Automatic Protein Shake Freestyle Vending Machine

Balaji Salunkhe
bs2051@rit.edu

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Master's Project is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
AUTOMATIC PROTEIN SHAKE FREESTYLE VENDING MACHINE

by
Balaji Salunkhe

GRADUATE PAPER

Submitted in partial fulfillment
of the requirements for the degree of
MASTER OF SCIENCE
in Electrical Engineering

Approved by:

__________________________
Mr. Mark A. Indovina, Lecturer
Graduate Research Advisor, Department of Electrical and Microelectronic Engineering

__________________________
Dr. Sohail A. Dianat, Professor
Department Head, Department of Electrical and Microelectronic Engineering

DEPARTMENT OF ELECTRICAL AND MICROELECTRONIC ENGINEERING
KATE GLEASON COLLEGE OF ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK
DECEMBER 2017
I would like to dedicate this thesis to my loving parents and all those people who always believed in me.
Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this graduate paper are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This graduate paper is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Balaji Salunkhe
Acknowledgements

I would like to acknowledge my advisor, Mark Indovina, for his valuable advice and direction and my start-up teammates for their continuous support.
Abstract

This paper discusses the design and implementation of an automatic protein shake freestyle vending machine. This machine is capable of providing protein shakes as per the customer’s requirement. Customers are able to decide what kind of protein supplement they want according to their fitness needs, and they can give the appropriate instructions to the machine to prepare their protein drink by use of a touch system. Other customers, who are unable to decide or new to fitness supplements can choose a pre-designed drinks according to their personal fitness. As such, the machine gives the freedom of selection to the customer if they doesn’t have any idea about their nutrition requirement.
# Contents

Contents viii

List of Figures vii

List of Tables viii

1 Introduction 1

2 Bibliographical Research 5

3 Idea Building 9
   3.1 Machine Flow chart ........................................ 9
      3.1.1 Option 1 - Customize your own shake ................ 11
      3.1.2 Option 2 - Pre-designed shake ......................... 12
      3.1.3 Payment .................................................. 12
   3.2 Finite State Machine (FSM) ................................. 13
      3.2.1 Mealy Machine .......................................... 13
      3.2.2 Moore Machine ......................................... 14
   3.3 FSM representation ........................................ 14

4 Design Description 17
   4.1 Customize your own shake .................................. 17
   4.2 Pre designed shake ......................................... 19
   4.3 Payment and Reorder ....................................... 21
   4.4 Watchdog .................................................... 23

5 Results 25
   5.1 FPGA Implementation ....................................... 25
   5.2 ASIC Benchmarking ......................................... 27

6 Conclusion 28
   6.1 Future Goals ................................................. 29
References 30

I Source Code I-1
I.1 Customize your shake module ........................................ I-1
I.2 Pre-designed shake module ........................................... I-26
I.3 Payment and Reorder module ....................................... I-51
I.4 Watchdog module ..................................................... I-60
I.5 Muscle Dispenser module ............................................ I-68

II Waveform II-1
List of Figures

3.1 Flow Chart ................................................................. 10
3.2 Mealy Machine ............................................................ 14
3.3 Moore Machine ............................................................ 14
3.4 FSM for protein shake vending machine .......................... 15

4.1 State Machine for Customize your shake ......................... 18
4.2 State Machine for Pre designed shakes .......................... 20
4.3 State machine for Payment and Reorder .......................... 22
4.4 Watchdog ................................................................. 24

5.1 Schematic view of Design ............................................... 26

II.1 Simulation Waveforms using the Xilinx Vivado Design Suite .... II-1
List of Tables

5.1 Xilinx Virtex 7 Implementation .............................................. 27
5.2 Comparison in ASIC technologies ............................................. 27
Chapter 1

Introduction

1.1 Vending Machines

In last few decades, the vending machine becomes an important thing in people’s day to day routine. The vending machine is recognized as a tool to aid modern and healthy living style. The vending machine is an auto self-serviced medium, which is able to cater or provide a customer’s requirement by pressing buttons. Request generated by the consumer is an actual instructions for a machine to operate. The mechanical and electrical circuitry performs all automatic work to eliminate manual work. Initially, the customer needs to decide what they want from a vending machine, as per that, by using a digital keypad or a touch system, they have to submit an order. The machine will generate the bill for the customer, as soon as the customer pays the bill by using currency, Debit or Credit cards, the machine will start dispensing process. The order placed by a customer is an input data for the machine. As a result, at the end of the process the customer obtains requested product from the machine. This is the general operation of the vending machine from start to end. The vending machines can be designed for specific functions like for beverages or food or other items. They offer portability, low cost, comfort, and also use less space for
setting up and can be installed in 24/7 service places. Vending machines discussed in reference papers are based on CMOS, SED and Micro-controllers technology[1–5].

The vending machine culture is, too, old as per the history. The first kind of vending machine was introduced in century 2-3 B.C. in Egypt. It was a mechanical machine which was created to sell water in Egypt. After that in the 19th century, vending machines for various functions were introduced during the industrial revolution in the United Kingdom[6]. The first ever commercialized vending machine for post-cards, which can be accessed through coins, was introduced in London.

1.2 Nutrition Supplementation[7]

1.2.1 Why is it needed?

In today’s world, people are becoming more careful about their health and physique. To achieve good physique and health, only workouts are not enough to achieve that level of fitness, people have to consume good quality nutritious food on proper intervals of time throughout the day, but in the real world, it’s really hard to get proper nutritious meals on time. In modern lifestyle, it is really difficult for a person to prepare food within no time without neglecting the nutrition factor. So another way to fulfill your daily requirement is through protein supplements. By a combination of solid foods and food supplements, one can achieve the proper nutrition fulfillment required for that perfect physique.

1.2.2 Varieties of Supplements

Supplements are categorized into meal replacement protein, muscle gaining protein, fat cutting protein, vegan protein, etc. Each protein category is designed with different ratios of proteins, carbohydrates, and fats which are the macros of our nutrition with some of
the micros of nutrition [8].

A) Meal replacement: This category stands for its name, it basically used in place of a meal. If in any case, a fitness enthusiast is going to miss a meal, this comes into action. One serving of meal replacement can give you enough calories to sustain for 2 to 4 hours. It usually contains a moderate amount of carbohydrates, a moderate amount of proteins and some amount of fats.

B) Mass gainer/ Muscle gainer: This category is a high calories nutrition category. Basically, it contains a high amount of carbohydrates and a moderate amount of proteins and fats. The ratio of carbohydrates-protein-fat varies along with the product.

C) Fat loss/ Maintenance: This category poses the purest form of protein nutrition. This is categorized into Whey, Isolate whey, and Hydrolyzed whey protein supplements. This is the very basic type which contains almost 80% to 100% of protein by weight per scoop. This is the most popular and widely used supplement type in the fitness world. This provides required protein blocks for muscle recovery.

D) Vegan: This category is for the vegan fitness enthusiast.

1.3 Objective:

The proposed protein supplement freestyle vending machine can offer a combination of affordability, convenience, and a variety of protein supplements for a niche market. The vending machine will act as a user-friendly catalyst for nutrition supplement distribution. The proposed paper is based on the FPGA (Field Programmable Gate Array) technology. FPGA based machines basically use minimum execution time, less power in comparison to micro-controller based designs[9]. FPGAs are reprogrammable, they allow for rapid prototyping, flexible, and reduce hardware in comparison with other designs[9].
1.4 Organization:

- Chapter 2: This chapter gives details about the articles, journals, papers and books which are used as a reference in building this project.

- Chapter 3: This chapter gives a brief explanation of idea or concept building with a flowchart. Global FSM of freestyle vending is described in detail.

- Chapter 4: Design implementation in parts described in this chapter. FSM for every major state is explained here.

- Chapter 5: This chapter outlines details about the benchmarking process on different technologies and implementation on FPGA.

- Chapter 6: This chapter concludes the project in detail with future work discussion.
Chapter 2

Bibliographical Research

[6] provides an introduction of unique services provided by the vending machine and also future possibilities in the advancement of vending machine culture. Vending machines can be installed in various fields and for various purposes. The vending machine installation is more convenient than a physical store, almost 120 times more convenient. The vending machines are called a “convenient unmanned shop”[6]. The joint venture of vending machine manufacturers and various commodities can enhance the user’s experience. Vending machines can be developed by using various technologies. They can be developed by using micro-controller, Application Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA), or Single Electron Devices (SED). All implementations are studied for this proposed paper.

[4] is giving an implementation of an automatic tea vending machine using SED. Single Electron Devices have some pros, like they consume low power, high density of integration, and switching power, that means they can do the operation faster. Because of these properties, SEDs are considered as the future of ULSI technology[4]. SEDs made up of SET (Single Electron Tunneling) technology where a single electron charge transfer on
principle of Coulomb Blockade and provide a new way to understand digital logic [3, 10]. Implementation of a coffee vending machine by using SED is proposed in detail in [3]. The Automated Beverage Dispenser (ABD) is discussed in [11] including management of the project. Under this report, the goal of ABD is to given as an efficient and cost-effective way of dispensing varieties of mixed drinks to their customer without any extra effort. The ABD provides freedom of customization to the customer, waiter, or other user so they can create the desired drink recipe. Also, the user can maintain a database for recipes and stock by using the menu bar.

The Automatic Beverage Dispenser based on Microcontroller technology, which is eliminates manual operation for dispensing, is proposed in [1, 5]. To eliminate the manual operation for measurement, this paper used cup sensors. It also stops the overflowing of liquids because of user negligence by using the electronic keypad to give user inputs for desired quantity. Assembly language is used for implementation and then compiled in assembler (MIDI – 51). This design is practically demonstrated on the AT89C52 microcontroller [1].

The FSM (Finite State Machine) vending machine with features of multi-select and auto-billing is implemented and demonstrated in [12]. The Mealy machine model with four states is created in this paper and successfully demonstrated on a Xilinx Spartan 3 FPGA board. An FPGA based machine is faster than CMOS based machine and SED based machine[12]. If using a micro-controller based machine, then the user will have to change the whole architecture even for small product changes[12]. Paper [9] is demonstrating the same design explained in [12] , but on a Xilinx Virtex 5 FPGA board.

A vending machine for frozen food is designed by using conceptual modeling in the paper[13, 14]. Conceptual modeling involves variables identification, finding simulation flows, and creating the simple formation of deliverable’s for easier understanding of the project. The combination of a process-oriented approach and a process flow logic is used
to construct the required model for this vending machine [13]. The motivation behind [15] is to develop a vending machine which can operate on low power. To achieve low power operation [15] has used various power reduction techniques, where the machine is tested on different frequencies to understand power consumption and also uses various IO standards to check best IO standard and frequency for power efficient vending machine. The comparison in a particular vending machine designed with 2 different FSM designs is explained in [16]. For comparison, the author considered the aspects like area, timing constraints, speed, and power dissipation. By comparing the results from 2 algorithms using the Altera Stratix FPGA family, the author is trying to find more reliable, more efficient and synthesizable designs for the vending machine.

Replenish system is one of the main features in this proposed paper. It is a smart feature which keeps the stock in order. If the products start hitting low-levels, the machine will start to take action for replenishing the products. The [2] is explaining an intelligent system to control the inventory of a variety of papers in the paper vending machine. This replenish system is implemented using fuzzy logic; the output of a fuzzy system represents 0-3 decimal value and each value posses status and action to take [2]. The real-time replenish system (RPS) with the help of GPRS and Wi-Fi technology is developed in [17, 18]. As per the proposed paper, the RPS can gather information, such as sales volume and inventory level, and send it back to the data center of the vending machine company. This way they can control and provide the quality service. This can help vending machine companies to cut operating costs for the vending machine[17, 19].

The ways of payment are the important application in the vending machine. There are generally 3 means of payment: coin/cash payment, card payment (i.e. Debit/Credit card payment), and mobile payment. [20, 21] proposes the idea of mobile payment by using IR (Infra Red technology) available in mobile; nowadays it is replaced by Bluetooth or Near
Field Communication (NFC) technology. In this process, the user will select an item and will connect with the vending machine through IR technology and then by taking help from the software running on the back-end server, the vending machine will complete the bill payment process [20, 21]. This paper is proposes the hybrid method for payment i.e. currency, manual mobile, and infrared mobile. In paper [22] the idea of payment through the Short Message Service (SMS) is discussed. After deciding the item from the vending machine, the user can make payment though SMS and can finish the shopping process.

Cleanliness is also a main concern in the vending machine industry. [23] is proposing a design with the help of actuators and sensors for the self-cleaning feature of a vending machine. This proposed design connects with the user through Bluetooth. The user adjust the coffee taste according to their preference and also can check the cleaning status of the vending machine. Actuators adjust materials for mixtures while sensors monitor the inner environment of the vending machine [23]. Speech recognition can be a great feature in the vending machine. To make vending machines more user-friendly paper [24] is proposing the self-service Ticket Vending Machine system. To execute this idea a Dynamic Time Warping (DTW) speech recognition algorithm is used.
Chapter 3

Idea Building

The idea is to develop a protein shake vending machine that works in a freestyle manner to give users the option of custom-made shakes for individuals who know their macro and micro requirements, and pre-calculated shakes for the individuals who don’t know their nutrition requirement but does know the physique goal. The paper explains the design of a vending machine that is versatile and easy for anybody to use.

3.1 Machine Flow chart

The flowchart in Fig. 3.1 explains the entire workings of the proposed protein shake vending machine. The machine will always remain in sleep mode until someone pushes the start button to access it. Once someone wants to access the machine, the machine comes out from sleep mode and prompts for option selection. There are two options for selection: 'Customize your shake' and 'Pre-designed shake'. Once an individual makes up their mind and selects one option, the process is forwarded to the next stage as shown which will eventually fulfill the nutritional requirement of an individual.
3.1 Machine Flow chart

Figure 3.1: Flow Chart
3.1.1 Option 1 - Customize your own shake

After selecting this option, the machine will ask for product selection from 6 different product types which have different nutritional profiles. Out of these 6 products, 2 products have the maximum level of different carbohydrate macros, 1 product has a high level of whey protein macro, 1 product has whey isolates protein macro, 1 product is full of casein protein macro, and 1 product is a vegan option. So, an individual can select 1 out of 6 product types at one time. This process will happen on the user experience level i.e. on screen, where an individual gives input to the machine after confirming nutritional profile. After confirming selection, the machine starts to work on the further process. The machine checks for product availability, if the product is not available then the machine will send a message to 'Replenish system’ section to take refilling action and at the same time it will send the customer back to the product selection screen to choose a different product instead of a currently selected product. If the product is available, i.e. it is passing the threshold level check for 0 level availability, then the machine will check for minimum 20% availability of product as 20% is the next threshold level for the products so no product will go out of availability. If product availability is more than 20%, then the machine will go to the purchase screen without taking any replenish action, else it will go to purchase screen after executing replenish action. On the purchase screen, the machine will give the option of ‘add more’ for adding more products in the currently selected product so to make it custom made for the different requirements of each individual. One individual can add 2 more products on top of the first selected product. For each selected product the machine repeats the same processes of availability check and replenish action.
3.1.2 Option 2 - Pre-designed shake

If an individual decides to go with this option, an user gets the category selection screen. The user has to select one category out of 4 pre-designed categories for 4 different goals: Muscle Gain, Meal Replacement, Weight Loss, and Vegan. Each category has at least 1 pre-designed shake option with different nutritional value. Each pre-designed shake is a combination of 2 different types of macros. Once an individual makes up their mind about the wanted category and desired pre-designed shake, the machine starts further steps of availability check. The machine checks for two different threshold levels for 1 product out of 2 products for a selected pre-designed shake. If the product availability is 0, then the machine will inform 'Replenish system' about this first, and then goes to category selection screen for an individual to go through other pre-designed shakes options. If the product is available, the machine will check for a threshold level check for 20%. If the product is below the threshold level of 20%, then the machine will inform this to 'Replenish system’, so the refilling process will start early. If it fails, then the machine will not send any message for refilling. After threshold check, either the product passes or fails, and the machine processes further to do availability check on a second product. It follows the same process like the first product.

3.1.3 Payment

After all the product selection processes described via option 1 or option 2, if the individual is satisfied with the detailed nutritional information and final price for a product, then the machine will move to the 'Payment' screen. If not, then the individual can either go back for other product selection process or can go to the cancel option. Under 'Payment' process, an individual can pay through cards or through mobile pay. The machine will
check for full expected payment, if full payment is received, the machine will create the desired protein shake and it will deliver it to the individual, else the machine will not go further until full payment is done. During this time, the machine will keep a copy of the finale product in its memory. So, if an individual wants to reorder same product, they don’t have to go through the same process again; the machine will access memory to get details about the finale product and will create the same product and will deliver it after full payment. If an individual doesn’t want to order the same product but they want to try something different then the machine will take them to the initial stage to start the entire process again. Once payment is recieved, the individual can’t cancel the process. Also, an individual can end the process after delivering product by pushing the exit button.

3.2 Finite State Machine (FSM)

The Finite State Machine representation is given in Figures 3.2 and 3.3. The FSM is a mathematical model used to design computer programs and sequential logic. The FSM is made up of numbers of state, the states come into work when there is expected inputs from the previous state. The outputs from current state drives next state. There are two types of FSM [25].

3.2.1 Mealy Machine

In a Mealy machine, the output signals are derived by the current state and the current inputs as shown in Fig. 3.2.
3.2.2 Moore Machine

In a Moore machine, the output signals are derived by the current state only as shown in Fig. 3.3.

3.3 FSM representation

As shown in Fig. 3.4, state machine always starts with Initiate state. Cust_shake (Customize your own shake), Pre_shake (Pre-designed shake), Payment, Deliver and Reorder are internal states. All these internal states also have sub-states within them. Those are discussed in a Chapter 4. After Initiate state, state machine moves into either Cust_shake or Pre_shake state depending upon the selection made by the individual. After the successful completion of current state of product selection, the machine flow goes into Payment state where the machine waits until full payment is done. Once the full payment is received the machine flow goes into Deliver state for delivering the expected product to the individual. The individual can also quit the payment process by sending the Cancel signal. After Deliver state, the machine goes to Reorder state, where the machine asks for reordering of product. If the individual doesn’t want to reorder, the machine will take flow to End state.
3.3 FSM representation

Figure 3.4: FSM for protein shake vending machine
Temp_mem is for the storing of the desired product so the machine can retrieve data for reordering purposes.
Chapter 4

Design Description

Fig. 4.1 presents the FSM state diagram for the proposed vending machine. This chapter explains in detail the design implementation of the FSM.

4.1 Customize your own shake

System reset or Watchdog reset resets the machine at the start as shown in Fig. 4.1. After the start, the state machine handler moves to the “Product_sel” state upon getting an option_sel1 signal, at this point the user can choose 1 of 6 products. Each product has different nutritional values and properties. Once the selection was made, the machine will check for availability of the product by moving to state “Threshold_check” where the machine checks for 2 types of threshold levels i.e. Product availability = 0 and Product availability <= 20%, where 0 and 20% are threshold levels. If the product clears the availability check, the machine moves to the purchase expectation state. If a product fails the availability check, the machine move to the replenish state. The “Replenish” state is divided into two sections. If the product is not available i.e. zero availability, the machine
4.1 Customize your own shake

![State Machine for Customize your shake]

Figure 4.1: State Machine for Customize your shake
will go to the “Replenish at 0” state where the machine will raise a signal for product unavailability and send a message for immediate action. If the product is available, but availability is below the threshold level of 20%, then the machine will raise a signal for low availability in the “Replenish at 20” state, the machine will send a message to that affect, and move to the Purchase expectation state. If an individual is satisfied with the choice, then the machine will give output as ‘ready1’ for further steps. If an individual wants to add more products, then the machine will save current product data in memory and go back to the product selection stage through the “Add_more_product” state to follow the selection process again. An individual can add two more products to the selection process for a maximum 3 products per transaction. An individual can cancel the purchase process if they don’t want to purchase by selecting Cancel.

4.2 Pre designed shake

Similar to the Cust shake module, this module also initially gets System reset or Watchdog reset to reset state machine as shown in Fig. 4.2. After, the state machine moves to “Category_sel” state when it gets an option_sel2 signal after the start. There are four categories in Category_sel state. These categories are: Meal replacement, Muscle gainer, Weight loss, and Vegan. An individual can select one category of choice as per requirement to move forward in the process. Each category contains a number of pre-designed shakes with different nutrition profiles and properties. As an individual selects one of the pre-designed shakes from the selected category, the machine starts to check the availability of the products required by each pre-designed shake. Each pre-designed shake is made up of two different products which have different nutritional profiles and functions. If either of the products are not available then the machine will take the individual back to the
Figure 4.2: State Machine for Pre designed shakes
previous state, i.e. category select, and raise the signal for test fail and send it to replenish state i.e “Replenish at 0” state. If products are available, then the machine will check for threshold level test, if product availability is above the threshold level, i.e. 20%, then the machine move to the “Purchase” state directly, else it will raise a signal for replenishing system first in the “Replenish at 20” state, and then it will move forward to purchase expectation state. If the amount is reasonable for an individual, then the machine will send a ‘ready2’ signal out for further process, else the individual can cancel the process.

4.3 Payment and Reorder

The payment and reorder module is the final state for the completion of the protein shake making process. As shown in Fig. 4.3, this module starts when it gets the signal ’ready1’ from Customize your shake module or signal ’ready2’ from Pre-designed shake module. The next state is “Pay_option” where the customer can decide payment method. There are two types of payment methods, one is Card payment and the other is Mobile payment. Card payment is a generic payment method and Mobile payment is for mobile application customers or members. After the payment method selection, the customer has to make full payment or the machine will not move ahead for the protein shake mixing process. Once the full payment is received, the machine will start the mixing process and once finished the customer will receive the desired protein shake. At this stage the machine will ask for a reorder. If a customer wants to reorder the same protein shake, then they just have to push the Same product button, and the machine will retrieve the memory for stored data so the customer doesn’t have to redo the process. If the customer wants to try something new, then they can push Different product button, at which point the machine will take the customer to the Option selecting process, where the customer can choose from a custom
Figure 4.3: State machine for Payment and Reorder
or pre-designed shake. If the customer doesn’t want to reorder, they only have to select the cancel button to exit the machine.

4.4 **Watchdog**

The function of the watchdog module is to check for customer response. If the customer is using the machine then the watchdog starts for the counter as shown in Fig. 4.4. For each stage in the vending process, the counter reset. If at any point the customer leaves the machine in the middle of a vending process, then watchdog will wait for pre-defined period of time and eventually resets the machine. So, on arrival of wait signal from any state such as “customize your shake” or “pre-designed shake”, or cancel signals, or the add more signal, the watchdog starts its functioning. Watchdog starts counting for 20 counts, so whenever the above-mentioned signals come to watchdog, it resets the counter to zero and starts counting for 20 counts. If all conditions for counter_20 fails, then the state machine goes on counter_30 counter, where the watchdog starts to count for 30 counts. At this point the watchdog checks for reordering stage, if the customer raises some signal in reordering and left then the watchdog counter resets the machine after 30 counts. In a worst case scenario, the watchdog moves forward to counter_40 where watchdog counts for 40 counts and it forcefully resets the entire system by sending out “wrst” signal.
Figure 4.4: Watchdog
Chapter 5

Results

5.1 FPGA Implementation

The required RTL for the automatic protein shake vending machine is designed using the Verilog Hardware Description Language (HDL). The tools used were the Xilinx Vivado Design Suite targeting a Xilinx Virtex 7 FPGA board. The standard flat test-bench code for behavioral verification was also developed in Verilog HDL. The design was tested under various circumstances to check its flexibility and reliability, and the design has successfully gave expected outputs. The design successfully cleared the synthesis and gate level simulation. A top level schematic for vending machine RTL is shown in the Fig. 5.1.

Tab.5.1 tabulates the power, timing and device utilization for Xilinx Virtex 7 FPGA. The total power mentioned is the addition of total dynamic power and total leakage power. The design is has a worst case slack of $6\text{ns}$ for a maximum operating frequency of $167\text{MHz}$. The design utilizes $18\%$ of device, where the FF are taking $1\%$, LUTs using $1\%$, $13\%$ is allocating for I/O, and the $3\%$ is utilizing by BUFG.
Figure 5.1: Schematic view of Design
5.2 ASIC Benchmarking

For ASIC benchmarking, the design is simulated and synthesized by using the Cadence and Synopsys tools respectively. Tab. 5.2 tabulates the area, power and timing for the various of target technologies. Results for the TSMC 180nm, TSMC 65nm, SEDK 90nm, and SEDK 32nm technologies are mentioned in the table.

Out of all technologies, the 180nm technology has largest cell area as compared to rest of the technologies, i.e. 8133um\(^2\). In 90nm technology, the design requires 4552um\(^2\)area, and a total power consumption of 21uW, the lowest power requirement among all technologies. 65nm technology requires the least cell area, i.e. 1118um\(^2\). Total power is the addition of the total dynamic power and the total leakage power and is measured in micro-watts. 180nm technology consumes the highest power, but it is using the smallest leakage power. Due to not having to over constraint the design, all technologies have almost the same timing. Out of all, the 65nm technology is the fastest design.

On the basis of area, power consumption, and timing the physical design implementation in 90nm is most efficient.

### Table 5.2: Xilinx Virtex 7 Implementation

<table>
<thead>
<tr>
<th>Properties / Technologies</th>
<th>Power (Dynamic + Leakage)</th>
<th>Timing (slack)</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xilinx Virtex 7</td>
<td>0.246 W</td>
<td>6 ns</td>
<td>18%</td>
</tr>
</tbody>
</table>

### Table 5.2: Comparison in ASIC technologies

<table>
<thead>
<tr>
<th>Properties / Technologies</th>
<th>Area</th>
<th>Power (Dynamic + Leakage)</th>
<th>Timing (slack)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 nm</td>
<td>1140um(^2)</td>
<td>120uW</td>
<td>18.5138ns</td>
</tr>
<tr>
<td>65 nm</td>
<td>1118um(^2)</td>
<td>26uW</td>
<td>18.5924ns</td>
</tr>
<tr>
<td>90 nm</td>
<td>4552um(^2)</td>
<td>21uW</td>
<td>18.1832ns</td>
</tr>
<tr>
<td>180 nm</td>
<td>8133um(^2)</td>
<td>164uW</td>
<td>18.4461ns</td>
</tr>
</tbody>
</table>
Chapter 6

Conclusion

An Automatic Protein Shake Freestyle Vending Machine is successfully designed and implemented in this paper. The idea is to provide the highly nutritional meals to the fitness enthusiast at any time at any place. This vending machine is able to provide four types of the pre-designed sports nutrition shakes to the consumers, categories are meal replacement shakes, weight loss shakes, mass gaining shakes, and vegan shakes. Also, the vending machine gives the consumer authority to create their own protein shake as per their requirements.

A Finite State Machine (FSM) is used to design automatic protein shake freestyle vending machine. The FSM is made modular for future design improvements. This paper gives detail descriptions about all mid-level and low-level modules with the main top level module. Verilog HDL is used for the final RTL design. The final design is implemented on Xilinx Virtex 7 FPGA board by using Xilinx Vivado tool. The design consumes 0.246W power to run, with the 18% of hardware utilization.

A vending machine is also benchmarked in ASIC for 32nm, 65nm, 90nm, and 180nm technologies. The benchmarking criteria for comparison are total cell area, total power
consumption, and timing results for all these mentioned technologies. The 65\textit{nm} technology is using least area, i.e. $1118\,\text{\mu m}^2$. The power consumed by design in 90\textit{nm} technology is the most efficient, i.e. $21\,\text{\mu W}$ out of the all physical design implementations. On timing analysis results the physical design on 90\textit{nm} technology is the fastest, i.e. $18.1832\,\text{ns}$. On overall comparison the physical design implementation on 90\textit{nm} technology is the best and most efficient among all physical design implementations on trade off area allocation.

\section{Future Goals}

Automatic Protein Shake Freestyle Vending Machine has good future scope. Some of the possible future goals are mentioned.

- The real-life prototyping of the design is the most important future goal.

- The design in the paper is verified by using the flat testbench. A more detailed SystemVerilog testbench environment can be a future improvement.

- Speech recognition can be added to the system to operate the vending machine. This would be a great feature for disabled people.

- A more sophisticated self-cleaning feature can be added as a future improvement.

- For a more reliable and efficient replenishing system, the vending machine is going to need communicate to remote servers in a data center.
References


[14] Y. Wang, O. K. Ho, G. Q. Huang, and D. Li, “Study on vehicle management in


[21] S. Z. Azami, N. Torabi, and M. Tanabian, “Modeling the customer behavior in the mobile payment on a non-connected vending machine platform,” in *Electrical and


Appendix I

Source Code

I.1 Customize your shake module

```
'timescale 1ns / 1ps

//
///////////////////////////////////////////////////////////////////////
// Author: Balaji Salunkhe.
// Module Name: cust_shake
// Project Name: Muscle Dispenser
// Advisor: Mark Indovina
//
///////////////////////////////////////////////////////////////////////////////

'define Product_sel 3'b001
'define Threshold_check 3'b010
```
I.1 Customize your shake module

#define Replenish_at0 3'b011
#define Replenish_at20 3'b100
#define Purchase_exp 3'b101
#define Add_more_prod 3'b110

module cust_shake (srst, wrst, clk, option_sel1, wait_prod1, sel, avl1, avl2, avl3, avl4, avl5, avl6, ack, cost_check, ready1, cancell1, add_more);

I.1 Customize your shake module

input

clk , //System clock

srst , //System reset

wrst ; //watchdog reset

input

option_sel1 , //Option selection signal

cancel1; //Exit from current process

input [7:0] avl1 , avl2 , avl3 , avl4 , avl5 , avl6; //Product availability check

input [2:0] sel; //Product # selection

input

cost_check , // total Cost of product

add_more; // For adding one more scoop

reg [1:0] counter;

reg more;

reg [5:0] checkfail; //product availability

checkfail status
output reg [5:0] ack; //Acknowledge signal for checkfail
   status

output reg ready1;        //Ready signal for payment stage

output reg wait_prod1;

reg [2:0] state1;       //States

reg [2:0] temp_bit;    //to save product number

reg [8:0] Temp_mem; //To save entire order up to 3 products for 
                      re-order

reg [5:0] prod;

parameter Threshold_limit = 8’d20;

always @(posedge clk)
begin
  if (srst == 1 || wrst == 1)
    begin
      ready1 = 1’b0;
    end
end

checkfail = 6'b000000;
ack = 6'b000000;
Temp_mem = 9'b000000000;
counter = 1;
temp_bit = 3'b000;
state1 = 3'b000;
prod = 6'b000000;
end
else if (option_sel1 == 1)
begin
    //display ('n Customize your own shake 
    n');
case (state1)
    'Product_sel:
        begin
        case (sel)
        3'b001:
        begin
        display ('Product 1 selected ');
temp_bit = 3'b001;
prod = 6'b100000;
wait_prod1 = 1'b1;
state1 <=
        'Threshold_check;
3'b010:
begin
$display ("Product 2 selected");
temp_bit = 3'b010;
prod = 6'b010000;
wait_prod1 = 1'b1;
state1 <= 'Threshold_check;
end

3'b011:
begin
$display ("Product 3 selected");
temp_bit = 3'b011;
prod = 6'b001000;
wait_prod1 = 1'b1;
state1 <= 'Threshold_check;
end

3'b100:
begin
  $display ("Product 4
           selected");
  temp_bit = 3'b100;
  prod = 6'b000100;
  wait_prod1 = 1'b1;
  state1 <= 'Threshold_check;
end

3'b101:
begin
  $display ("Product 5
           selected");
  temp_bit = 3'b101;
  prod = 6'b000010;
  wait_prod1 = 1'b1;
  state1 <= 'Threshold_check;

// state <=
  'Purchase_exp;
end

3'b110:
begin
    $display ("Product 6 selected");
    temp_bit = 3'b110;
    prod = 6'b000001;
    wait_prodl = 1'b1;
    state1 <= 'Threshold_check;
end

default:
    begin
        $display ("Error in Selection");
        state1 <= 'Product_sel;
    end
endcase
end

‘Threshold_check:
    begin
        case (prod)
            6'b100000:
                if (avl1 == 0)
                    begin
                        // Code here
                    end
        endcase
    end
checkfail = 6'b100000;

`display("Product is not available. Please try different product."");
temp_bit = 3'b000;
state1 <= 'Replenish_at0;
end
else if (avl1 <
Threshold_limit)
begin
  checkfail = 6'b100000;
  state1 <=
    'Replenish_at20;
end
else
begin
  state1 <=
    'Purchase_exp;
end
6'b010000;
if (avl2 == 0)
begin
  checkfail = 6'b010000;
  $display("Product is not available.
  Please try different product.'
);
  temp_bit = 3'b000;
  state1 <= 'Replenish_at0;
end
else if (avl2 < Threshold_limit)
begin
  checkfail = 6'b010000;
  state1 <= 'Replenish_at20;
end
else
begin
  state1 <= 'Purchase_exp;
end
6'b001000:

    if (avl3 == 0)
        begin
            checkfail = 6'b001000;

            $display("Product is not available. Please try different product.");

            temp_bit = 3'b000;
            state1 <= 'Replenish_at0;
        end

    else if (avl3 < Threshold_limit)
        begin
            checkfail = 6'b001000;
            state1 <= 'Replenish_at20;
        end

    else
        begin

        end
state1 <= 'Purchase_exp;
end

6'b000100:
if (avl4 == 0)
begin
checkfail = 6'b000100;

$display("Product is not available.
Please try different product.");
temp_bit = 3'b000;
state1 <= 'Replenish_at0;
end
else if (avl4 < Threshold_limit)
begin
checkfail = 6'b000100;
state1 <= 'Replenish_at20;
end
else
begin
    state1 <=
        'Purchase_exp;
end

6'b000010:
if (avl5 == 0)
begin
    checkfail = 6'b000010;
    $display("Product is not available.
Please try different product.");
    temp_bit = 3'b000;
    state1 <=
        'Replenish_at0;
end
else if (avl5 < Threshold_limit)
begin
    checkfail = 6'b000010;

I.1 Customize your shake module

state1 <=

'Replenish_at20;

end
else
begin
  state1 <=
  'Purchase_exp;
end

6'b000001:
if (avl6 == 0)
begin
  checkfail = 6'b000001;

  $display("Product is not available. Please try different product.'
  );
  temp_bit = 3'b000;
  state1 <=
  'Replenish_at0;
end
else if (avl6 <
  Threshold_limit)
begin
    checkfail = 6'b000001;
    state1 <= 'Replenish_at20;
end
else
    begin
        state1 <= 'Purchase_exp;
    end

default:
    begin
        state1 <= 'Product_sel;
    end
endcase
end

-Replenish_at0:
    begin
        case (checkfail)
            6'b100000:
                begin
                    ack = 6'b100000;
                    ready1 = 1'b0;
                end
        endcase
    end
$display (*
  Acknowledgement sent;
  Refil the Product 1
  ASAP*);
state1 <= `Product_sel;
end

6'b010000:
begin
  ack = 6'b010000;
  ready1 = 1'b0;
  $display (*
    Acknowledgement sent;
    Refil the Product 2
    ASAP*);
  state1 <= `Product_sel;
end

6'b001000:
begin
  ack = 6'b001000;
  ready1 = 1'b0;
  $display (*
    Acknowledgement sent;
    Refil the Product 3
I.1 Customize your shake module

6'b000100:

begin
  ack = 6'b000100;
  ready1 = 1'b0;
  $display (*
    Acknowledgement sent;
    Refill the Product 4
    ASAP*);
  state1 <= 'Product_sel;
end

6'b000010:

begin
  ack = 6'b000010;
  ready1 = 1'b0;
  $display (*
    Acknowledgement sent;
    Refill the Product 4
    ASAP*);
  state1 <= 'Product_sel;
end

6'b0000010:

begin
  ack = 6'b000010;
  ready1 = 1'b0;
  $display (*
    Acknowledgement sent;
    Refill the Product 4
    ASAP*);
  state1 <= 'Product_sel;
end
6'b000001:
   begin
      ack = 6'b000001;
      ready1 = 1'b0;
      $display (''
         Acknowledgement sent;
         Refill the Product 6
         ASAP'');
      state1 <= 'Product_sel;
   end
default:
   begin
      state1 <= 'Product_sel;
   end
endcase
end

'Replenish_at20:
   begin
      case(checkfail)
      6'b100000:
         begin
            ack = 6'b100000;
         end
      esac...
$display ('Acknowledgement
 sent; Refil the
 Product 1 ASAP');

state1 <=
  'Purchase_exp;
end

6'b010000:
begin
  ack = 6'b010000;
  $display ('Acknowledgement
 sent; Refil the
 Product 2 ASAP');

state1 <=
  'Purchase_exp;
end

6'b001000:
begin
  ack = 6'b001000;
  $display ('Acknowledgement
 sent; Refil the
Product 3 ASAP');
state1 <=
    'Purchase_exp;
end

6'b000100:
begin
  ack = 6'b000100;
  $display ('Acknowledgement
            sent; Refil the
            Product 4 ASAP');
state1 <=
    'Purchase_exp;
end

6'b000010:
begin
  ack = 6'b000010;
  $display ('Acknowledgement
            sent; Refil the
            Product 4 ASAP');
state1 <=
    'Purchase_exp;
end
6’b000001:

begin

ack = 6’b000001;

$display ('Acknowledgement sent; Refil the Product 6 ASAP');

state1 <= 'Purchase_exp;

end

default:

begin

state1 <= 'Product_sel;

end
endcase
end

'Purchase_exp:

begin

$display ('\n Detailed nutritional info xxxxxxxxxx');
$display ("Total cost of product = xxxxxx");
$display ("\n Products selected: %d", counter);
wait_prod1 = 1'b0;
if (add_more)
    begin
        $display ("Yes, want to add more product");
        Temp_mem [2:0] = temp_bit;
        temp_bit = 3'b000;
        counter = counter + 1;
        state1 <= 'Add_more_prod;
    end
else if (cost_check)
    begin
        Temp_mem [2:0] = temp_bit;
        ready1 = 1'b1;
        state1 <= 'Purchase_exp;
I.1 Customize your shake module

end

else if (cancel1)
begin
$display ('Request canceled ');
temp_bit = 3'b000;
state1 <= 'Purchase_exp;
end

else
begin
state1 <= 'Purchase_exp;

// 'Product_sel;
end

end
d ‘Add_more_prod:
begin
if (counter > 3)
begin
$display ('Highest limit has reached')
I.1 Customize your shake module

ready1 = 1;

else

begin

Temp_mem = {Temp_mem[5:0], 3'b000};

state1 <= 'Product_sel;

end

end

default:

begin

state1 <= 'Product_sel;

end

endcase

end

else

begin

// $display ('Error from Customize your own shake');

end

end
I.1 Customize your shake module

endmodule
I.2 Pre-designed shake module

`timescale 1ns / 1ps

//

///////////////////////////////////////////////////////////////////////////////

// Author: Balaji Salunkhe.
// Module Name: pre_design
// Project Name: Muscle Dispenser
// Adviser: Mark Indovina

///////////////////////////////////////////////////////////////////////////////

`define Category_sel 3'b001
`define Threshold_check 3'b010
`define Replenish_at0 3'b011
`define Replenish_at20 3'b100
`define Purchase_exp 3'b101

module pre_design(
    srst ,
    wrst ,
    clk ,
    option_sel2 ,
);
I.2 Pre-designed shake module

```verilog
wait_prod2,
sel,
avl1,
avl2,
avl3,
avl4,
avl5,
avl6,
ack,
cost_check,
ready2,
cancel2
);

input
srst,    //System reset
wrst,    //Watchdog reset
clk;      //System clock

input
option_sel2,    //Option selection signal
cancel2;        //Exit from current process

input [7:0] avl1, avl2, avl3, avl4, avl5, avl6;  //Product availability check
```
I.2 Pre-designed shake module

input [2:0] sel; //Product # selection

input cost_check; // total Cost of product

reg [5:0] checkfail;

reg [6:0] prod; //mr1, mr2, mg1, mg2, wl1, wl2, veg;

output reg [5:0] ack;

output reg ready2; //Ready signal for payment stage

output reg wait_prod2;

reg [2:0] state1; //States

parameter Threshold_limit = 7'd20;

always @(posedge clk)
  begin
    if (srst == 1 || wrst == 1)
      begin
        ready2 = 1'b0;
      end
checkfail = 6'b000000;
ack = 6'b000000;
state1 = 3'b000;
prod = 6'b000000;

else if (option_sel2 == 1)
begin

  // $display ('\n Select from Pre-Designed
shakes');
  case (state1)
    'Category_sel:
      begin
        case (sel)
          3'b001:
            begin

              $display ('Meal
replacement 1 selected
');

              prod = 7'b1000000; //mr1
              = 1;

              wait_prod2 = 1'b1;

              state1 = 'Threshold_check
;

            end

          end
        end
      end

end
I.2 Pre-designed shake module

3’b010:
begin
$display("Meal replacement 2 selected ");
prod = 7’b0100000; //mr2 = 1;
wait_prod2 = 1’b1;
state1 <= 'Threshold_check;
end

3’b011:
begin
$display("Muscle gainer 1 selected ");
prod = 7’b0010000; //mg1 = 1;
wait_prod2 = 1’b1;
state1 <= 'Threshold_check;
end

3’b100:
begin
$display("Muscle gainer 2 selected");
prod = 7'b0001000; //mg2 = 1;
wait_prod2 = 1'b1;
state1 <= 'Threshold_check;

end

3'b101:
begin
$display("Weight loss 1 selected");
prod = 7'b0000100; //wl1 = 1;
wait_prod2 = 1'b1;
state1 <= 'Threshold_check;

//state <=
'Purchase_exp;
end

3'b110:
begin
$display("Weight loss selected");
prod = 7'b0000010; //wl2 = 1;
wait_prod2 = 1'b1;
state1 <= 'Threshold_check;
end

3'b111:

begin
$display("Vegan selected");
prod = 7'b0000001; //veg = 1;
wait_prod2 = 1'b1;
state1 <= 'Threshold_check;
end

default:
begin
$display("Error in Selection");
state1 <= 'Category_sel;
end
end
endcase
end

`Threshold_check:
begin
  case (prod)
    7'b1000000:
      if (avl2 == 0)
        begin
          checkfail = 6'b010000;
          $display("Ingredient is not available, please select new product");
          state1 <= `Replenish_at0;
        end
      else if (avl4 == 0)
        begin
          checkfail = 6'b000100;
        end
  endcase
end
$display ('Ingredient is not available, please select new product');
state1 <= 'Replenish_at0;
end
else if (avl2 < Threshold_limit)
begin
  checkfail = 6'b010000;
  state1 <= 'Replenish_at20;
end
else if (avl4 < Threshold_limit)
begin
  checkfail = 6'b000100;
  state1 <= 'Replenish_at20;
end
else
begin
state1 <=
    'Purchase_exp;
end

7'b0100000:
if (avl3 == 0)
begin
    checkfail = 6'b001000
    ;
    $display ('Ingredient is not available, please select new product');

    state1 <=
    'Replenish_at0;
end
else if (avl5 == 0)
begin
    checkfail = 6'b000010
    ;
    $display ('Ingredient is not available, please select new product');
state1 <= 'Replenish_at0;

end
else if (avl3 < Threshold_limit) begin
  checkfail = 6'b001000;
  state1 <= 'Replenish_at20;
end
else if (avl5 < Threshold_limit) begin
  checkfail = 6'b000010;
  state1 <= 'Replenish_at20;
end
else begin
  state1 <= 'Purchase_exp;
end
7'b0010000:

    if (avl1 == 0)
    begin
        checkfail = 6'b100000 ;
        $display ("Ingredient is not available, please select new product");
        state1 <= 'Replenish_at0 ;
    end

else if (avl3 == 0)
begin
    checkfail = 6'b001000 ;
    $display ("Ingredient is not available, please select new product");
    state1 <= 'Replenish_at0 ;
end
else if (avl1 < Threshold_limit)
begin
    checkfail = 6'b100000;
    state1 <= 'Replenish_at20;
end
else if (avl3 < Threshold_limit)
begin
    checkfail = 6'b001000;
    state1 <= 'Replenish_at20;
end
else
begin
    state1 <= 'Purchase_exp;
end

7'b0001000:
if (avl2 == 0)
begin
checkfail = 6'b010000
;
$display ('Ingredient is not available, please select new product');
state1 <= 'Replenish_at0;
end

else if (avl3 == 0)
begin
  checkfail = 6'b001000;
  $display ('Ingredient is not available, please select new product');
  state1 <= 'Replenish_at0;
end

else if (avl2 < Threshold_limit)
begin
  checkfail = 6'b010000;
  }
I.2 Pre-designed shake module

    state1 <= 'Replenish_at20;
    end

else if (avl3 < Threshold_limit)
    begin
        checkfail = 6'b001000;
        state1 <= 'Replenish_at20;
    end

else
    begin
        state1 <= 'Purchase_exp;
    end

7'b0000100:

    if (avl3 == 0)
    begin
        checkfail = 6'b001000;
        $display ('Ingredient is not available, please select new
I.2 Pre-designed shake module

```verilog
7'b0000010:
    if (avl4 == 0)
        begin
            checkfail = 6'b000100
        end

else if (avl3 <
    Threshold_limit)
begin
    checkfail = 6'b001000
    state1 <=
        'Replenish_at20;
end
```

```verilog
7'b0000010:
    if (avl4 == 0)
        begin
            checkfail = 6'b000100
        end
```

```verilog
state1 <=
    'Replenish_at0;
end
```

```verilog
else if (avl3 <
    Threshold_limit)
begin
    checkfail = 6'b001000
    state1 <=
        'Replenish_at20;
end
```

```verilog
else
    begin
        state1 <=
            'Purchase_exp;
    end
```

```verilog
7'b0000010:
    if (avl4 == 0)
        begin
            checkfail = 6'b000100
        end
```

```verilog
end
```
$display ('Ingredient is not available, please select new product');

state1 <= 'Replenish_at0;
end
else if (avl4 < Threshold_limit)
begin
  checkfail = 6'b000100;
  state1 <= 'Replenish_at20;
end
else
begin
  state1 <= 'Purchase_exp;
end

7'b0000001:
if (avl6 == 0)
begin
checkfail = 6'b000001;

$display ("Ingredient is not available, please select new product");

state1 <= 'Replenish_at0;
end

else if (avl6 < Threshold_limit)
begin
    checkfail = 6'b000001;
    state1 <= 'Replenish_at20;
end
else
begin
    state1 <= 'Purchase_exp;
end

default:
state1 <= 'Category_sel;
    endcase
end

'Codecase':
begin
  case (checkfail)
  6'b100000:
    begin
      ack = 6'b100000;
      $display("Acknowledgement sent; Refill the Product 1 ASAP");
      state1 <= 'Category_sel;
    end
  6'b010000:
    begin
      ack = 6'b010000;
      $display("Acknowledgement sent; Refill the Product 2 ASAP");
      state1 <= 'Category_sel;
    end
  6'b001000:
    begin

  end
ack = 6'b001000;
$display ("Acknowledgement sent; Refill the Product 3 ASAP");
state1 <= 'Category_sel;
end

6'b000100:
begin
  ack = 6'b000100;
  $display ("Acknowledgement sent; Refill the Product 4 ASAP");
  state1 <= 'Category_sel;
end

6'b000010:
begin
  ack = 6'b000010;
  $display ("Acknowledgement sent; Refill the Product 5 ASAP");
  state1 <= 'Category_sel;
end

6'b000001:
begin
  ack = 6'b000001;
I.2 Pre-designed shake module

```verilog
$display("Acknowledgement sent; Refil the Product 6 ASAP");
state1 <= 'Category_sel;
end

default:
state1 <= 'Category_sel;
endcase

// $display("Ack pre_d = %b", ack);
end

'Replenish_at20:
begin
case(checkfail)
6'b100000:
begin
ack = 6'b100000;
$display("Acknowledgement sent; Refil the Product 2 ASAP");
state1 <= 'Purchase_exp;
end

6'b010000:
begin
ack = 6'b010000;
```

```
$display ('Acknowledgement sent; Refil the Product 2 ASAP');
state1 <= 'Purchase_exp;
end

6'b001000:
begin
ack = 6'b001000;
$display ('Acknowledgement sent; Refil the Product 3 ASAP');
state1 <= 'Purchase_exp;
end

6'b000100:
begin
ack = 6'b000100;
$display ('Acknowledgement sent; Refil the Product 4 ASAP');
state1 <= 'Purchase_exp;
end

6'b000010:
begin
ack = 6'b000010;
end

6'b000010:
begin
ack = 6'b000010;
end
$display ("Acknowledgement sent; Refill the Product 5 ASAP");
state1 <= 'Purchase_exp;
end

6'b000001:
begin
ack = 6'b000001;
$display ("Acknowledgement sent; Refill the Product 6 ASAP");
state1 <= 'Purchase_exp;
end

default:
state1 <= 'Category_sel;
endcase
end

'Purchase_exp:
begin
$display ("\n Detailed nutritional info xxxxxxxxxxxx");
$display ("Total cost of product = xxxxxx");
wait_prod2 = 1'b0;

if (cost_check)
  begin
    ready2 = 1'b1;
    state1 = 'Purchase_exp;
  end
else if (cancel2)
  begin
    $display ('Request canceled');
    state1 = 'Purchase_exp;
  end
else
  begin
    state1 <= 'Purchase_exp;
  end
end

default:
  begin
    state1 <= 'Category_sel;
  end
endcase
end
begin

// $display ("Error from Pre−designed shake \
'");

end

endmodule
I.3 Payment and Reorder module

\begin{verbatim}
//timescale 1ns / 1ps

//Author: Balaji Salunkhe.
//Module Name: pay_reorder
//Project Name: Muscle Dispenser
//Adviser: Mark Indovina

//define Card_pay 3'b010
//define Mob_pay 3'b011
//define Deliver 3'b100
//define Reorder 3'b101
//define Pay_option 3'b001

module pay_reorder(
    srst, wrst,
    clk,
)
ready1, ready2, cardpay, mobpay, payment_check, same, same_prod, different, different_prod, not_required, exit, product_del );

input clk;

input srst, wrst;

input ready1, ready2, cardpay, mobpay, product_del;
input payment_check;

input
  same,
  different,
  not_required;

output reg
  same_prod,
  different_prod,
  exit;

reg
  [2:0] state;

//reg
//  done;
//  //product_del;

always @(posedge clk)
  begin
    if (srst == 1 || wrst == 1)
      begin
        same_prod = 1'b0;
different_prod = 1'b0;
exit = 1'b0;
state = 3'b000;
end

else if (ready1 == 1'b1 || ready2 == 1'b1)
  begin
    case (state)

      'Pay_option:
        begin
          if (cardpay == 1)
            begin
              state <= 'Card_pay;
            end
          else if (mobpay == 1)
            begin
              state <= 'Mob_pay;
            end
          else
            begin
            end
        end

      'Card_pay:

begin
    $display('Card payment selected')
    if (payment_check)
        begin
            $display('Thank you for payment')
            //done = 1;
            state <= 'Deliver;
        end
    else
        begin
            state <= 'Card_pay;
        end
end

'Mob_pay:
begin
    $display('Mobile payment selected')
    if (payment_check)
        begin
            $display('Thank you for payment')
            //done = 1;
            state <= 'Deliver;
end
else
begin
    state <= 'Mob_pay;
end

'Deliver:
begin
    $display ("\n Product is ready to deliver \n")
    //product_del = 1;
    state <= 'Reorder;
end

'Reorder:
begin
    $display ("*************** Fresh shake is on the way, please wait")
;
    if (product_del == 1)
        begin
            if (same == 1)
                begin
$display ('Want to order one more same shake, cause I liked it');
same_prod = 1;
different_prod = 0;
exit = 0;
end
else if (different == 1)
begin
   $display ('Want to try something new');
different_prod = 1;
same_prod = 0;
exit = 0;
end
else if (not_required == 1)
begin
   $display ('I am full, Thank you for service');
   exit = 1;
same_prod = 0;
different_prod = 0;
end
else
begin
end
end
else
begin

state <= 'Reorder;
end
end

default:
begin

state <=

'Pay_option
;
end
endcase
end

else
begin
end
end
endmodule
I.4 Watchdog module

`timescale 1ns / 1ps

//

////////////////////////////////////////////////////////////////////////////

// Author: Balaji Salunkhe.
// Module Name: watch_dog
// Project Name: Muscle Dispenser
// Adviser: Mark Indovina

////////////////////////////////////////////////////////////////////////////

`define Counter_20 2'b01
`define Counter_30 2'b10
`define Counter_40 2'b11

module watch_dog(
  clk ,
  srst ,
  // sel ,
  // cost_check ,
add_more,
same_prod,
different_prod,
exit,
option_sel1,
option_sel2,
wait_prod1,
wait_prod2,
cancel1,
cancel2,
wrst
);

input clk;
input srst;
//input ready;
input same_prod, different_prod;
input exit;
input option_sel1, option_sel2;
//input [2:0] sel;
input add_more;
//input cost_check;
input wait_prod1, wait_prod2;
input cancel1, cancel2;
output reg wrst;
reg [5:0] count;

reg [1:0] state;
//wire c_sel;


always @(posedge clk)
  begin
    if (srst)
      begin
        count = 5'b00000;
        state = 2'b00;
      end
    else if (option_sel1 == 1'b1 || option_sel2 == 1'b1)
      begin
        case (state)
          'Counter_20:
            begin
              if (wait_pro1 == 1'b1 ||
                  wait_pro2 == 1'b1 ||
                  cancel1 == 1'b1 ||
                  cancel2 == 1'b1 ||
add_more == 1'b1)
begin
  if (count == 20)
  begin
    $display ("Are you still there?");
    count = 5'b00000;
    state <= 'Counter_20;
  end
else
begin
  count = count + 1'b1;
  $display ("#####
    SELECTION PROCESS
    TIMER = %d",
    count);
state <= 'Counter_20
;
end
end
else
begin
  count = 5'b00000;
  state <= 'Counter_30;
end
end

'Counter_30:
begin
  if (count == 30)
  begin
    if (same_prod == 1'b0 &&
        different_prod == 1'b0
        && exit == 1'b0)
      begin
        wrst = 1'b1;
        $display ('System
          is resetting'
      );
      count = 5'b00000;
I.4 Watchdog module

```verilog
state <= 2'b00;
end
else
begin
state <= 'Counter_40;
end
end
else if (wait_prod1 == 1'b1 || wait_prod2 == 1'b1 || cancel1 == 1'b1 || cancel2 == 1'b1 || add_more == 1'b1)
begin
    count = 5'b00000;
    state <= 'Counter_20;
end
else
begin
    count = count + 1'b1;
    $display("#### SYSTEM WATCH DOG TIMER = %d", count);
    state <= 'Counter_30;
end
```
end

‘Counter_40:
begin
if (count == 40)
begin
wrst = 1'b1;
$display ('System is resetting');
count = 5'b00000;
state = 2'b00;
end
else
begin
count = count + 1'b1;
$display ('#### FORCING FOR RESET TIMER = %d',
count);
state <= ‘Counter_40;
end
end

default:
begin
state <= ‘Counter_20;
end

dncase
end

elese
begin
$display ("\t \t MUSCLE DISPENSER");
end
end
endmodule
module Muscle_dispenser(
    clk,
    srst,
    //wrst,
    option_sel1,
    option_sel2,
    avl1,
    avl2,
    avl3,
    avl4,
)

'} timescale 1ns / 1ps

//
/////////////////////////////////////////////////////////////////////////////
// Author: Balaji Salunkhe.
// Module Name: Muscle_dispenser
// Project Name: Muscle Dispenser
// Adviser: Mark Indovina
//
/////////////////////////////////////////////////////////////////////////////

I.5 Muscle Dispenser module
I.5 Muscle Dispenser module

avl5,
avl6,
sel,
cost_check,
add_more,
payment_check,
cardpay,
mobpay,
product_del,
same,
different,
not_required,
cancel1,
cancel2,
ack1,
ack2
);

//Inputs
input clk,       //System clock
srst;          //System reset
wire wrst;     //watchdog reset

input option_sel1,
     option_sel2;
input [7:0] avl1, avl2, avl3, avl4, avl5, avl6;

input [2:0] sel;

input cost_check;
input add_more;
input cardpay, mobpay;
input payment_check;
input product_del;
input cancel1, cancel2;
input same,
    different,
    not_required;

// Outputs
output [5:0] ack1, ack2;

wire same_prod,
    different_prod,
    exit;

// wires
wire ready1, ready2;
wire wait_prod1, wait_prod2;

cust_shake cust_shake_i(
    .srst(srst),
    .wrst(wrst),
    .clk(clk),
    .option_sel1(option_sel1),
    .wait_prod1(wait_prod1),
    .sel(sel),
    .avl1(avl1),
    .avl2(avl2),
    .avl3(avl3),
    .avl4(avl4),
    .avl5(avl5),
    .avl6(avl6),
    .ack(ack1),
    .cost_check(cost_check),
    .ready1(ready1),
    .cancel1(cancel1),
    .add_more(add_more)
);

pre_design pre_design_i(
    .srst(srst),
    .wrst(wrst),
I.5 Muscle Dispenser module

(clk(clk),
.option_sel2(option_sel2),
.wait_prod2(wait_prod2),
.sel(sel),
.avl1(avl1),
.avl2(avl2),
.avl3(avl3),
.avl4(avl4),
.avl5(avl5),
.avl6(avl6),
.ack(ack2),
.cost_check(cost_check),
.ready2(ready2),
.cancel2(cancel2)
);

pay_reorder pay_reorder_i(
  .srst(srst),
  .wrst(wrst),
  .clk(clk),
  .ready1(ready1),
  .ready2(ready2),
  .cardpay(cardpay),
  .mobpay(mobpay),
  .payment_check(payment_check),
I.5 Muscle Dispenser module

	.same(same),
	.same_prod(same_prod),
	.different(different),
	.different_prod(different_prod),
	.not_required(not_required),
	.exit(exit),
	.product_del(product_del)

);
endmodule
Figure II.1: Simulation Waveforms using the Xilinx Vivado Design Suite