International Journal of Innovative Research and Advanced Studies (IJIRAS)
Volume 4 Issue 7, July 2017

ISSN: 2394-4404

Finite State Machine Approach To Electronic Equipment Design

Kamalu, U.A.

Felix Ogbor.

University of Port Harcourt, Rivers State, Nigeria

Abstract: ***In this report, a finite state machine approach to electronic design presented. A finite state machine approach to electronics design proposed in this work has methods; the sequential logic and embedded system approaches. The three states represent card types of the proposed machines. At a time, only one of the states will be active which, Finite State Machine (FSM) modelling is the most crucial part in developing proposed model as this reduces the hardware. In this paper the process of four state (card insertion, card type (state) Selection, waiting for money dispensing) has been modelled using MEALY Machine Model. Also, a milk vending machine was used to explore the embedded system approach which was achieved.***

Keywords: MEALY, FSM, sequential, logic, circuits, deterministic, machines, non-deterministic

I. INTRODUCTION

A finite-state machine (FSM) approach to electronic design is a mathematical model approach to electronic design. It is an approach that gives an electronic machine the intelligence of being in that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some requested external inputs; the change from one state to another is called a transition. A FSM is defined by a list of its states, its initial state, and the conditions for each transition. The Finite State Machine is a theoretical mathematical model of a sequential logic function. It has limited inputs, outputs and number of states.

A finite state machine is one of the most popular design patterns in embedded systems (electronic). Many applications from simple home appliances to complex communication systems implement event based state machines. The finite state machine is made up of multiple states. At any point of time, the system is in one state and an event triggers certain actions in that state along with a possible change in state. The event could be due to an interrupt in the system, an RTOS signal, a timer expiry indication or an input or indication from another module in the system.

Finite-State Machines, FSM, is a primitive, but useful computational model approach for both hardware electronic equipment design and certain types of software. It also involves regular expressions, the correspondence between non-deterministic and deterministic machines, and more on grammars. To describe typical hardware components that are essentially physical realizations of finite-state machines. Finite-state machines provide a simple computational model with many applications. Recall the definition of a Turing machine: a finite-state controller with a movable read/write head on an unbounded storage tape (). If we restrict the head to move in only one direction, we have the general case of a finite-state machine. The sequence of symbols being read can be thought to constitute the input, while the sequence of symbols being written could be thought to constitute the output. We can also derive output by looking at the internal state of the controller after the input has been read. Finite-state machines, also called finite-state automata (singular: automaton) or just finite automata are much more restrictive in their capabilities than Turing machines. For example, we can show that it is not possible for a finite-state machine to determine whether the input consists of a prime number of symbols. Much simpler languages, such as the sequences of well-balanced parenthesis strings, also cannot be recognized

ISSN: 2394-4404

by finite-state machines. Still there are the following applications:

- ✓ Simple forms of pattern matching (precisely the patterns definable by "regular expressions", as we shall see).
- ✓ Models for sequential logic circuits, of the kind on which every present-day computer and many device controllers is based.
- ✓ An intimate relationship with directed graphs having arcs labeled with symbols from the input alphabet. Even though each of these models can be depicted in a different setting, they have a common mathematical basis. The following diagram shows the context of finite-state machines among other models we have studied or will study.

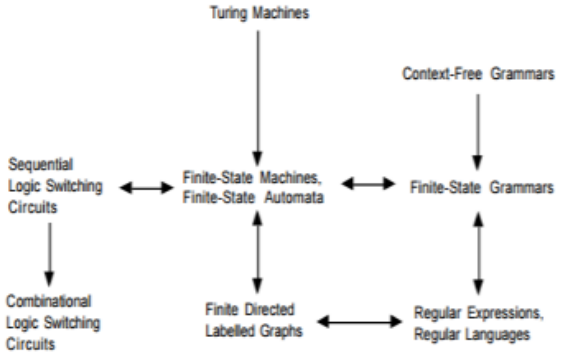

Figure 1: The interrelationship of various models with respect to computational or representational power

II. RELATED WORKS

In sequential circuit, electronic equipment design, an effective approach to reduce power dissipation is to "turn off" portions of the circuit, and hence reduce the switching activities in the circuit. Two attempts have been made to exploit such an approach. Alidina et al. (1994) proposes a precomputation-based approach in which the output values of a sequential circuit are precomputed so that the original circuit can be turned off in the next clock cycle. Benini and De Micheli (1995b) described a scheme to stop the clocking of a finite state machine (FSM) when the machine is in a self-loop and the outputs do not change.

(SUE-HONG et al, 2005) In his article proposed a technique that is also based on selectively turning off portions of a circuit. Their approach was motivated by the observation that, for an FSM, active transitions occur only within a subset of states in a period of time. Therefore, synthesizing an FSM in such a way that only the part of the circuit which computes the state transitions and outputs will be turned on while all other parts will be turned off, power consumption will be reduced.

Fred, (2015) presented a finite state machine approach for fault tolerance and implement decentralized control in distributed systems. This paper reviews the approach and identifies abstraction for two different failure models – Byzantine and fail- stop are discussed.

Ana et al (2012) Proposed machines in these countries is on the top worldwide. This is due to the modern lifestyles which require fast food processing with high quality. They described the designing of multi select machine using Finite State Machine Model with Auto-Billing Features. Finite State Machine (FSM) modelling is the most crucial part in developing proposed model as this reduces the hardware

FSM (Finite State Machine) In a Finite State Machine the circuit's output is defined in a different set of states i.e. each output is a state. A State Register to hold the state of the machine and a next state logic to decode the next state. An output register defines the output of the machine. In FSM based machines the hardware gets reduced as in this the whole algorithm can be explained in one process. Two types of State machines are: MEALY Machine: In this machine model, the output depends on the present state as well as on the input. The state diagram for a Mealy machine associates an output value with each transition edge (in contrast to the state diagram for a Moore machine, which associates an output value with each state).

When the input and output alphabet are both Σ, one can also associate to a Mealy Automata an Helix $(S \times \Sigma, (x, i) \rightarrow (T(x, i), G(x, i)))$ .This graph has as vertices the couples of state and letters, every nodes are of out-degree one, and the successor of (x, i) is the next state of the automata and the letter that the automata output when it is instate x and it reads letter i. This graph is a union of disjoint cycles if the automaton is bi-reversible.

The MEALY machine model is shown in figure 2.

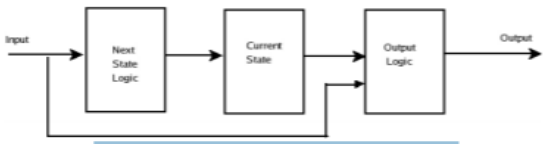

Figure 2: MEALY Machine Model MOORE Machine

In Moore machine model the output only depends on the present state. A Moore machine can be defined as a 6-tuple consisting of the following:

- ✓ a finite set of states
- ✓ a start state (also called initial state) which is an element of
- ✓ a finite set called the input alphabet
- ✓ a finite set called the output alphabet
- ✓ atransition function mapping a state and the input alphabet to the next state
- ✓ an output function mapping each state to the output alphabet

A Moore machine can be regarded as a restricted type of finite-state transducer.

The MOORE machine model is shown in figure 3.

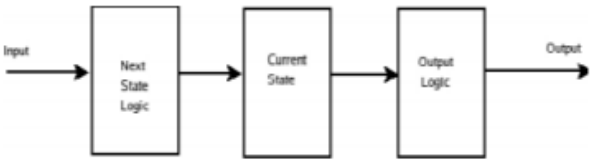

Figure 3: MOORE Machine Model

III. METHODOLOGY

In this paper a state diagram is constructed for the proposed Automated Teller machine which can accept three cards that is Master, Visa and Verve. Three select (select1, select2, select3) inputs are taken for selection of cards. Select1 is used for the selection of MasterCard.

Similarly, select2, select3 are used for Visa and Verve respectively. A cancel input is also used when the user wants to withdraw his request and also the money will be returned through the return output. Return, product and change are the outputs. Return and change vectors are seven bits wide. Money is an in/out signal which can be updated with the total money of your account delivered at a time. Money signal is seven bits wide. Money count is an internal signal which can be updated at every transition. This signal is also seven bits wide. If the inserted money is more than the total money of allowed daily withdrawal limit, then the request will not be successful returned through the change output signal.

The state diagram mainly consists of three states (card insertion, card type Selections, and waiting for the money to come out. Initially when the reset button is pressed, the machine will be ready for the users to insert card and select card type, select the amount to be withdrawn. This state is the initial state of the design. This methodology is explained using a flow diagram shown in figure 3.

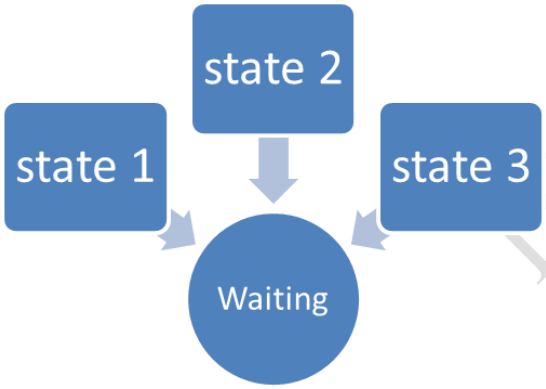

Figure 3: State diagram for the three states

The fundamental idea behind FSM design is that a large class of electronic problems can be viewed from the perspective of finite state machines and can therefore be solved using methods borrowed from other disciplines, particularly digital electronics. The type of software products that lend themselves best to the FSM model can be categorized as having distinct "modes" or being "control intensive"; that is, they contain a fairly complex logic structure that results in a relatively large number of control blocks, such as case structures and conditional statements.

Embedded and real-time systems are good candidates, as are multitasking executives, command interpreters, language processors, communication drivers, device handlers--the list goes on. And while the FSM pattern could also be appropriate for implementing some user-oriented applications such as accounting software or other business packages, it may not be ideal for designing algorithmic or computational routines.

Two methods of the finite state machine approach to electronics design are discussed here:

The finite state approach in design of electronic embedded system: A finite state machine is one of the most popular design patterns in embedded systems. Many applications from simple home appliances to complex communication systems implement event based state machines.

The finite state machine is made up of multiple states. At any point of time, the system is in one state and an event triggers certain actions in that state along with a possible change in state. The event could be due to an interrupt in the system, an RTOS signal, a timer expiry indication or an input or indication from another module in the system. The embedded system method has two different approaches for implementing state machines using C.

Figure 1 shows a sample state machine for a milk vending machine. It has three different states:

- ✓ Idle,
- ✓ Money Inserted and
- ✓ Option Selected.

The system relies on user inputs and signals from the milk dispensing unit. Additional events like debug timer expiry signal could be added.

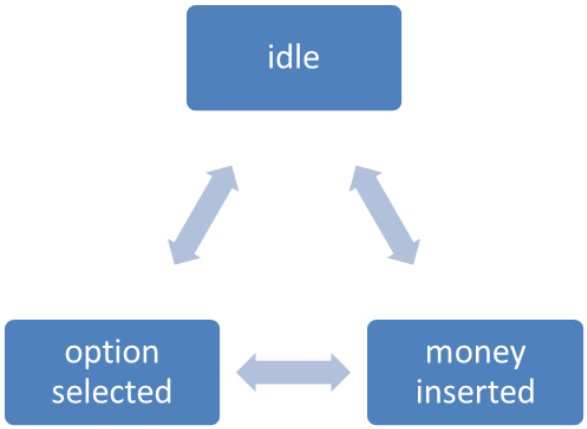

Figure 4: finite state machine approach for a milk vending machine design

There are two common approaches for implementing an event based finite state machine approach to electronic design like the one above:

- ✓ Using conditional statements
- ✓ Using lookup table

Using conditional statements is an extremely simple approach to get the implementation started. When the number of states and events is few, this method is intuitive and developers get a quick picture of what the state machine is doing. However, as the number of states or events grow, the code can easily go unwieldly. Debugging and maintenance get difficult as the state machine runs into multiple screen pages. It gets even more unmanageable when the handlers span multiple files.

Apart from readability and maintenance issues, designers also need to be aware of the overhead introduced with the conditional statements and account for this during design phase, especially for time critical systems.

TABLE BASED APPROACH

In this approach, the finite state machine is coded in a table with one dimension as states and the other as events. Each element in the table has the handler for the state/event combination. The table could be implemented in C using a two-dimensional array of function pointers

This is an elegant method to translate the state diagram to actual implementation as the handling for every state and event combination is encapsulated in the table. Developers get a quick picture of the state machine and software maintenance is also much more under control.

However, when the table is sparse, that is, there are many invalid state/event combinations, this approach leads to a wastage of memory. There is also a memory penalty as the number of states and events grow. Software designers need to accurately account for this during initial design.

OTHER CONSIDERATIONS

In addition to having handlers for every state and updating the next state, many implementations also have logic to clean up the state machine for the current state and initialize for the next state during a state change. This could be achieved by defining entry and exit functions for every state and invoking them during a state change.

Transition tables could also be defined capturing the special handling needed for transition from one state to another.

Overall, there is no default implementation choice for developing a finite state machine like the one described above. One needs to accurately budget for the overheads introduced and also bear in mind factors such as scalability and readability when choosing an implementation approach.

The discrete digital system methods of finite state machine approach to electronic design are implemented in real-life circuits through the use of Flip Flops

In a digital circuit, an FSM may be built using a programmable logic device, a programmable logic controller, logic gates and flip flops or relays. More specifically, a hardware implementation requires a register to store state variables, a block of combinational logic that determines the state transition, and a second block of combinational logic that determines the output of an FSM. One of the classic hardware implementations is the Richards controller.

In a Moore circuit, the output is directly connected to the state flip-flops minimizing the time delay flip-flops and output.

Through state encoding for low power state machines may be optimized to minimize power consumption.

IV. RESULTS AND DISCUSSION

The state diagram shown in figure 3 is a finite machine approach to the design of automated teller machine, ATM. The concept is to select three card types represented by 3-states. Whenever the card is accepted, a waiting process begins which includes counting and dispensing of money. In another example, a milk vending machine was designed using the finite state machine approach. The milk vending machine has states such as idle, insert money and select option. Once money is inserted, the vending machine releases the quantity of milk equivalent to the money.

V. CONCLUSION

In this report, a finite state machine approach to electronic design was studied, the FSM algorithm is such that most of the time, only one of the states will be active which, consequently, lead to polling system. The designed approach which can be used for many applications and can easily enhance the number of selections. The finite state machine was applied to design an ATM and a milk vending machine. And the objectives of the study were achieved.

REFERENCES

[1] Ana Monga1 , Balwinder Singh (2012) Finite State Machine based Vending Machine Controller with Auto-Billing Features Centre for Development of Advanced Computing(C-DAC), Mohali, India

[2] B. Caulfield & M.O Mahony (2005) "Passenger Requirements of a Public Transport Ticketing System" Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems Vienna, Austria, pp-32-37.

[3] Bhaskar "VHDL primer" Second Edition,

[4] Biplab Roy & Biswarup Mukherjee (2010) "Design of Coffee Vending Machine using Single Electron Devices" Proceedings of 2010 International Symposium on Electronic System Design. Pp 38-43.

[5] C. J Clement Singh, K Senthil Kumar, Jayanto Gope, Suman Basu & Subir Kumar Sarkar (2007) "Single Electron Device based Automatic Tea Vending Machine" proceedings of International Conference on Information and Communication Technology in Electrical Sciences (ICTES 2007, pp 891-896.

[6] Fauziah Zainuddin, Norlin Mohd Ali, Roslina Mohd Sidek, Awanis Romli, Nooryati Talib & Mohd. Izham Ibrahim (2009) "Conceptual Modeling for Simulation: Steaming frozen Food Processing in Vending Machine" International Conference on Computer Science and Information Technology, University Malaysia Pahang, pp.145-149.

[7] J.Komer (2004) "Digital logic and state machine design", 2nd ed., Oxford.

[8] M. Zhou, Q. Zhang & Z. Chen (2006), "What Can Be Done to Automate Conceptual Simulation Modelling?" Proceedings of the 2006 Winter Simulation Conference, pp. 809 – 814. International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.2, April 2012 27

[9] M. Zhou, Y. J. Son, & Z. Chen, (2004), "Knowledge Representation for Conceptual Simulation Modeling" Proceedings of the 2004 Winter Simulation Conference, pp. 450 – 458.

[10] Muhammad Ali Qureshi, Abdul Aziz & Hafiz Faiz Rasool "Design and Implementation of Automatic Ticket System using Verilog HDL" proceedings of tnternational conference on Information Technology, pp- 707-712.
[11] P. Smith (1997) "Automatic Hot-food Vending Machine," Trends in Food Science & Technology October 1997, Vol. 81, and pp. 349.
[12] Peter Minns & Ian Elliott, "FSM-based Digital Design using Verilog HDL", John Wiley & Sons Ltd 2008.
[13] Seyed Bahram Zahir Azami & Mohammad Tanabian "Automatic Mobile Payment on a nonConnected Vending Machine" proceedings of Canadian Conference on Electrical and Computer Engineering Steve Kilts," Advanced FPGA Design: Architecture, Implementation, and optimization", Wiley-IEEE press, 2007.
[14] Xilinx Inc., Spartan 3 Data sheet: http://ww w.xilinx.com.
[15] Zhang Wen & Zhang Xin Long (2010) "Design and Implementation of automatic vending machine Based on the short massage payment" International Conference on Information and Communication technology in Electrical Sciences, Neijiang, Sichuan, China.pp.978-981. 2004, pp-731-734.
[16] Ted Carmely (1992) Fundamentals of Finite State Machines/Better Software Through Finite State Design TC Systems.