Design, Modeling and Implementation of 8-bit Processor for Intelligent Automatic Chocolate Vending Machine (AVM)

Ankit Chadha, Shreyas Gaonkar, Aditi Desai
Department of Electronics and Telecommunication
Vidyalankar Institute of Technology
Mumbai, India

ABSTRACT
This paper describes the design, modeling and simulation of an 8-bit dedicated processor for a Chocolate Automatic Vending Machine (AVM). The proposed dedicated processor is modelled by writing appropriate programs in. The system supports 8 different 8-bit instructions.

System simulation was carried out using Xilinx ISE Design Suite 14.6. After accepting the user input which is in the form of desired amount (Re.1, 5, 10 or 20) and second input i.e. choice of the chocolate, the system formulates the chocolate using pre-programmed appropriate proportions and vends it automatically to the user. The system would formulate the recipes with embedded logic. Each recipes involves mixture of various other ingredients which finally results in the desired type of chocolate formation. The Chocolate AVM unlike most other vending machines, prepares chocolate instantly. When the entire process is complete, the vending machine should produce the chocolate and return if any change left.

Keywords
AVM, Chocolate Vending, Xilinx, Processor.

1. INTRODUCTION
In 1883, Percival Everitt invented the first vending machine. It soon became quite popular at railway stations and post offices, dispensing envelopes, postcards and note paper.

Vending machine marks its first appearance in 1888. It was built by Thomas Adams Gum Company in New York selling gums on the train platforms. [1] Today, these machines have more sophisticated than ever and have also become more complex, dealing with multiple products. A modern day vending machine is capable of vending out 20-30 different products from a single machine. "Cioccolato Machine" or the chocolate vending machine works the same as a traditional vending machine, except the fact that it produces instant chocolate then and there and then vends it out.

2. FINITE STATE MACHINE (FSM)
A finite state machine (FSM) [2] is a digital sequential circuit. It consists of many pre-defined states that can be controlled by various inputs [3]. The output of finite state machine remains unchanged until the inputs changes. There are two types of finite state machines: 1- Synchronous FSMs (also known as Mealy machine) 2-Asynchronous FSMs (also known as Moore machine). Synchronous FSMs have a clock input whereas asynchronous FSMs do not have a clock input in addition to the data input [4], [5]. The proposed algorithm vending machine is a sequential circuit which is based on Mealy Model. In a Moore machine Model the output of the machine totally depends on the present state, while in a Mealy machine Model the output is depend on the present state as well as the previous input.

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 as shown in the figure.

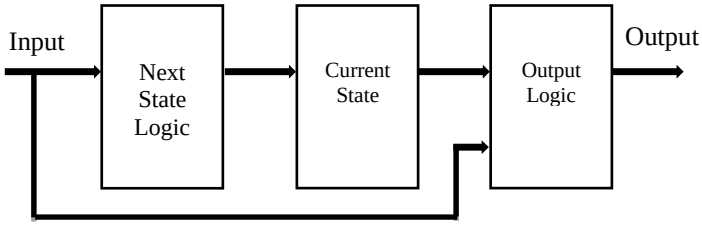

Figure 1: Mealy Machine Model

MOORE Machine: In Moore machine model the output only depends on the present state as shown in the figure.

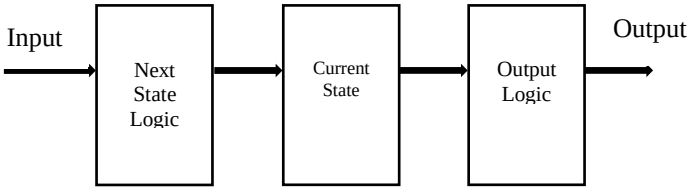

Figure 2: Moore Machine Model

3. OPERATION OF VENDING MACHINE
1. When the user puts the money, Money counter tells the control unit, the amount of money inserted in

Vending Machine.

2. When the user presses the button to purchase the products that he wants, the control unit turns on the motor and dispenses the product if correct amount is inserted.

3. If there is any change, machine will return it to the user.
4. The machine will demand for servicing when the products are not available inside the machine.

4. RELATED WORK
Various researches have been carried out in order to design the Vending Machines. A few of them are discussed here as: Fauziah Zainuddin [6] has proposed vending machine for steaming frozen food by means of conceptual modelling. This has three main states (user selection state, freezer state and steaming state) and has used process approach, which

emphasizes on the process flow or control. Conceptual modelling is described in "Passenger Requirements of a Public Transport Ticketing System" [7]. In "FSM-based Digital design using Verilog HDL" [8] the concept of automatic mobile payment is discussed. In "Passenger Requirements of a Public Transport Ticketing System" [7] the passenger's necessities for ticketing system are given. In "Knowledge Representation for Conceptual Simulation Modeling" [9] a coffee vending machine is designed using single electron encoded logic. The designed circuit is tested and its power and switching time is compared with the CMOS technology.

5. CHOCOLATE VENDING MACHINE

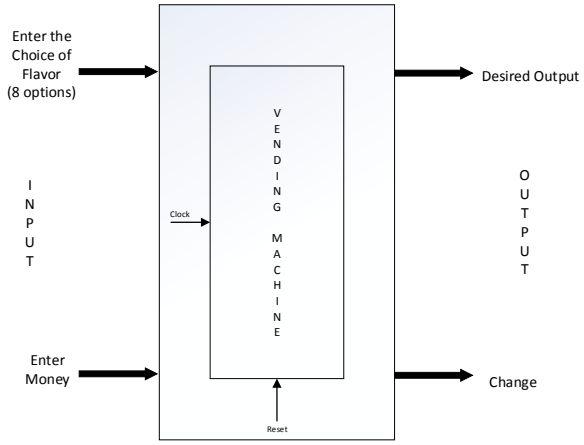

Figure 3: Block diagram of Vending Machine

Inputs to the vending machine are the different variety of chocolates to choose from and entering the money itself. Depending upon the type of chocolate, the selection and the cost will vary. Once correct amount is entered, the machine will produce the chocolate, vend it out and return change, if any.

5.1 Inputs

The amount for each chocolate varies with the type of chocolate to vend. At the input of the vending machine a user has to select a button which gives him a choice of 8 different chocolate flavors. And, the money denominations are accepted are Rs1, 5, 10, and 20. Change (if any) would be given back to the user after vending the chocolate.

Table 1: List of Products with prices

Chocolate flavors	Price
Gianduja	Rs.1
Cou-ver-ture	Rs.3
Dark Chocolate	Rs.4
Sweet Chocolate	Rs.5
Chocolate Caramel	Rs.6
Milk Chocolate	Rs.8
White Chocolate	Rs.10
Chocolate Praline	Rs.20

The following table shows the product details along with cost of each chocolate.

5.2 Outputs

5.2.1 Desired Product

Vendee gets the desired product when
(Money inserted) > Or = (Price of the product)

5.2.2 Change

(Change) = (Money inserted) – (Price of the product)
(Money inserted) > (Price of the product)

6. IDEA OF THE PROPOSED SYSTEM

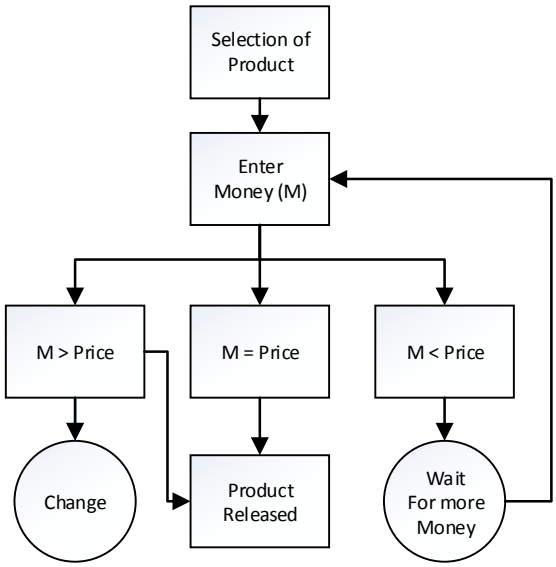

Figure 4: Flowchart of AVM

The recipe for each type of chocolate is given and using this set of instruction sequences for each recipe can be obtained. This gives us the list of opcodes that will be fed into the instruction memory (IM).

6.1 Features

1) In all, 8 instructions will be used in order to realize the entire functioning of the machine. A detailed data path for each instruction is first prepared.

2) A combined data path is then prepared. A combined data path is the one which is capable of executing all the instructions, given one at a time.

3) In order to increase the number of outputs in a given time i.e. the throughput, concept of pipelining has been used.

4) Upon obtaining the combined data path, it is examined for various possible hazards.

5) Using the details of each recipe, a sequence of instructions have been planned for each recipe such that minimum data hazards is observed from the sequence.

6) Each unit required for the processor (memory, register file, ALU, control unit, hazard unit, and buffers) is designed and implemented in Verilog.

7) All are linked together to get a final code for the processor.

6.2 Hazards

6.2.1 Structural Hazards:

Special care has been taken into consideration to avoid structural hazards by using separate units. For the adder in the ALU and the adder in the incrementing the Program counter is different so that the resources are not blocked.

6.2.2 Control Hazards:

Control hazard has been avoided by using control signal to a unit when its function has to be performed and not when the control signal is generated.

Eg. In LOAD instruction, RegWrite3 is generated during ID stage but is only used during the WB stage when the data is written is the register file.

6.2.3 Data Hazards:

There are instances where the results are generated immediately required as an operand for the next instructions. This leads to data hazards.

6.3 Specifications

To prepare the recipes, storage of chocolate (that is available in limited quantity and hence its count is updated after every chocolate flavor is released.) is taken into consideration. It is assumed to have infinite milk storage in the vending machine. Hence while making recipes, milk count is not checked after every clock.

These are represented in main memory location as follows:

Main memory 000 is the updated count of chocolate parts available for use. This is used later while making the chocolate recipes. Also, the main memory 001 is used to store milk needed in some recipes.

In order to produce the chocolate with all the ingredients, it is important to standardize all the proportions. Here, R1 is assumed to be the mixing bucket (accumulator), which is where all the ingredients will be mixed. Register R2 is the current chocolate parts in the machine for the recipe. R3 represents one part of milk and R4 represents one part of the chocolate.

7. IMPLEMENTATION

In this paper, a vending Machine is designed which can be able to vend eight products viz. Gianduja, Cou-ver-ture, Dark Chocolate, Sweet Chocolate, Chocolate Caramel, Milk Chocolate, White Chocolate and Chocolate Praline; each with different recipes and costs. At every transition, money count which is an internal signal can be updated and this signal is also seven bits wide. The change will be returned through the change output signal when the inserted money is more than the total money of product clk and reset are also two input signals. The proposed machine will work on the positive edge of clock and when the reset button is pressed, machine will return to its initial state.

7.1 Data-path for Instructions

Table 2: General Instruction Format with assigned opcodes

Instructions	8-Bit Instruction format								Operations
	Opcodes			Reg		Memory			
Add	1	0	0			-	-	-	**Acc <- Acc + Reg**
Sub	1	0	1			-	-	-	**Acc <- Acc - Reg**
Mul	1	1	0			-	-	-	**{Reg, Acc} <- Acc*Reg**
Store	1	1	1						**[Mem] <- [Reg]**
Load	0	1	0						**[Reg] <- [Mem]**
MemI	0	1	1						**[Mem] <- 8'b111_1111**
RegI	0	0	1			I	M	M	**[Reg] <- 5'b0,IMM**
EDP	0	0	0	-	-	-	-	-	

7.2 TL Schematic of Main Processor:

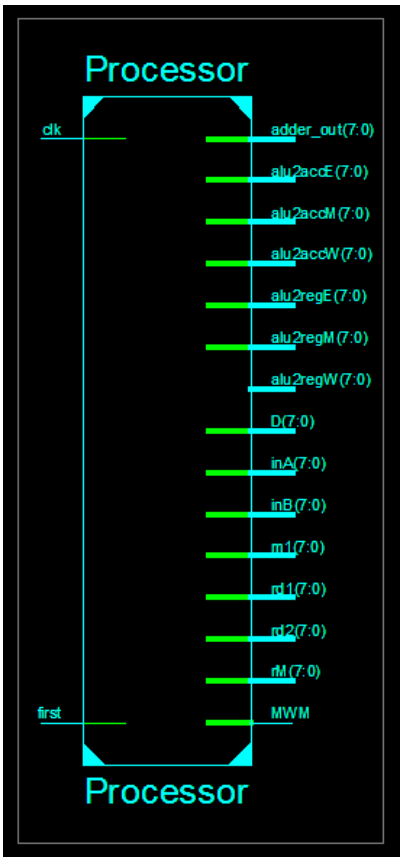

Figure 5: RTL Schematic diagram of the processor

RTL schematic shows a general idea of all the inputs and the outputs of the processor designed. Here, 'first' is one the input of the processor which will determine the recipe to be produced. Clk is the clock input provided to the processor. It is seen that multiple outputs are observed which are used for carrying out the advanced operations later on. Figure below shows the internal gate instantiation of the same processor, in much details.

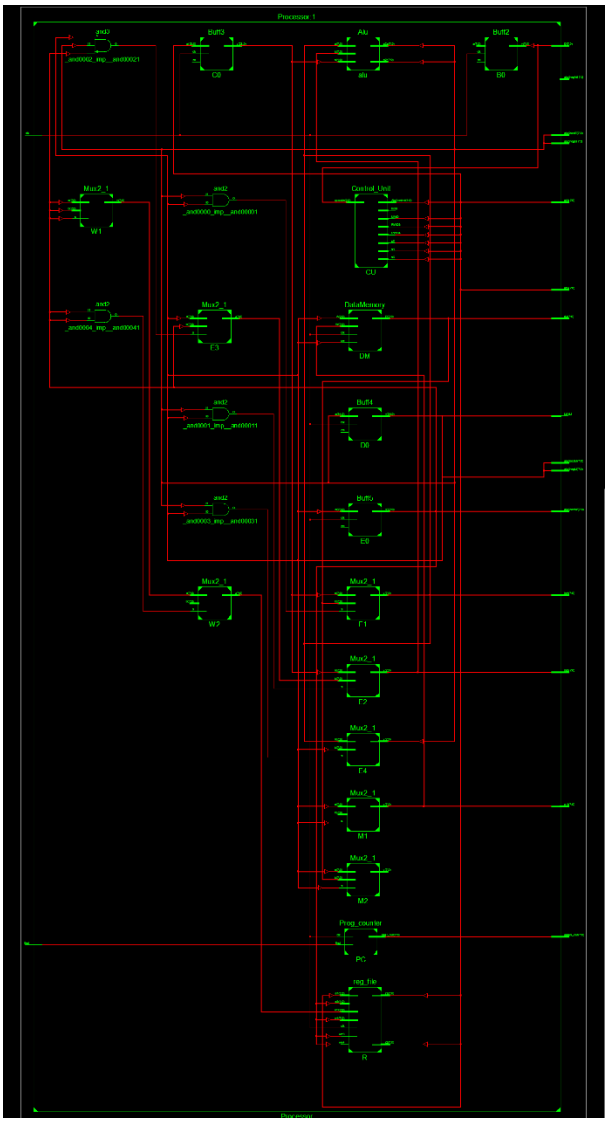

Figure 6: Gate level processor instantiation.

Gate level processor instantiation shows us the exact combination of all the physical connections in between the digital circuits. Here, since the entire processor is made up of various basic digital building blocks, the final product looks like the one shown above. It is important to note that this schematic diagram is of much use in finding out the time requirement, or in other words, how fast this processor works. Ideally, this is calculated with taking into consideration the worst possible path for the data to flow from point A to point B. Lesser the data path, faster is the processing power and vice versa.

7.3 Flowchart for 1st Recipe

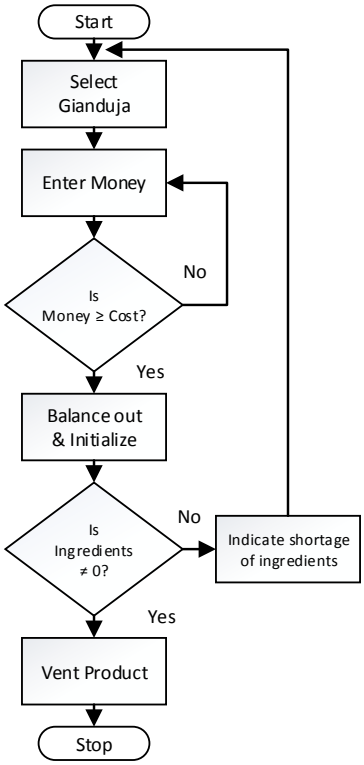

Figure 7: Flowchart for "Gianduja" recipe

Once the Vending machine is initialized, it would prompt the user for the desired chocolate. Here, let's take an example of the first chocolate – Gianduja. So, when the user selects Gianduja, the vending machine will prompt the user to enter the money. Assuming that the coin detection is already present inside the machine, it would detect the amount of money entered by the user. If the amount is equal or greater than the required amount (here Re.1) then the machine would first make the chocolate by the predetermined recipes and would produce the chocolate. If any balance money is pending, that would also be returned to the user. After every iteration, the processor of the vending machine will check the amount of ingredients. If any ingredients is low in quantity or completely used up, it would show the user a warning by blinking suitable LED's. The entire process will be repeated.

7.4 Simulation Results

All the results of each instruction along with the final "Vending machine" have been verified for their accuracy in the Xilinx ISE Software. Some of the simulation results for a few of the recipes can be seen below.

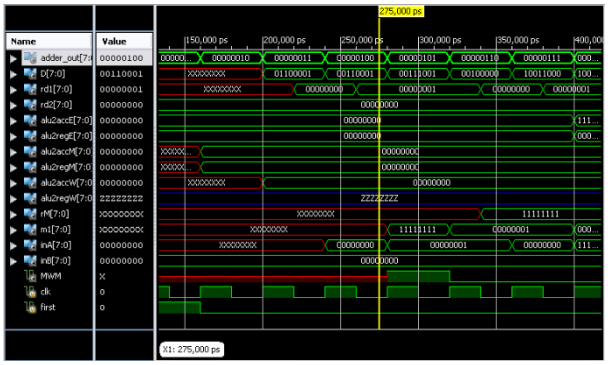

Figure 8: Test bench for "Gianduja" recipe

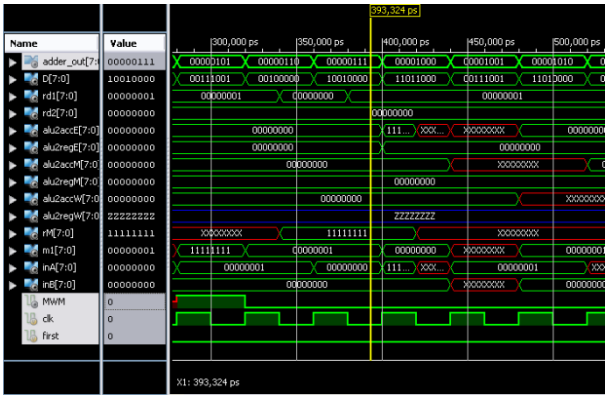

Figure 9: Test bench for "Sweet Chocolate" recipe

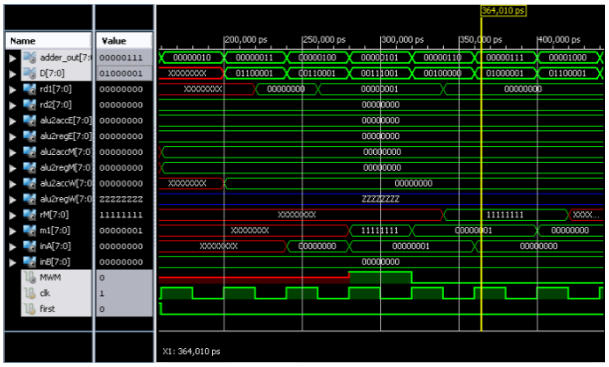

Figure 10: Test bench for "Chocolate Praline" recipe

8. TIMING REPORT

Table 3. Table showing timing report of each data path

Maximum Data Path	Total time
DM/Mram_mem8 to DM/R_7	2.384ns
PC/addr_update_0 to PC/addr_update_6	2.811ns
PC/addr_update_1 to PC/addr_update_6	2.672ns
PC/addr_update_3 to PC/addr_update_6	2.893ns
PC/addr_update_3 to PC/addr_update_5	2.847ns
PC/addr_update_0 to PC/addr_update_5	2.765ns
PC/addr_update_1 to PC/addr_update_5	2.626ns
PC/addr_update_2 to PC/addr_update_3	0.815ns
PC/addr_update_6 to PC/addr_update_6	0.815ns
PC/addr_update_0 to PC/addr_update_1	0.855ns
TOTAL	21.483ns

8.1 Timing Summary

Table above shows the timing report for the designed processor. It also portrays individual delay along the various digital circuits.

VLSI circuits can be designed for Area and speed constraints. Area constraints will ensure that the area requirement of the chip is kept to its minimum value. This would have been clearer from the RTL schematic of the same. On the other hand, VLSI circuits can also be designed for speed optimization. Meaning, it will focus more on achieving lesser timing delay, without considering space requirements. This processor was designed with speed optimization.

Constraints cover 37 paths, 0 nets, and 42 connections. Minimum period observed was 4.768ns with maximum frequency of 209.732MHz

Data path for Add Instruction.

Add (Acc ← Acc + Reg)

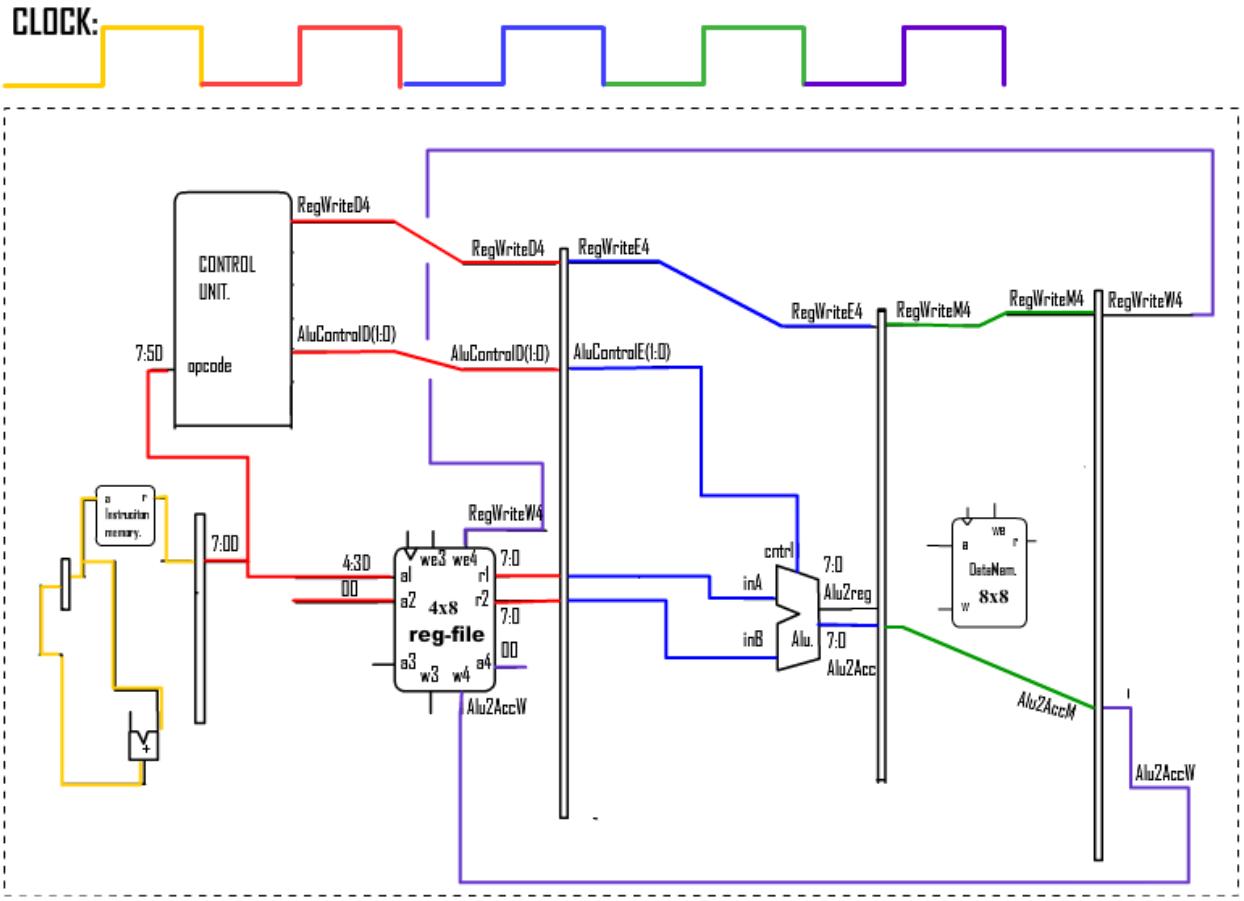

Figure 11: Detailed working of 'Add' Instruction

Table 3: Add Instruction Format

ID/RF stage		Exe stage (ALU)				Mem stage		WB stage	
Ctrl s/g generated [7:5]	Reg-File read stage	Input		Output		Input r/w	Ctrl s/g used	Reg-File write stage	Ctrl s/g used
		in A	in B	Alu2Acc	Alu2Reg				
RegWrite4	[4:3] to a1	R1		result	---	---		w4 (a4 hardwired to 00 =>result stored in Acc)	RegWrite4
	00 to a2 (Acc)		R2						

9. CONCLUSION

The present FPGA based 8 bit vending machine processor is implemented using FSMs with the help of Xilinx ISE Design Suite. This design is verified on the Spartan 3 development Board. State machines based vending Systems enhances productivity, reduces system development cost, and accelerates time to market. Also FPGA based vending machine give fast response and easy to use by an ordinary person. The designed machine can be used for many applications and number of selections can be easily enhanced. The next step from here would be actual implementation of the same project on a FPGA board. Also, other factors like the cost, power computation and reliability is to be tested.

10. REFERENCES

[1] Vending Machine - http://en.wikipedia.org/wiki/ Vending_machine

[2] Peter Minns, Ian Elliott, "FSM-based Digital Design using Verilog HDL", John Wiley & Sons, Ltd 2008.

[3] Xilinx Inc., Spartan 3 Data sheet: http://www.xilinx.com

[4] Peter Minns & Ian Elliott, "FSM-based Digital Design using Verilog HDL", John Wiley & Sons Ltd 2008.

[5] Phong P. Chu, "FPGA Prototyping Using Verilog HDL", John Wiley & Sons, Ltd 2008.

[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] 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.

[8] Peter Minns & Ian Elliott, "FSM-based Digital Design using Verilog HDL", John Wiley & Sons Ltd 2008.

[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.