FINITE STATE MACHINE USING COMPUTER AIDED DESIGN

Noveri Lysbetti Marpaung

Jurusan Teknik Elektro, Fakultas Teknik Universitas Riau
Kampus Bina Widya, Jl Subrantas – Simpang Baru, Pekanbaru, 28293
e-mail: noverim@yahoo.com

Abstract

The aim of this study (Finite State Machine Using Computer Aided Design) is to analyse the framework of a simulation created to inform the State Assignment and State Diagram, whereas the State Diagram depends on the number of State Assignment. A Finite State Machine (FSM) that used in the simulation is a Moore Machine. The programming languages used to implement the FSM are ABEL and Java. The ABEL language is simpler than Programming Logic Devices and Very Large Scale Integration. Compared to C++, The Java Programming Language also offers same advantages.

By using the two programming languages, author plans firstly to display information that contains data about state transition of Moore machine in a test file. Secondly, author wants to display the state assignment of Moore machine and read it line by line, depend on which line that user wants to read. In this case, the user must inform the program like as how many state assignments that user wants to read. Then the program will ask user to input the number of assignments. Thirdly, the author plans to produce the state diagram depend on the number of state assignment that user wants. This research worked consistent with the framework.

Keywords: Finite State Machine, Moore Machine, State Table, State Transition, State Diagram, State Assignment.

1. Introduction

The development in technology is something that interesting to discuss, especially if it involves about computer and software. This phenomenon constitutes a reason from the author to do this research.

Technology in computer science develops rapidly. In short time, many companies have succeeded in improving their technology tools to balance this growth. This also happens in the software environment. In a short period of time, a new version of the software has been developed in response to customer demand. This phenomenon is an interested thing to the author to use software in this research.

The aim of this research is to produce an implementation of a Finite State Machine (FSM).

In this case, this research will pay more attention to Moore Machine using Computer Aided Design (CAD).

The result of this research will :

- Display data of FSM in text file form.
- Show the number of the State Table.
- Show the State table of FSM in Moore Machine form.
- Read the State Table data line by line, depend on the number of line to be read.
- Display State Diagram of Moore Machine.

2. Teoritical References

In general, there are two types of logic circuit according to outputs depend on only its current state inputs or not. The two types of logic circuit are a combinational circuit and a sequential circuit. In a combinational logic circuit, all of the outputs depend only on its current inputs. For instance gate i.e. OR-gate, AND-gate and NOT-gate or inverter. If outputs at any time depend not only its current inputs but also on the past sequence of the inputs, is called a sequential circuit. It can be said that the sequential circuit has memory of the past events. For example is a counter.

The sequential circuits are also known as sequential circuits/machines or Finite State Machines (FSM). A sequential circuit can be modeled as a combinational circuit with the additional properties of storage or memory device (to remember the previous sequence of inputs) with feedback. FSM can be drawn as a block Diagram in Figure 1.

According to Figure 1, memory devices contain information of the current state. Correspond to the current state of the circuit, the secondary inputs (also called state

variables), y, represent the next state, whereas secondary outputs, $Y = f(x, y)$ and the primary outputs (Z), is a function of inputs (x), and internal states, y, so, $Z = f(x, y)$.[1]

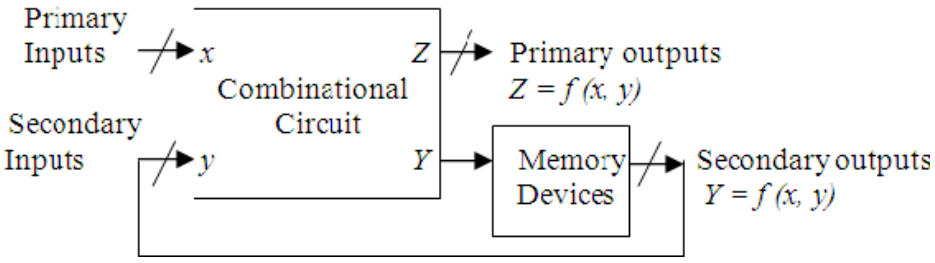

Figure 1. Model of a Finite State Machine

3. Synchronous and Asynchronous Circuit

Referring to the classification depends on the timing of its signal, there are two main classes of sequential FSM: Synchronous sequential circuit (clocked) and Asynchronous sequential circuit (un-clocked).

Generally, all synchronous sequential circuits change their state and output values at fixed points of time, whereas asynchronous sequential circuits change their state and output values when there is a change in input values occurs. The state change of most sequential machine occurs at times which are specified by the rising or falling edge of a free-running clock signal. The timing Diagram and nomenclature for a typical clock signal can be seen in Figure 2.

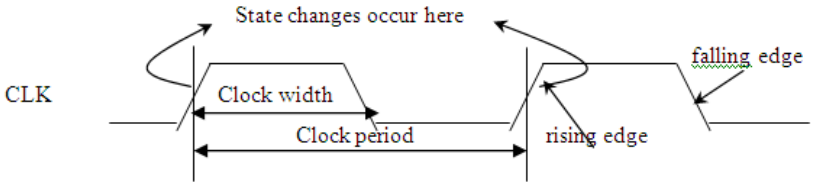

Figure 2.a. Active High

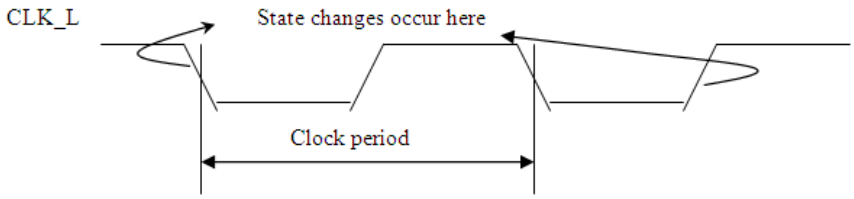

Figure 2.b. Active Low

Figure 2. The timing Diagram For A Typical Clock Signal

4. Finite State Machine

The most general model of a sequential circuit has inputs, outputs and internal states.[4] There are two models of sequential circuits: Mealy Model and Moore Model. Since this research will emphasize Moore Model of FSM, so Mealy Model will not be discussed. The general structure of clocked synchronous state-machine with Moore Model is represented in Figure 3.

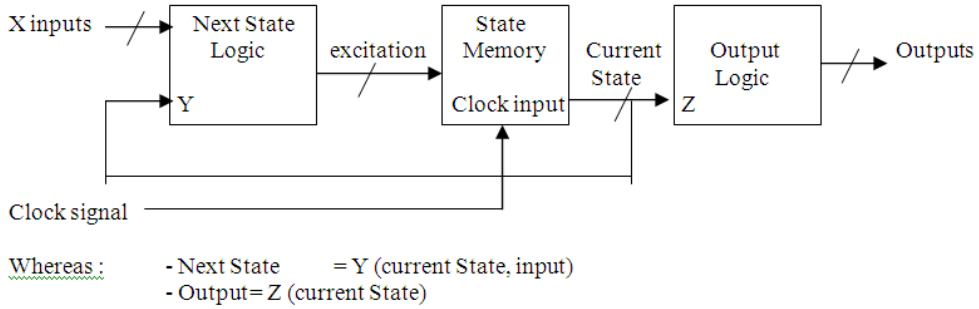

Figure 3. Clocked Synchronous State-Machine Structure With Moore Machine

5. Moore Machine

In the case of a state-based or Moore type sequential circuit, however, the output values depend solely on its current state. The State Table in Moore Model sequential circuit, next state and output be placed in the different column, so the next-State Table just represent the next state. It can be seen from example in Table 1.a. and Table 1.b. (from Table 1. The State Table of Moore Type).

In the State Diagram of Moore Model sequential circuit, the value of the output signal is added to each current state, separated by slash (/). In Figure 4. shows the State Diagram of Moore-type sequential circuit.

Table 1. The State Table of Moore type

Table 1.a. Unassigned Table

Current State	Next State		Output
	x = 0	x = 1	
A	A	B	0
B	B	C	0
C	C	D	0
D	D	A	1

Table 1.b. Assigned Table

Current State	Next State		Output
	x = 0	x = 1	
00	00	01	0
01	01	10	0
10	10	11	0
11	11	00	1

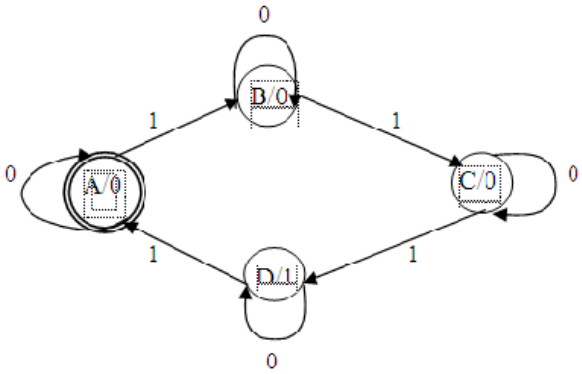

Figure 4. The State Diagram of Moore Type

6. The ABEL Hardware Description Language

ABEL (Advanced Boolean Equation Language) is a hardware description language (HDL) that is used to allow designers to specify logic function for realisation in PLDs (Programmable Logic Devices).[3]

An ABEL program is a text file which containing several elements i.e.:

- Documentation contains program name and comments.
- Declaration is used to identify the inputs and outputs of logic functions to be performed.
- Statements to specify the logic functions to be performed.
- A declaration of the PLD type or other devices in which the specified the logic functions, usually, are to be performed.
- Usually, test vectors are used to specify the logic functions expected outputs for certain inputs.

ABEL is supported by an ABEL language processor or called ABEL compiler. The job of the compiler is to translate the ABEL text file into a fuse pattern that can be download into a physical PLD. ABEL language allows PLD functions to be expressed with truth tables or nested if statements as well as by any algebraic expression format even though, physically, patterns corresponding to sum-of-products expressions can be used to program most PLDs. If possible the ABEL compiler manipulates text formats and minimizes the result of equations to fit into the available structure of PLD.

The advantage of ABEL rather than VHD :

- ABEL is common used as a specification language for programmable logic components. ABEL is a simple language, suitable for digital systems of moderate

complexity. VHDL is more complete and it is capable of describing systems of considerable complexity.

An example of an ABEL program for module half-adder. .[6]

```
MODULE half_adder;
A, B, Sum, Carry PIN 1, 2, 3, 4;
TRUTH_TABLE
([A, B]) -> ([Sum, Carry])
    [0, 0] -> [0, 0]
    [0, 1] -> [1, 0]
    [1, 0] -> [1, 0]
    [1, 1] -> [0, 1]
END half_adder;
```

- ABEL can specify the relationship between circuit inputs and outputs in several alternative ways, including equations and truth tables. Part of the ABEL specification joints input and output variables with specific pins on the programmable logic component chosen to implement function.

- VHDL is a widely used language for hardware description based on the programming ADA. VHDL is used as the input description for a number of commercial available Computer Aided Design (CAD) systems.

7. The Java Programming Language

Java is both a programming language and an environment for executing programs that are written in the Java language. Java is unlike any other programming language. Java is designed to work easily within the World Wide Web (WWW) of computers through commonly available. User-friendly software is called browsers. All browsers are now Java-enabled. Actually, Java programs that run on the web are called applets (short for ’little applications’). A scheme of Java applet can be seen in Figure 5.

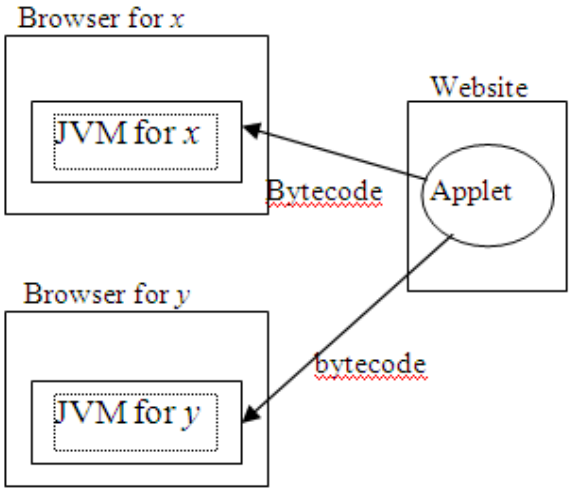

Figure 5. Java Applet With Two Different Computer (X And Y).[8]

Java-enabled means that there is a program inside the browser, known as *Java Virtual Machine (JVM)*, which can run Java for the particular computer. The advantage of being Java-enabled is that rather than of just passive text and images appearing on the screen, interaction and calculation take place as well. Information can be sent back to the *host site* to get more Java programs and more documents and generally, it also performs in a very effective way.

Java Virtual Machine is a Machine, which is a system program that has been built to run on many different computers. The JVM is an interpreter that interprets each *byte code* instruction into an appropriate Machine language instruction and also executes it. Java compiler creates an object code called *byte code*. Java compiler translates Java source code into instructions that are interpreted by the runtime Java Virtual Machine. So, can be said that Java is an *interpreted language*.

The programming process involves creating a sequence of instructions to the computer that is written in the particular programming language. The process is expressed through an *editor*. Once ready, the program is submitted to a compiler. If there are any errors (called *compilation* or *syntax errors*), then these must be corrected and the program resubmitted for compilation.

8. Research Procedure

This research will demonstrate the simulation of Finite State Machine Using Computer Aided Design. The procedure of this research is :

Firstly, understand the condition of control system that is wanted. It can be got from the theoretical references. Secondly, design the State Assignment, State Table

and State Diagram of Moore Machine. Thirdly, construct the State Assignment, State Table and State Diagram of Moore Machine. Fourth, construct the program of the State Assignment, State Table and State Diagram.

After the planning process done, the next step is check the program. Is the program right or not? If the program is right, so test the program. If the program is still not right, check the program again and improve it until obtain the right program. Finally, make analyses of the result until get the conclusion.

9. Results Of Simulation

The results of the research are:

1. Display of the data from text file which is an input to the Java program. Data that is read can be used as input for the other programs. It can be seen in Figure 6.

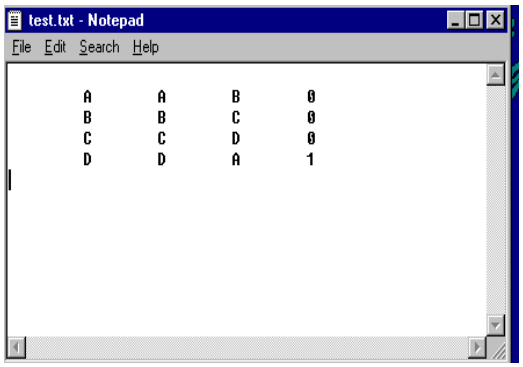

Figure 6. The Data That Is Read From The Text File

2. Display of the whole information of Moore Machine information as a State Table. It can be seen in Figure 7.

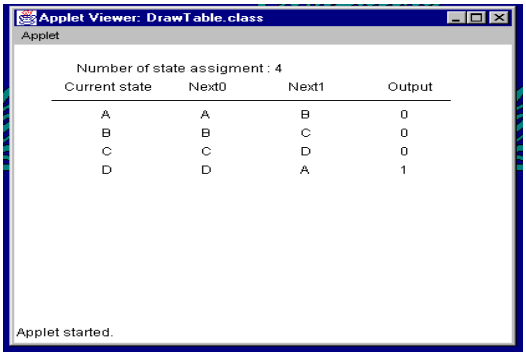

Figure 7. The Data Information Of State Table

3. Displays of the number of State Assignments for the Moore Machine. The program will also display data, line by line from the information that is read from the text file. The user can decide which line from the text file to display. It can be seen in Figure 8.

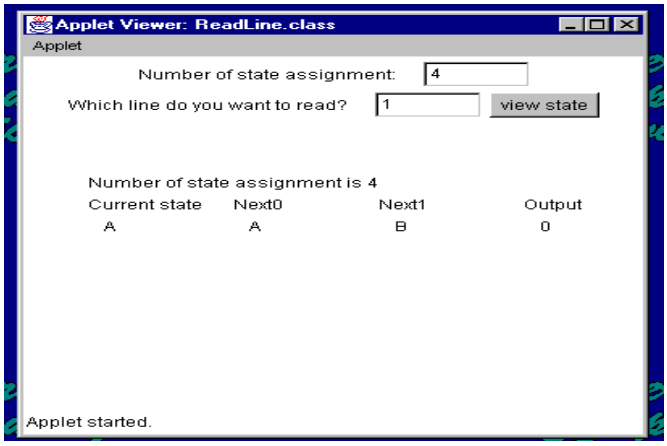

Figure 8. Data Per Line That Is Read By A User From State Table

4. Display of the certain line but it is not a part of State Assignment. It can be seen in Figure 9.

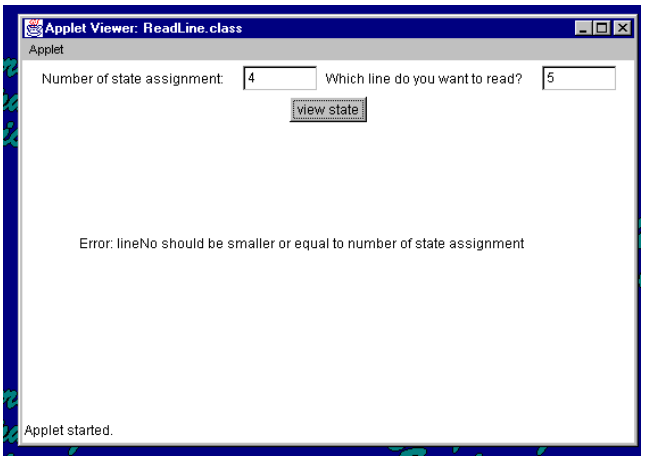

Figure 9. The Certain Line Which Is Not A Part of State Assignment

5. Display of the State Diagram of Moore Machine. It can be seen in Figure 10.

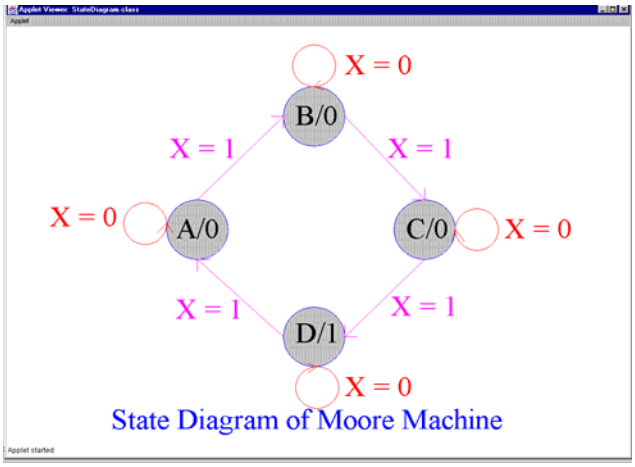

Figure 10. The State Diagram of Moore Machine

10. Analysis of The Result

1. The user must write down the data from text file. This data will be read as an input to the program. This data can be seen in Figure 6.
2. When the program running, the computer will show the '*Number of State Assignment*' (depend on the input that was entered) and a State Table of the data that had been entered. It can be seen in Figure 7.
3. If the user wants to see a certain line of State Assignment, so the program will show the '*Number of State Assignment*'. Then it asks user to enter which line will be read by user. So, the program will show the certain line that is asked. the It can be seen in Figure 8.
4. If the user wants to see the certain line but not a part State Diagrams so the display will show like as Figure 9.
5. After shows the State Assignment, the computer will show the State Diagram of Moore Machine depends on the number of State Assignment such as Figure 10.

11. Conclusion

1. This research worked consistent with the framework. The data from text file be an input to the Java program.
2. The input data produces the State Assignment, State Table and State Diagram.
3. This research shows that the State Diagram depends on the number of State Assignments.
4. The State Diagram is built up is merely drawn by the graphical user interfaces.

5. In this implementation, the result produced performs the State Assignment and State Diagram in different space.
6. If user wants to see the certain line of State Assignment, so in the display will appear the certain line that is needed.

References

[1] Lewin, D. and Protheroe, D., *Design of Logic Systems, Second Edition*, Chapman and Hall, 1992.

[2] Gajski, D., *Principles of Digital Design*, International Edition, Prentice-Hall International, Inc., 1997.

[3] Wakerly, J., *Digital Design*, Third Edition, Prentice-Hall International, Inc., 2000.

[4] Manno, M. M., *Digital Design*, Second Edition, Prentice-Hall, Upper Saddle River, NJ07458, 1991.

[5] Wakerly, J., *Digital Design*, Second Edition, Prentice-Hall International, Inc., 1994.

[6] Katz, R. H., *Contemporary Logic Design*, The Benjamin/Cummings Publishing Company, Inc., 1994.

[7] Koffman, E. and Wolz, U., *Problem Solving with Java*, Addison-Wesley, 1999.

[8] Bishop, J., *Java Gently*, Third Edition, Addison-Wesley, An Imprint of Pearson Education, 2001.

[9] Flanagan, D., *Java in a Nutshell*, O'Reilly and Associates Inc., 1996 - 1997.

[10] Cohn, M., Morgan, B., Morrison, M., Nygard, M. T., Joshi, D., and Trinko, T., *Java Developer's Reference*, Sams Net Publishing, 1996.

[11] Lewin, D., *Design of Logic Systems*, Van Nostrand Reinhold (United Kingdom) Co. Ltd., 1985.

[12] Arnold, K., Gosling, J., and Holmes, D., *The Java™ Programming Language*, Third Edition, Addison-Wesley, November 2000.

[13] Clare, C. R., *Designing Logic Systems Using State Machines*, McGraw Hill, New York, 1973.

[14] Lala, P. K., *Practical Digital Logic Design and Testing*, Prentice-Hall, 1996.

[15] Stonham, T. J., *Digital Logic Techniques Principles and Practice*, Chapman and Hall, 1991.

[16] Chan, P. and Lee, R., *The Java ™ Class Libraries : An Annotated Reference*, Addison-Wesley, Longman, Inc., 1997.

[17] Van der Linen, P. *Just Java*, SunSoft Press – A Prentice Hall Title, 1996.