Bilkent University

CS 223

Digital Design

Project Final Report

Author:
Selim Firat Yilmaz

Student Number:
21502736
Section 3

December 18, 2017

Contents

1 Project Description **2**

2 User Guide **2**
2.1 Game Modes 2
2.1.1 Play Mode 2
2.1.2 Training Mode 2
2.2 Code Mapping Table 3

3 Project Design **5**
3.1 High Level Design 5
3.1.1 Block Diagram 5
3.1.2 Block Diagram Description 5
3.2 Low Level Design 7
3.2.1 Finite State Machine of Game Logic 7
3.2.2 Random Number Generator Logic 7

4 Discussions **8**
4.1 Did you change your high-level design compared to progress report? . 8
4.2 How did you implement each module in your high level design? 8
4.3 How was the implementation stage? 8

1 Project Description

This game named **Game of Codes** is developed for educational purposes as a project for CS 223 course in Bilkent University.

2 User Guide

2.1 Game Modes

2.1.1 Play Mode

When game starts, the stepmotor moves according to a random sequence of movements. Then, it waits for player to press a button on 4x4 keypad. If player presses the correct button, score will be incremented by 1. If player presses the wrong button, score will be decremented by one. Score is shown on the most right part of the 7-segment and is limited to the range of 0-9. If player

Player can turn on the most right switch on Basys3 (V17) to go back to the training mode.

2.1.2 Training Mode

In this mode, when a key in the 4x4 keypad is pressed, the the stepmotor moves according to the corresponding sequence of moves in the mapping table. As a remark, if player goes in to training mode, the score will reset to 0.

Player can turn off the most right switch on Basys3 (V17) to go back to the play mode.

2.2 Code Mapping Table

Key	First Move	Second Move
1	Short Left	Short Left
2	Long Right	Long Right
3	Long Left	Long Right
4	Long Left	Long Left
5	Short Right	Long Right
6	Short Left	Long Right
7	Short Left	Long Left
8	Long Right	Short Right
9	Long Left	Short Right
0	Short Right	Short Right
A	Long Right	Long Left
B	Short Right	Long Left
C	Long Right	Short Left
D	Short Right	Short Left
*	Long Left	Short Left
#	Short Left	Short Right

Moves	Key
SL - SL	1
LR - LR	2
LL -LR	3
LL - LL	4
SR - LR	5
SL - LR	6
SL -LL	7
LR - SR	8
LL - SR	9
SR - SR	0
LR - LL	A
SR - LL	B
LR - SL	C
SR - SL	D
LL - SL	*
SL - SR	#

Same table with abbreviations of LR (Long Right), LL (Long Left), SL (Short Left), SL (Short Right) shown. Moves are shown in the "Moves" column with the format "First move - Second move"

3 Project Design

3.1 High Level Design

3.1.1 Block Diagram

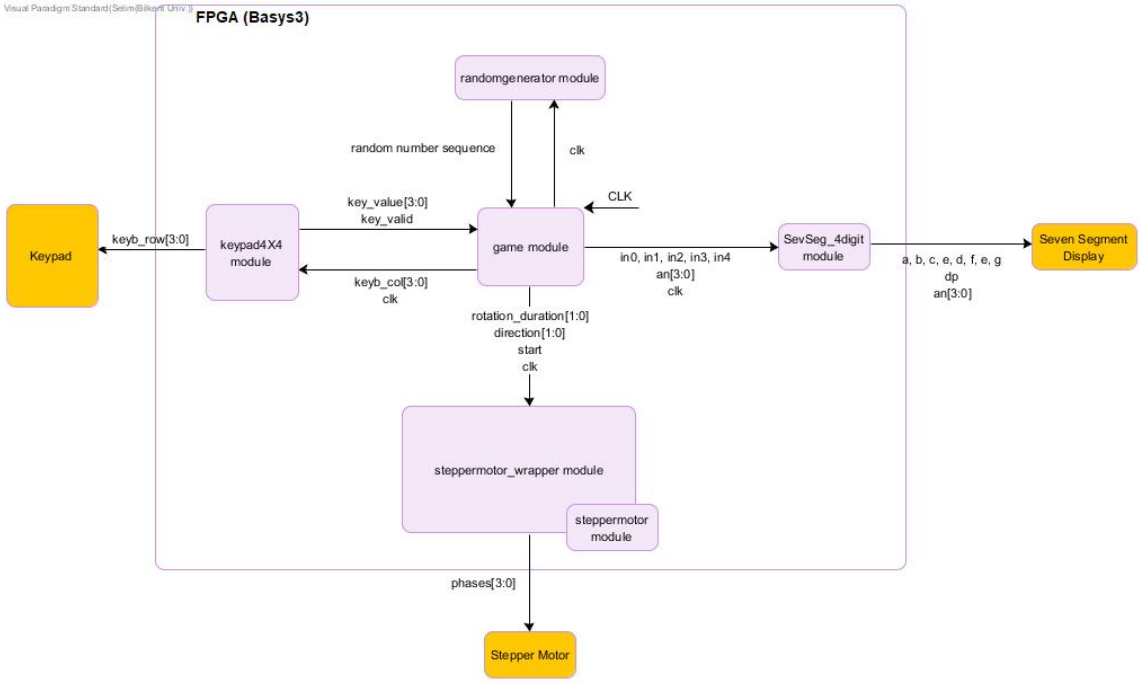

Figure 1: Block Diagram

3.1.2 Block Diagram Description

Game module is the top module that is is control of the gameplay synchronously. Takes the random number and maps it to pre-defined movement sequence, and gives it to the steppermotor_wrapper module in order to move the stepper motor. When user clicks the button in Keypad, evaluates the key with the value of the movement sequence. If they match, the score increases, and if they do not match, the score decreases. And the new score will be displayed on the Seven Segments module through SevSeg_4digit module. Player can also reset the score via the middle push button on Basys3.

Randomgenerator module is the module that generates a pseudo-random se-

quence. It will be implemented using a linear-feedback shift register (LFSR).

Steppermotor_wrapper module makes the movement of the Stepper Motor through the **steppermotor module** using the randomly generated number's mapped movement.

SevSeg_4digit module makes the Seven Segment Display show the score of the player which is given by the game module.

Keypad4X4 module gets the player's input as the button player clicks, and gives it to the game module.

This part is taken from the progress report for completeness since nothing is changed in terms of high level design.

3.2 Low Level Design

3.2.1 Finite State Machine of Game Logic

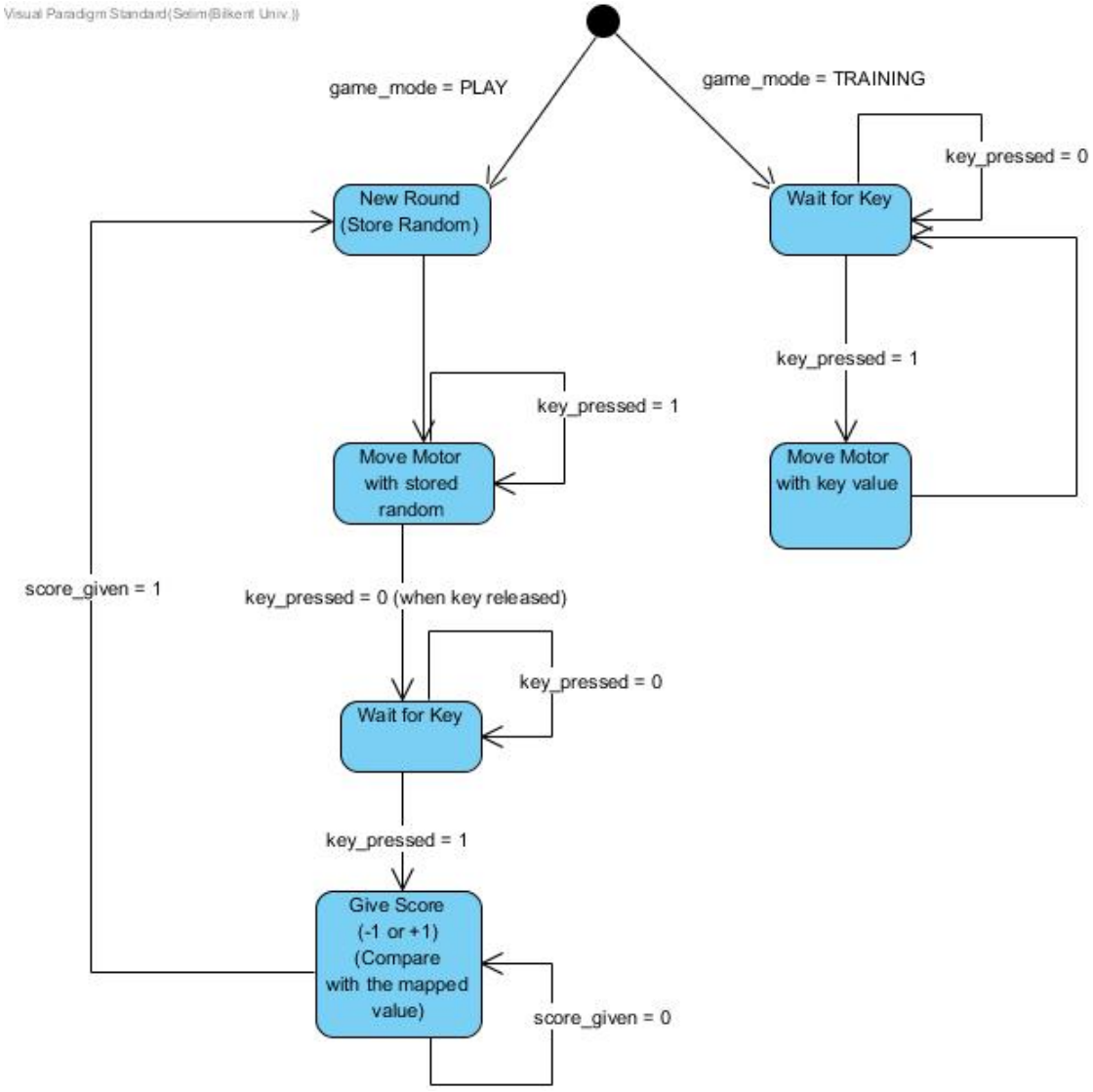

Figure 2: FSM of the game logic

3.2.2 Random Number Generator Logic

Let o be the output of the LFSR(Linear Feedback Shift Register). Then my implementation takes $XNOR$ of o's least significant 3 bits and determines the value of

new least significant as is. Namely gives it back to the LFSR as an input.

4 Discussions

4.1 Did you change your high-level design compared to progress report?

No, I did not change my high level design compared to progress report. I was able to implement it directly so I believe it fits the problem.

4.2 How did you implement each module in your high level design?

I used ready modules for 7 segment display, stepper motor, and 4x4 keypad. To implement game module, I first tried to code it directly. However, it was too complex but gave me an insight about how implementation should be. Then I divided the problematic parts as an FSM and named them as in the FSM above. After doing this the implementation became trivial. Details on game Logic and the random number generator Logic can be found above in the low level design part.

4.3 How was the implementation stage?

Since we have many assignments, it was hard to find time for this project. However, I managed it. Another problem was BETI board since not all of them are working perfectly. Also, when I connect FPGAs to one computer in the lab it do not work but in others, it works well. I believe left 3 columns of the keypad was not able to get enough power.