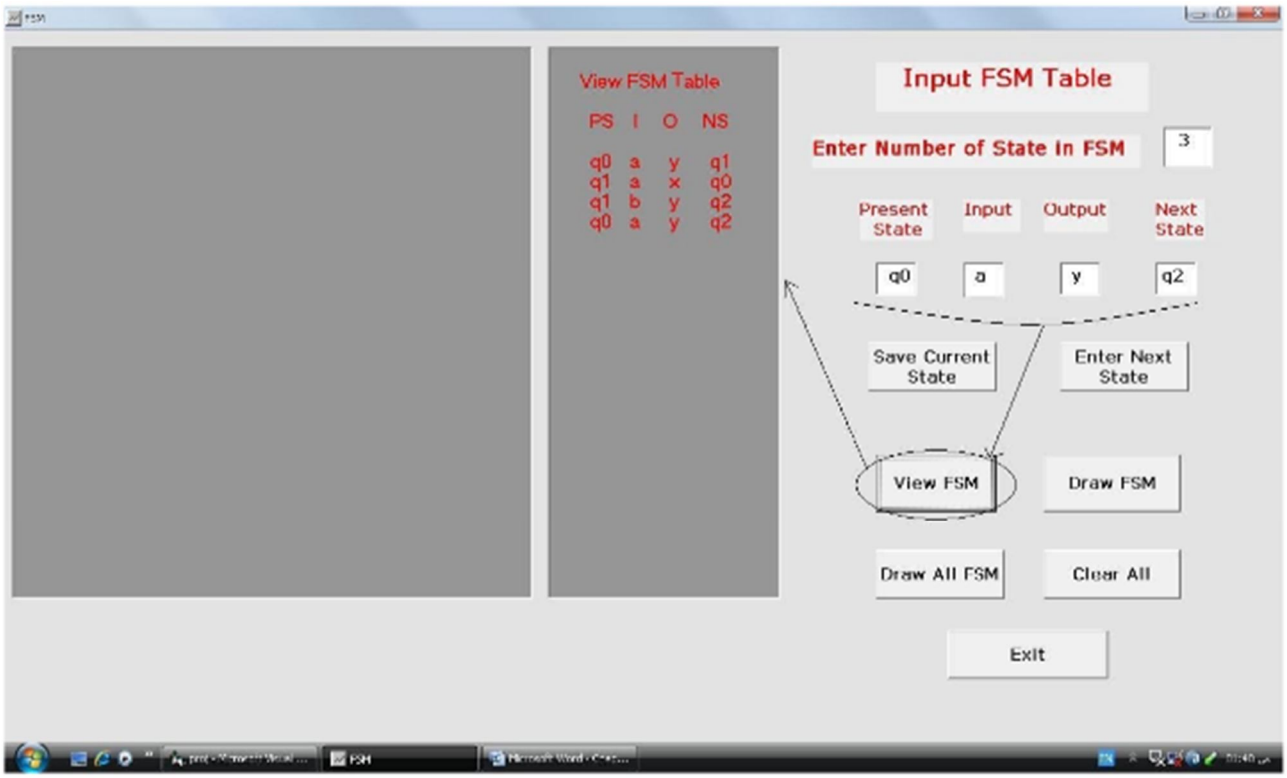


Fig.4: View FSM button.

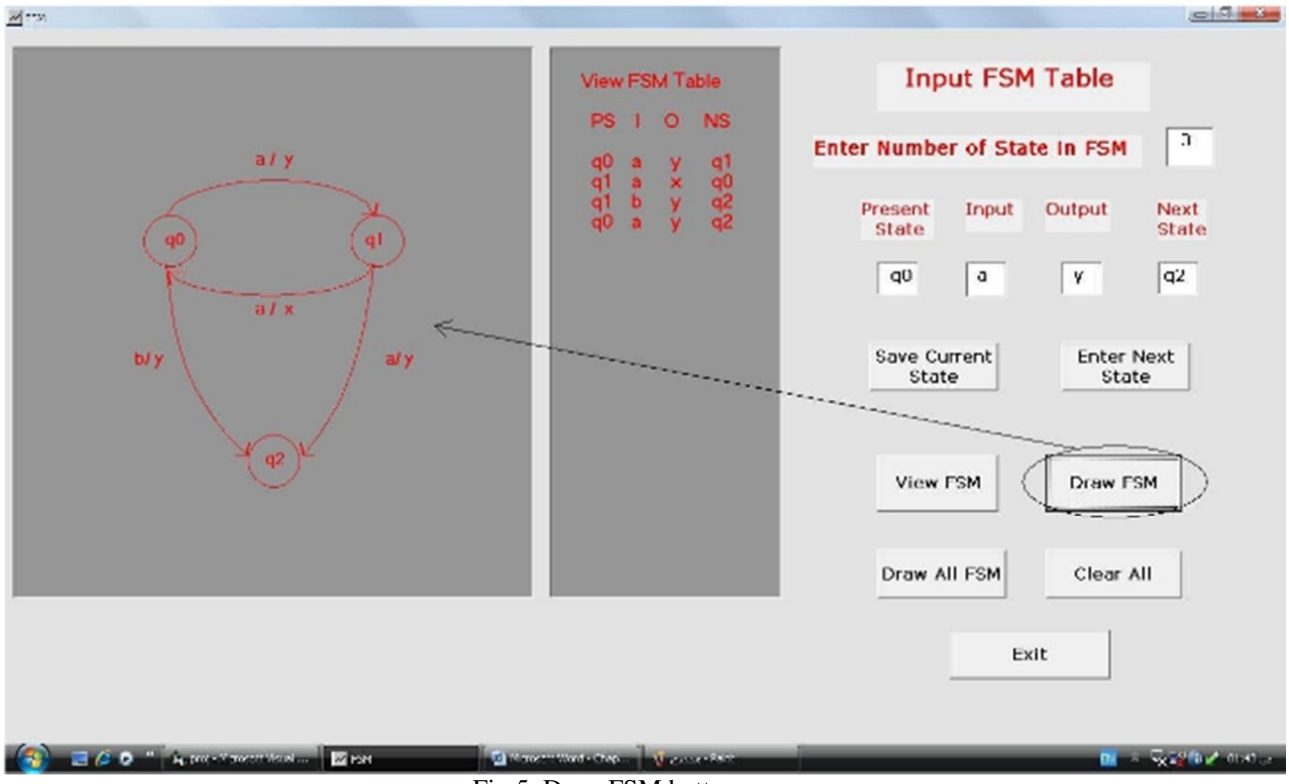


Fig.5: Draw FSM button.

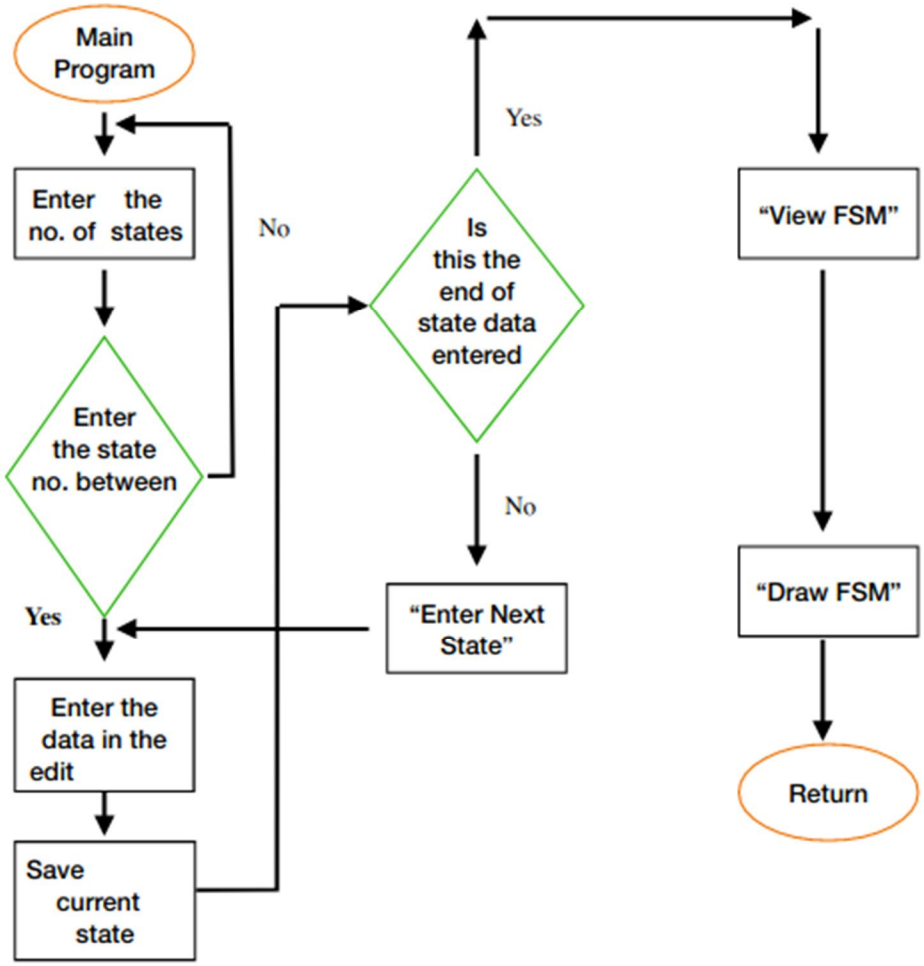


Fig.(6) :Flowchart of the FSM system.

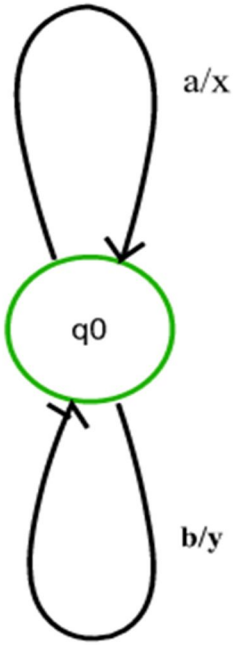


Fig.7: State diagram.

Fig.8: Draw FSM diagram.

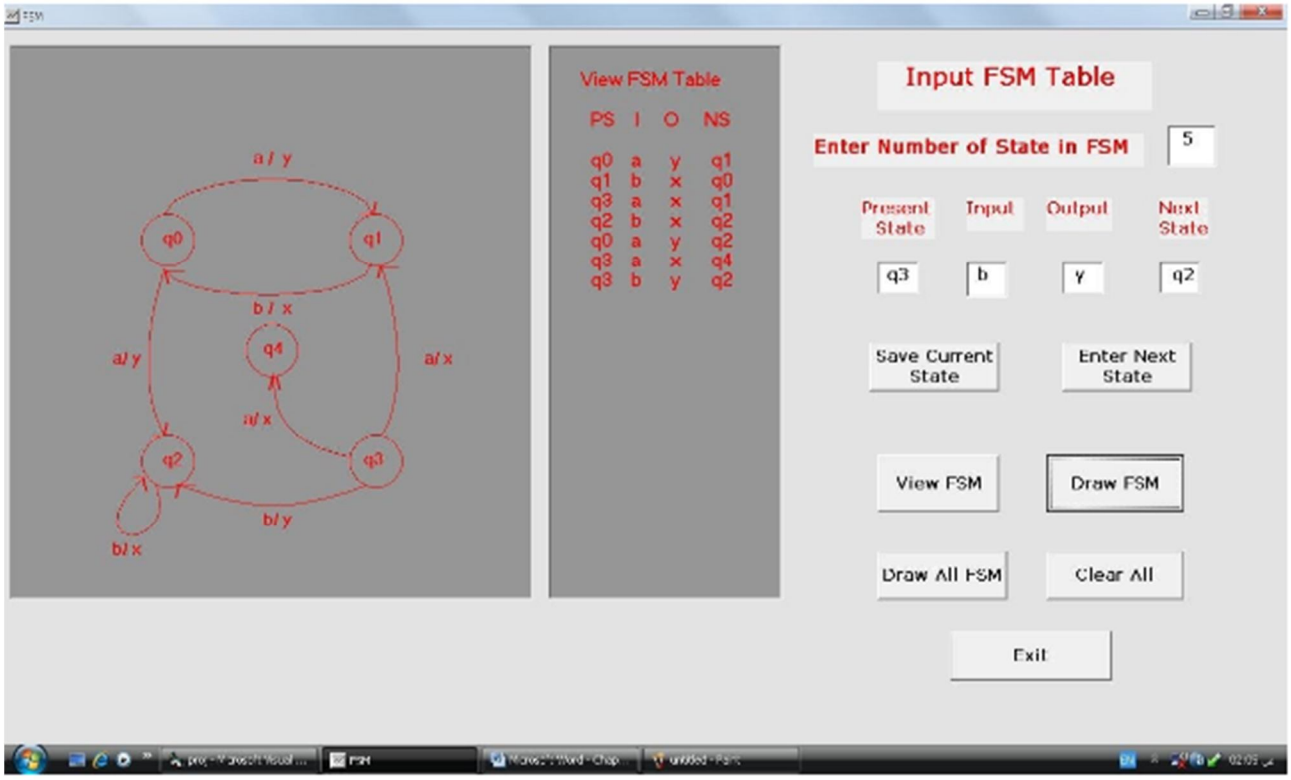


Fig.10: Applying of example two.

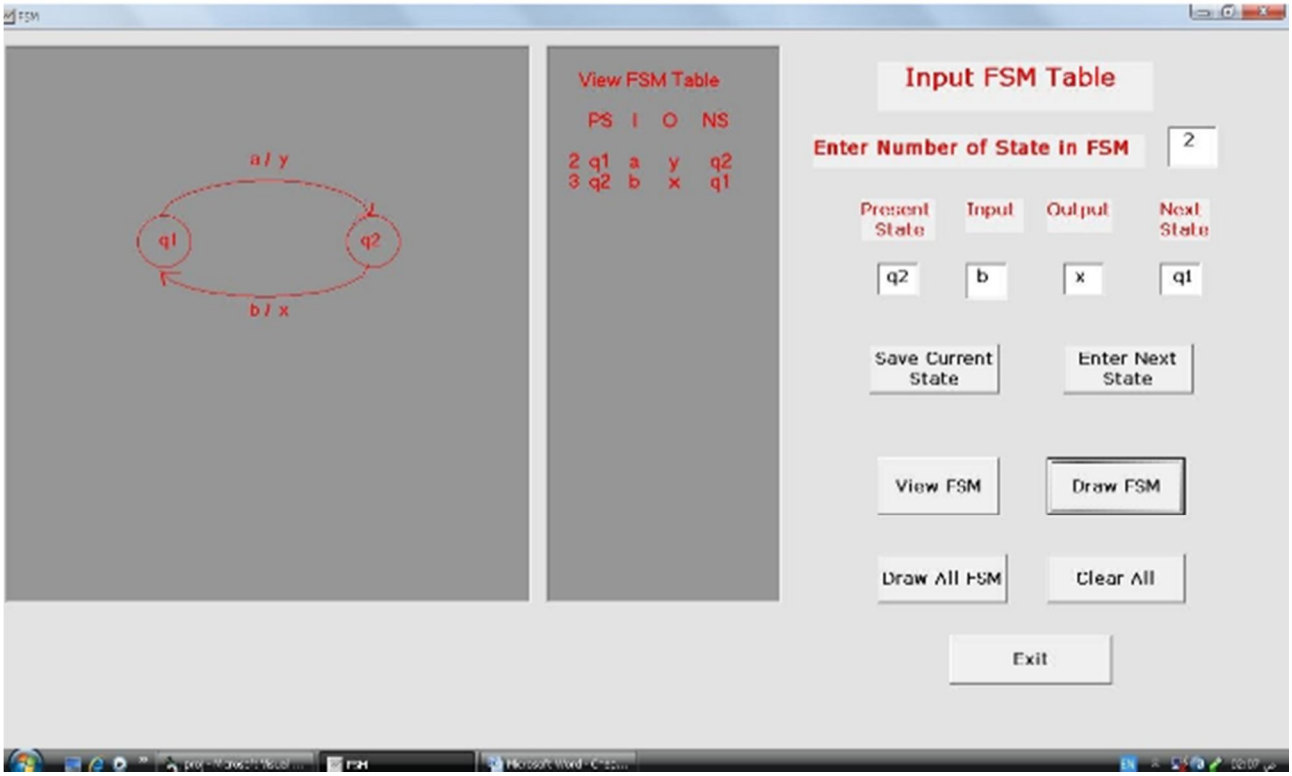


Fig.11: Applying of 2-states.

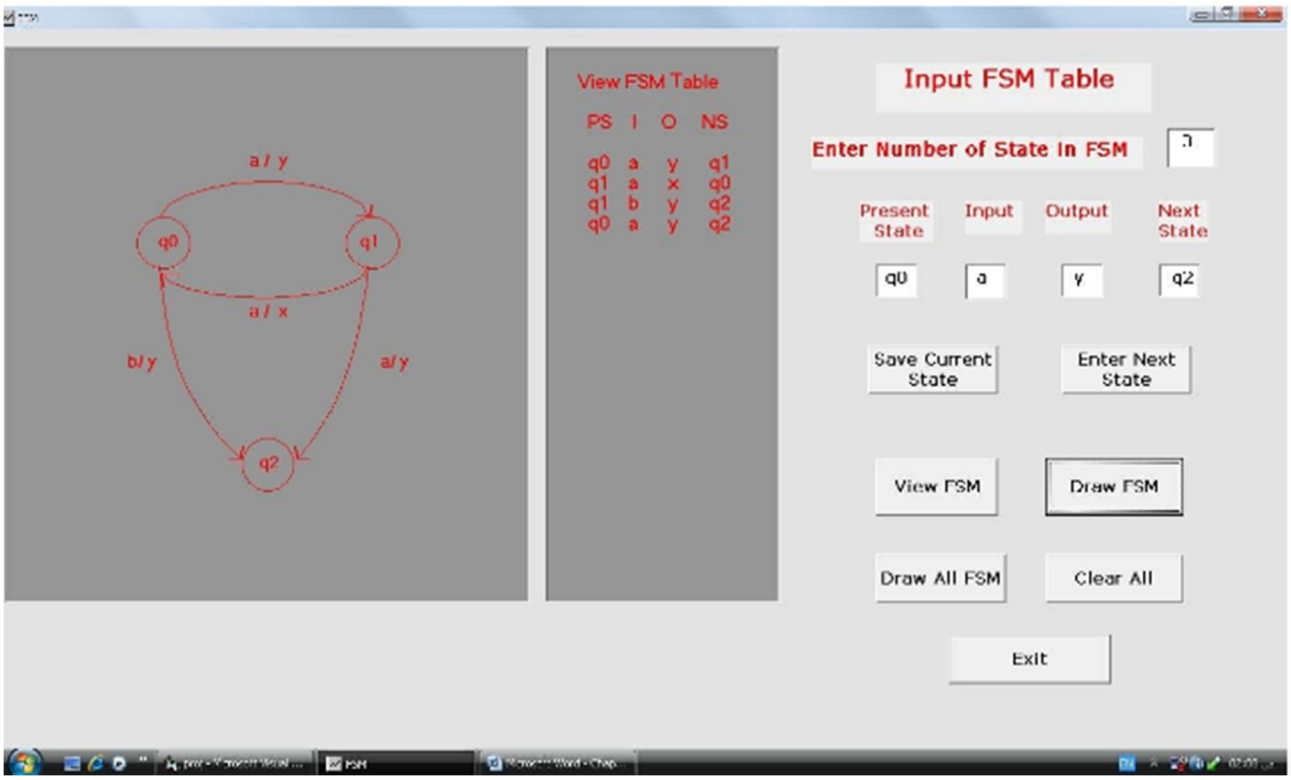


Fig.12: Applying of 3-states.

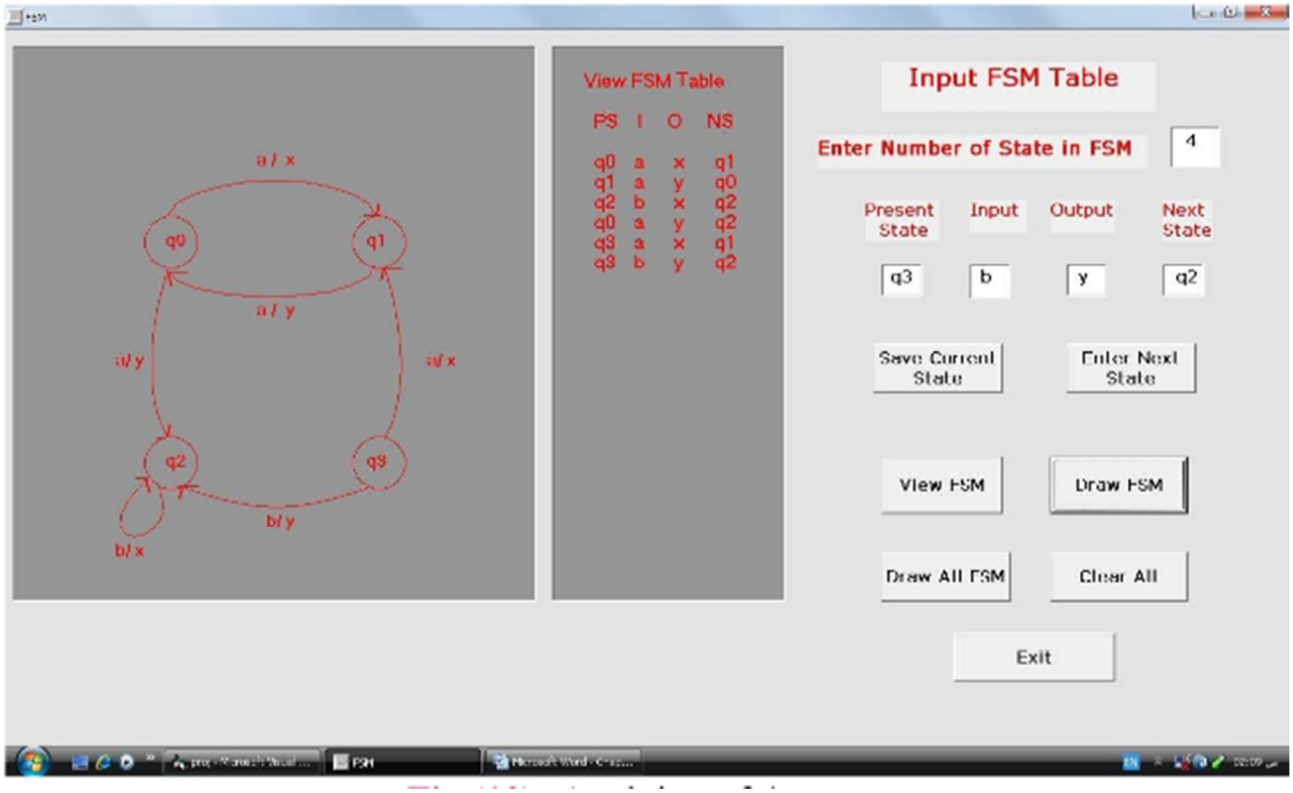
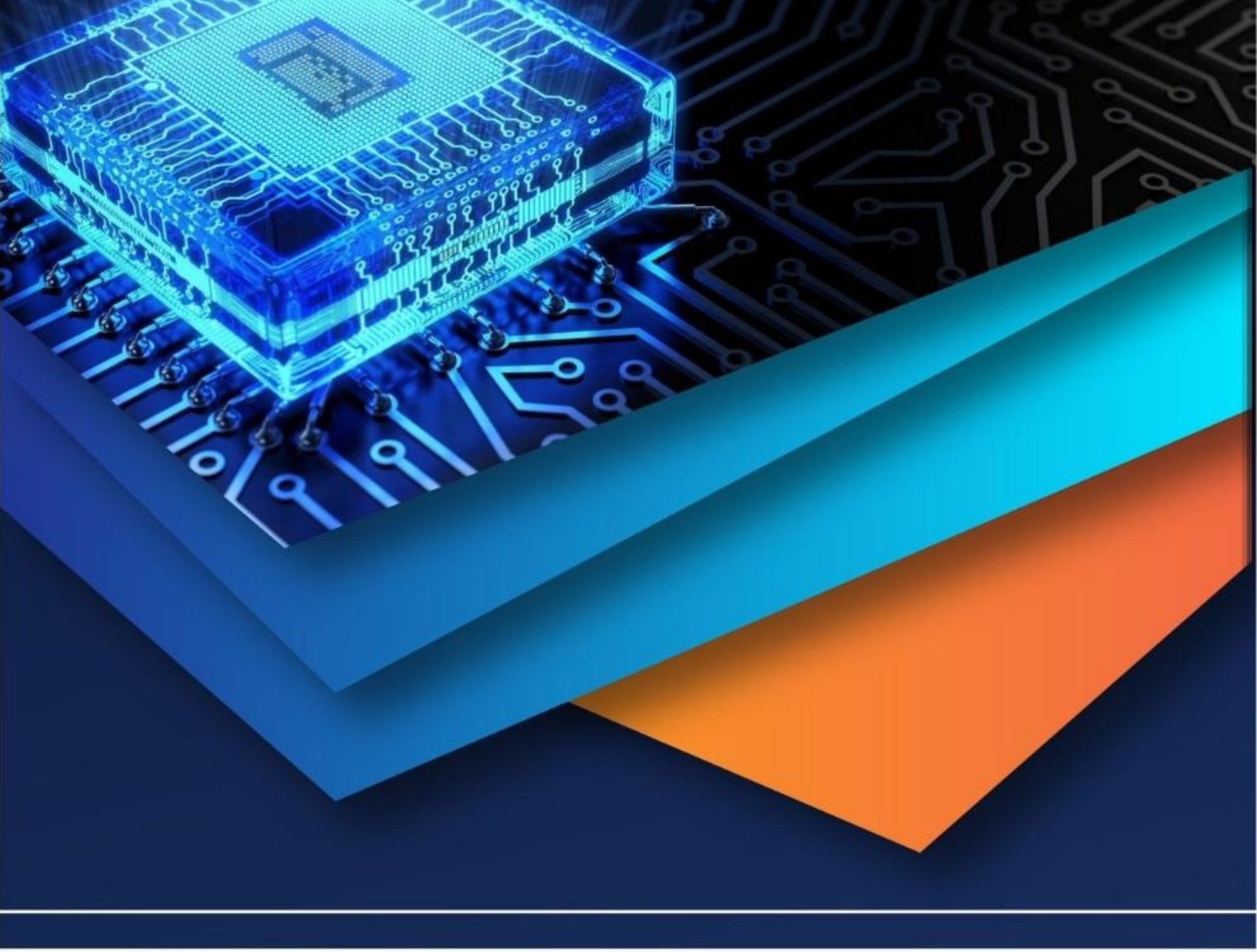


Fig.13: Applying of 4-states.

REFERENCES

[1] Kolman, Bernard, and C, Robert, (1984), "Discrete Mathematic Structure for Computer Science", Busby Prentice Hall Inc..
[2] Deyin, JIN , and Yuji, ITO, (October 2008), "Application of Finite State Machine Theory to the Simulation of Reversed Non-Linear Hysteretic Relationships", TSINGHUA SCIENCE AND TECHNOLOGY ISSN1007-0214 08/67, Volume 13, Number S1, pp46-52,http://qhxb.lib.tsinghua.edu.cn/myweb/english/2008/2008es1/46-52.pdf
[3] F. Wagner, (2006), "Modeling Software with Finite State Machines: APractical Approach", New York,USA:AuerbachPublications,ISBN0-8493-8086-3.http://is.ifmo.ru/.../modelingsoftwarewithfinitestatemachinesapracticalappro ach.pdf
[4] Hendry, Dr DC, (March 9, 2006), " Finite State Machine Design", internet document, http://www.abdn.ac.uk/~eng186/eg3560/presentation12.pdf.
[5] Zolt_an Juh_asz, _Ad_am Sipos, and Zolt_an Porkol_ab, (2005), "Implementation of a Finite State Machine with Active Libraries in C++?", H-1117 Budapest, P_azm_any P_eter s_et_any 1/C. http://aszt.inf.elte.hu/~gsd/s/cikkek/alistair/active.pdf
[6] Mano, M. Morris, (1999), "Digital Design", Second Edition, Prentice, Hall Inc.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY