ECSE 323

Digital Systems Design

Lab 5: System Integration for the Mastermind Game System

Lab Report

Group – 04

Sharhad Bashar: 260519664

Richard Wu: 260509483

Dec 7th, 2015

Table of Contents

Introduction

Mastermind is a code breaking game played between two players. There are four pegs, and each peg has six different colors. Player one sets a hidden pattern using the pegs and the second player has to guess the exact pattern of the hidden pegs. After each round, player one gives feed back to the second player based on his guess. This feedback consists of the total number of exact guess (color and position) and color match (the number of color matcher, bit not position). Throughout five labs, we are to develop all the building blocks for the mastermind game and integrate them into a complete, user-friendly system using the Altera Cyclone V FPGA board

Description of System Features

This lab utilizes all system features from previous labs combined with newly implemented features including the 7 segment decoder, the possibility table, the datapath, the controller, and score generator. The system is split into 2 modes:

User Guessing Mode

The system generates the original pattern (4 random colors on each peg). The user inputs guesses and receives a score for the guess. Both the user input and the score will be generated on the 7-segment displays. As shown in *Figure 1*, if the user chooses the user guessing mode, the system will generate a random pattern, then the user will continue to input guesses and receive scores from the system until the user receives a score of (4,0).

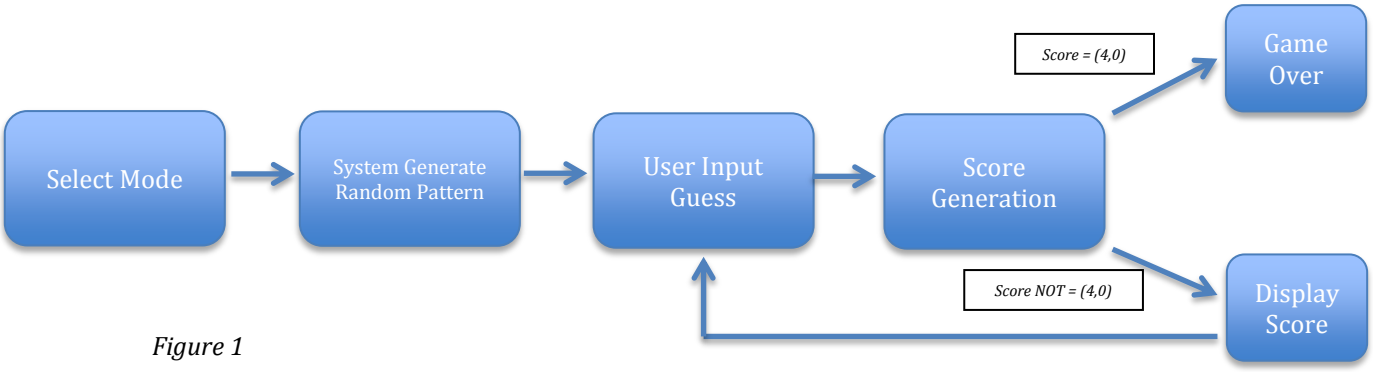


Figure 1

System Guessing Mode

This is the case where the system implements a solution algorithm. At each level in the algorithm, the system displays current guess and allows user to enter score, this judgment is made based on the answer that the user chose. The current system will guess and the user assigned score are also displayed on the 7-segment display. This is shown in Figure 2

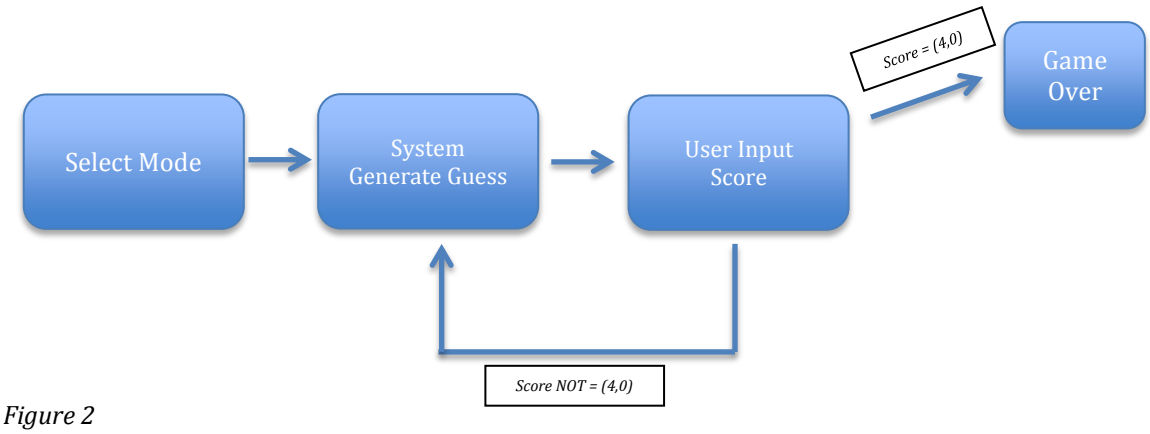


Figure 2

Description of Component Features:

7-Segment Decoder
The 7-segment decoder is used to output values to the 7-segments. It maps values in the registers to be displayed on the 7-segment. The values displayed consist of game score and system/user guesses.

Possibility Table
The possibility table goes through every single possibility that can be an output as a guess. Each 7-segment can display 6 different values, and there are 4 different segments, therefore there is a total of 1296 total possibilities for guesses to choose from.

Datapath
The datapath module is used to process and manipulate data passed in from the controller and send back status signals. The datapath in this lab contains various registers and multiplexers, as well as the possibility table from lab 3. The registers to store user or system data while multiplexers to select between the two in different scenarios. Each of the units in the datapath run off the same clock signal.

Controller
The control module generates control signals that modify the processing of the datapath modules. The controller is a finite state model and takes in input from the user as well as supply inputs to the datapath. Figure 1 displays the State Diagram that the controller operates on.

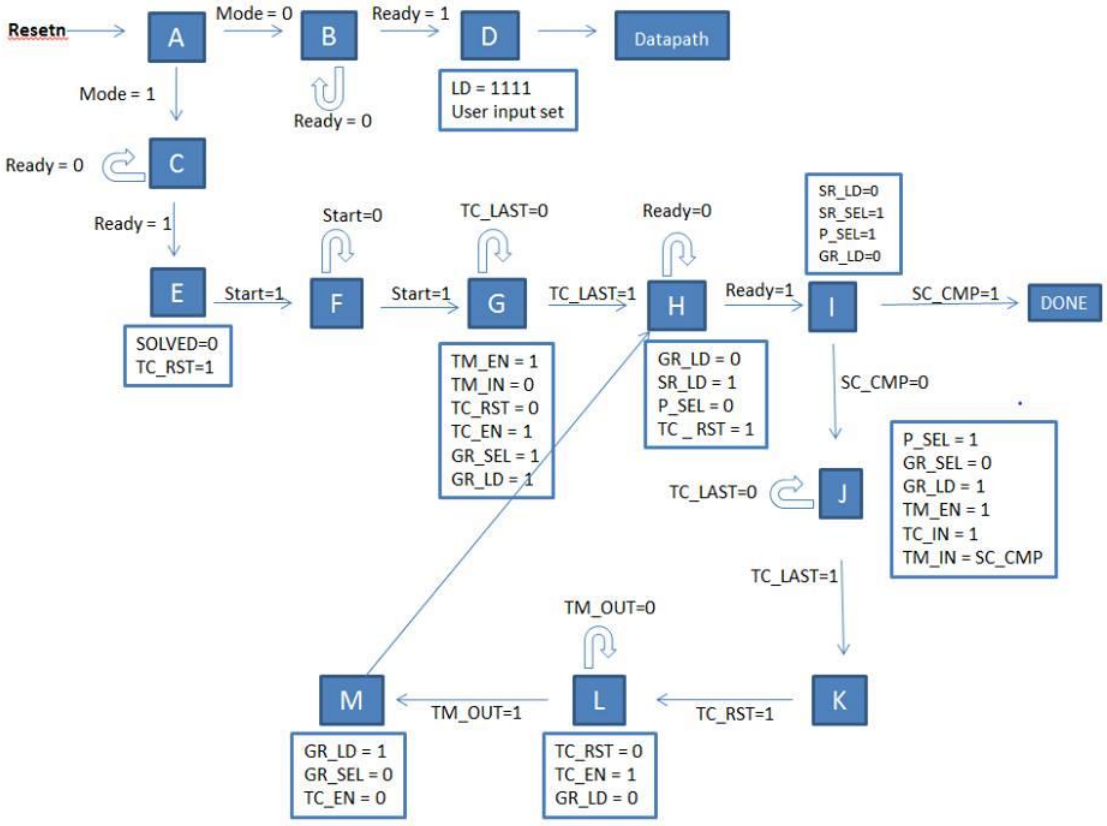


Figure 3

Score Generator

The score generator is for the user mode of the game. It is calculated after the user enters their guess. There are 14 possible scores, and the scores come in two parts. First every bit of the users entry is compared to the hidden pattern. The first part of the score tells the user how may exact matches (both color and position vise) they have. And the second part is used to give feedback based on just how many color matches the user got. Using this feedback, if the user did not get all four perfectly matched, they would then input another guess.

The system mode also makes use of a score. After the system makes a guess, the player would enter a score based on the guess, and the algorithm would go through the possibility table and generate a new and improved guess. On the Cyclone 5 board, the left most two 7-segment displays were used for displaying the score, with the exact matches on the left, and the color matches on the right, as shown on the right.

Block Diagram of System

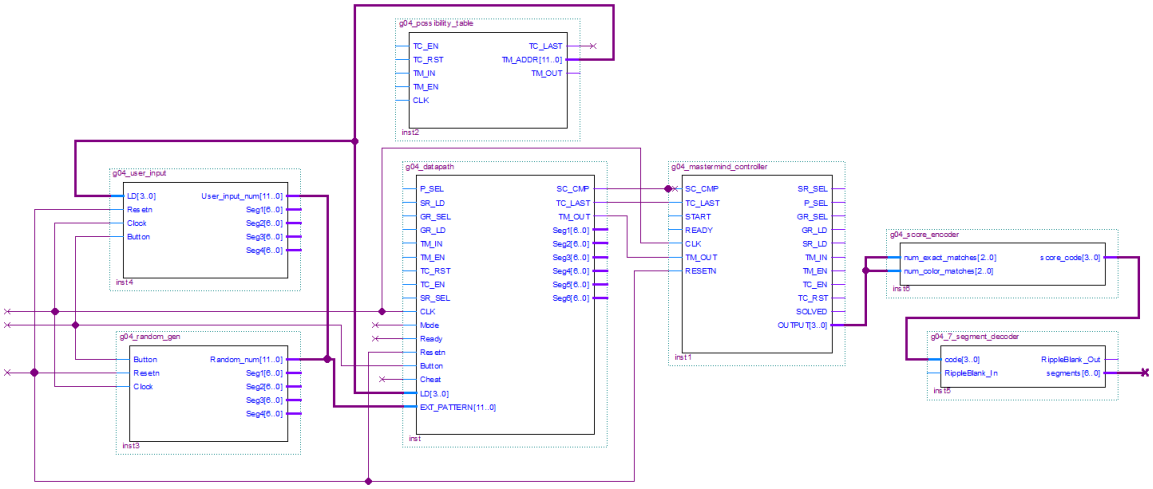


Figure 4

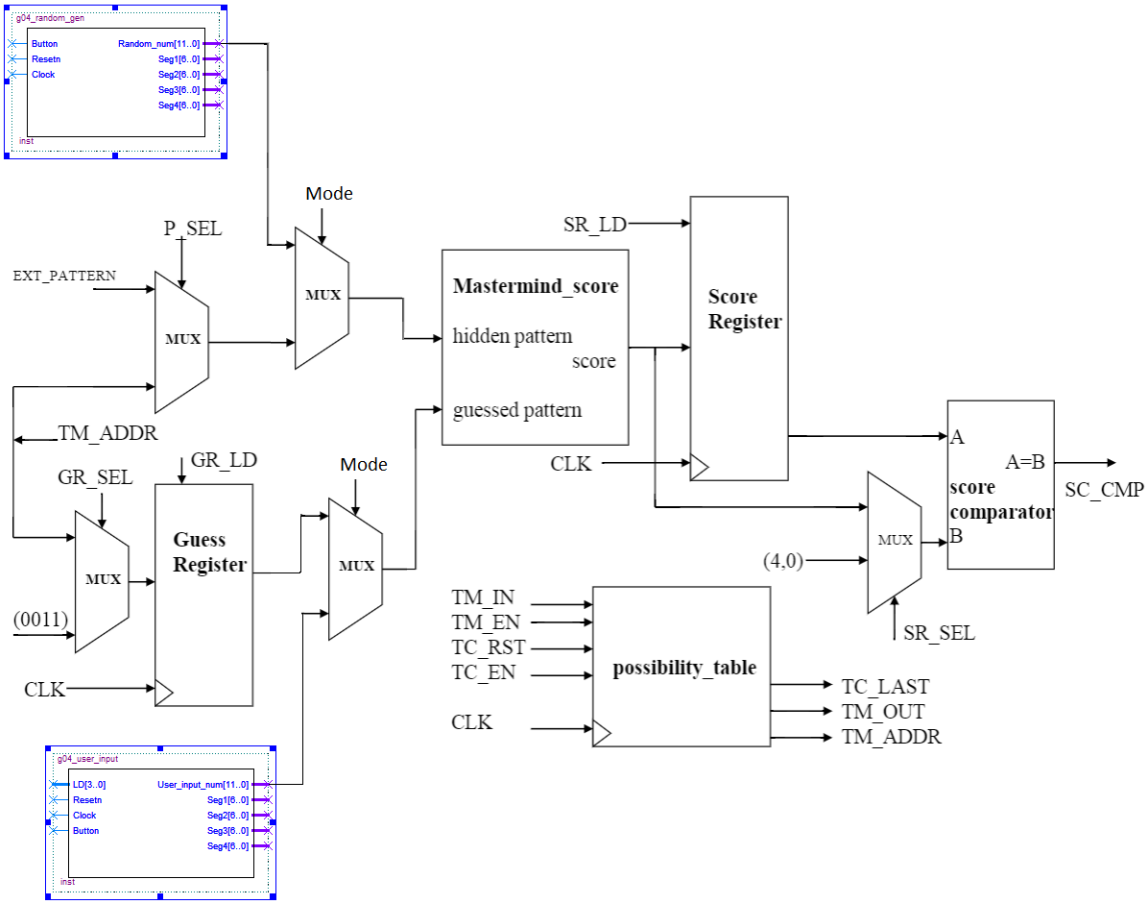


Figure 5

Description of System Functionality

The block diagram describes the functionality of the entire mastermind game. The design is implemented using Data path/Controller architecture. The Data path is made up of various data processing and manipulation modules. The controller is a Finite State Machine that generates control signal that modifies the behaviour of the data path element using a given algorithm.

The data path is connected to user input and random generator through two multiplexers, as we can see in the updated data path pin-out diagram. The data paths output SC_CMP, is a comparison to check if the user entry and the random generated guess and similar. The controller uses that data, and outputs many different things, some of which are inputs back to the data path, while others are inputted in a score encoder, where the score is calculated. The final block is the 7-segment decoder, where the output from the score encoder is decoded once again so that it can be displayed in using the six 7-segment displays available to us.

Description of User Interface

The user interface is split into 2 modes:

User Guessing Mode
In user guessing mode, the user needs to input 4 color guesses and their positions onto the board to be displayed by the 7-segments. As shown in *Figure 6*, the user needs to first choose the user guessing mode, followed by flicking on the SW0 switch on the board, now the user inputs the first color in the right-most 7-segment. The user can do this by tabbing on KEY1 switch on the board. The following sequences are used for the tabbing:

000 → b
001 → n
010 → r
011 → j
100 → l
101 →o

Each tab represents input represents a color that the user can choose from.

Each represents a color. Once the color is set the user will flick off the SW0 switch to signal to the system that the first color is chosen. Then the user will choose the next right most color using the same sequence by flicking on SW1 switch on the board.

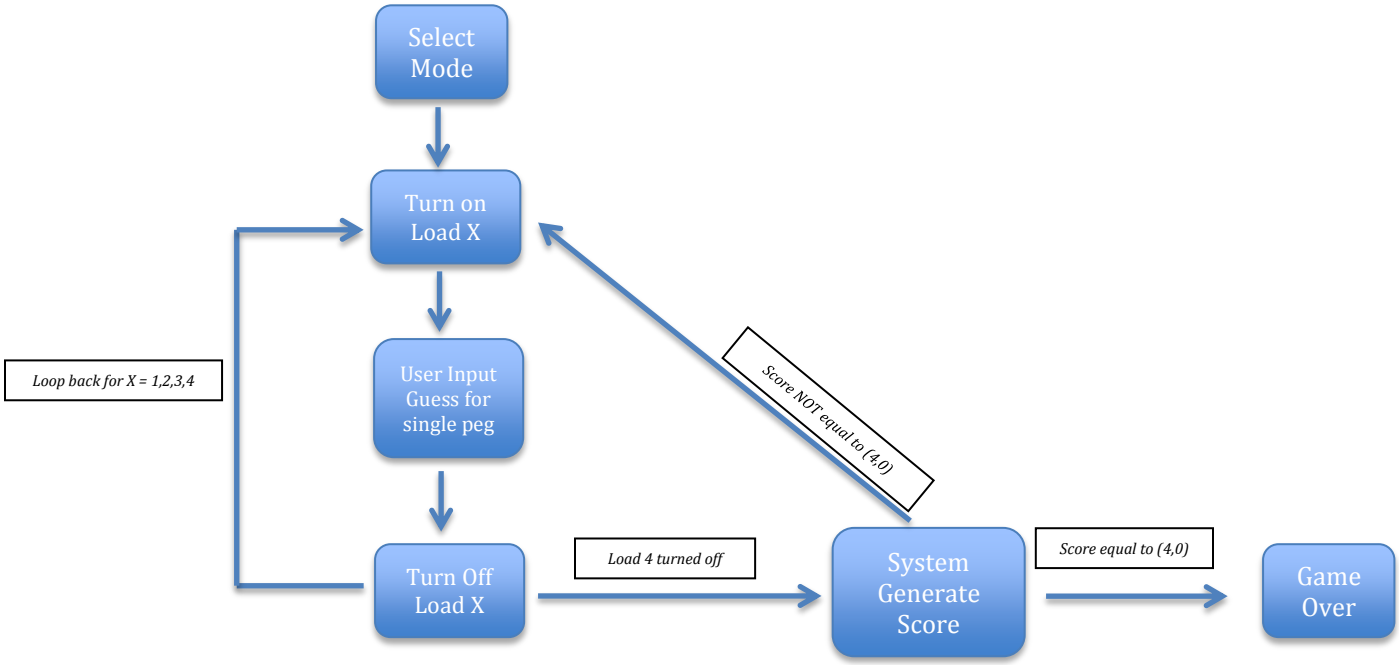


Figure 6

Figure 7 displays the actual board with the each button assigned as follows:

A →Mode
B → Ready
C → Toggle user & random cheat
D → 4 buttons for 4 loads
E → Reset
F → Button
G → Clock

As mentioned in Figure 3, the user would set the mode using button A, turn on load 1,2,3,4 for the four guesses one at a time using 1 of the 4 buttons in C, and click on D incrementally to set the value of guess per peg.

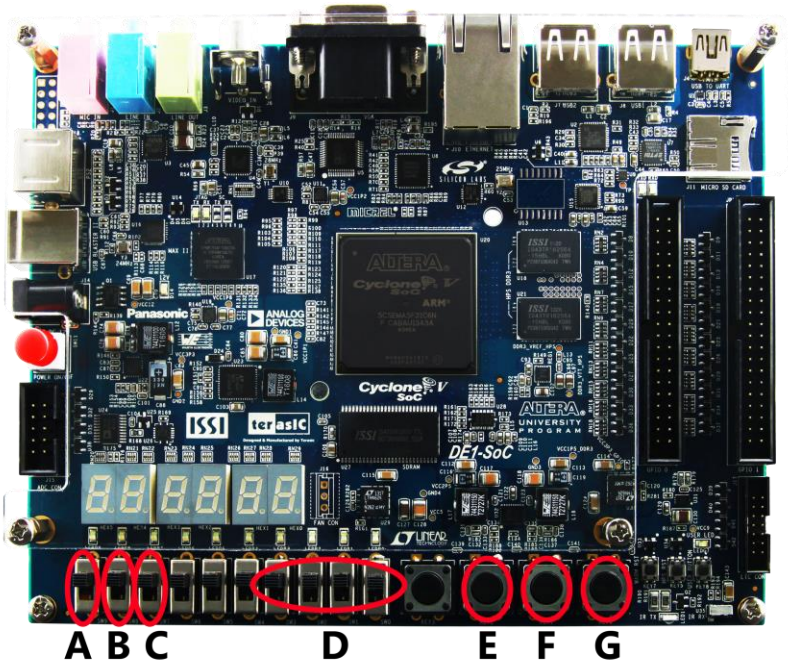


Figure 7

System Guessing Mode

In the system guessing mode, the system generates the random 4 colors for the pegs and displays it. The system will output it's guesses and the user needs to input the score for each guess. Since our team has access to 6 7-segments instead of the 4, we will enable to user to enter the score onto the 2 leftmost 7-segments, and display the score on the 4 rightmost 7-segments. The user will input the score as shown in *Figure 8*.

In this mode, all button and functionalities are similar to that displayed in Figure 4, with 1 key difference that the user is inputting 2 values into the 7-segments rather than 2, so there are 2 load switches instead of 4 shown in Figure 4.

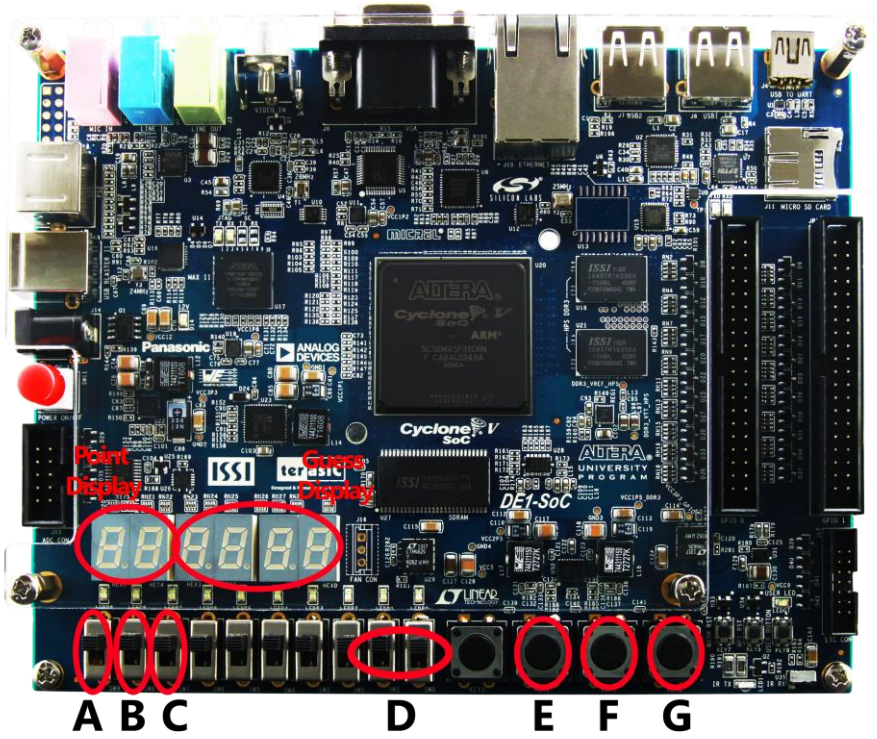


Figure 8

Discussion of Complete System Testing

The system testing of the hardware was conducted to evaluate the system's compliance with the requirements of the Mastermind game. The complete system testing targets the design of hardware and a wide range of other tests including the user interface and user experience. The complete system testing was split into a few categories as follows:

User Interface Testing

User interface testing is used in user-centered interaction design to evaluate a product by testing it on users. The tests focus on the users' ability to play the game, including choosing modes, inputting guesses, or inputting scores in an intuitive way. Furthermore, this test takes into consideration delay of outputs, as feedback from a system directly affects the user experience with the game.

The ModelSim diagram below shows how the system accepts and stores input from users. The four LD switches are first set to high, one at a time. Then the user uses the push button to enter a number from 0 to 5. Once the user has selected the number, they set the LD switch to low, and the data entered is stored in a variable. The diagram below is the time analysis for an input of "0112" The stored value User input will be red, until and LD switch goes to high and then low again. This is shown in Figure 9.

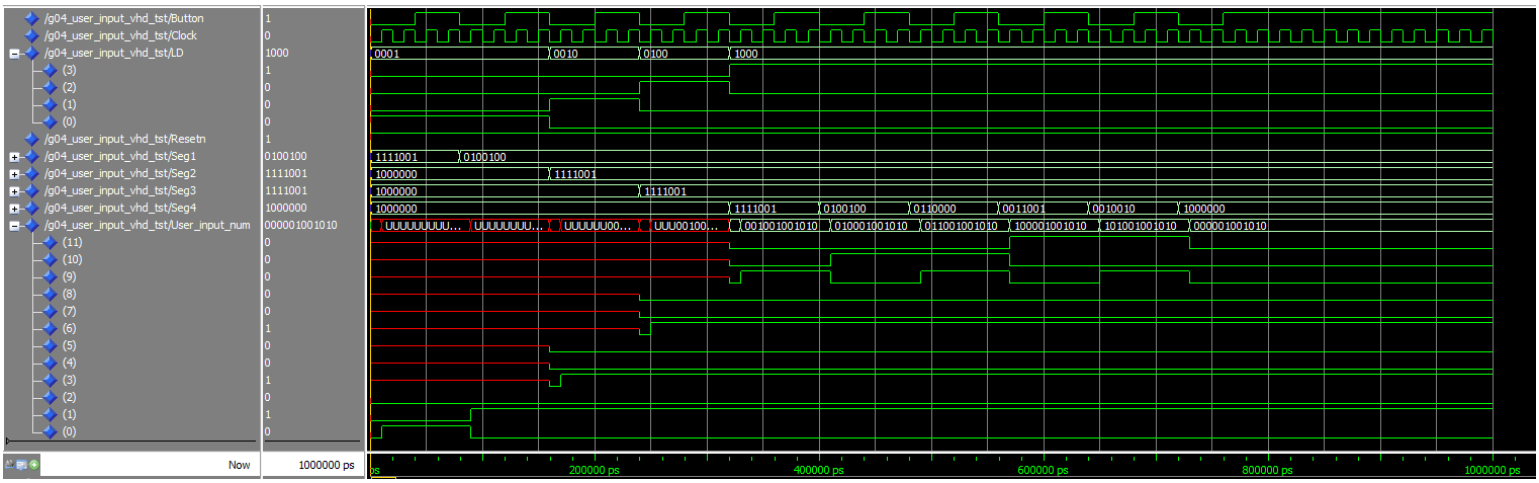


Figure 9

Sanity Testing
Sanity testing is a basic set of tests to confirm that the results are within expectations, for instance if a color that wasn't within the choices of colors appear, this would break the sanity test. The sanity test was conducted by re-running the game iteratively to ensure no obviously false results occur.

Reset Testing
This test is essentially testing for the ability of the game to be reset at any point. This test also ties in with recovery testing, which determines how well the application is able to recover from crashes or other hardware failures.

Unit Testing
Unit testing was very important for this particular lab. To ensure that each component functions accordingly on its own, including: 7-segment decoder, controller, path, possibility table and score generator.

Integration Testing
Integration testing was used when each unit was combined to meet system specifications. This test was only conducted when unit tests were complete.

Input / Output Testing

Input / output testing is a of a method of testing in which VHDL test cases were written and results were gathered from running the code on the FPGA boards. The data gathered were critical in debugging and completing test cases.

Mastermind Testing

<u>System Guessing Test</u>

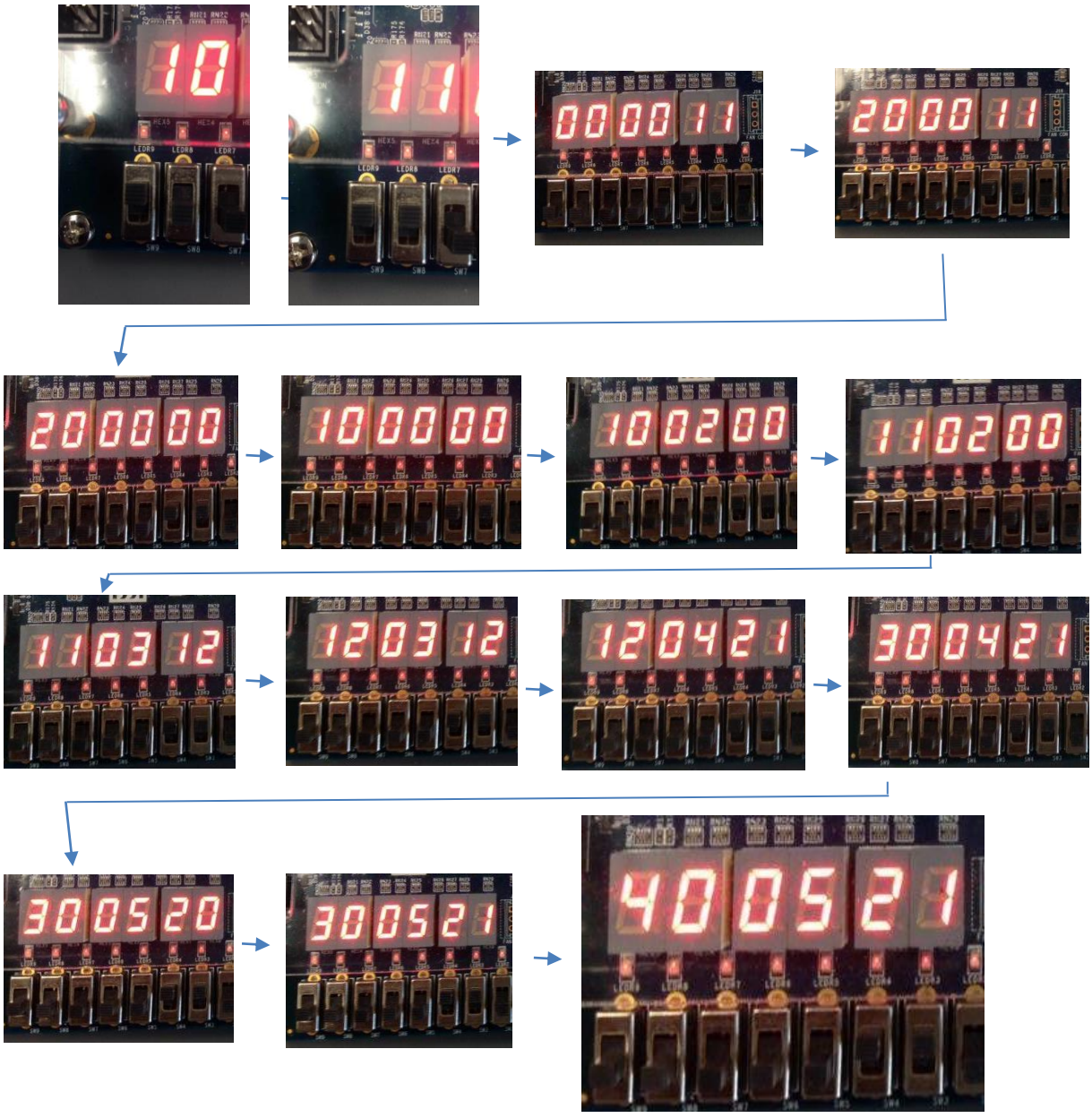


Figure 10

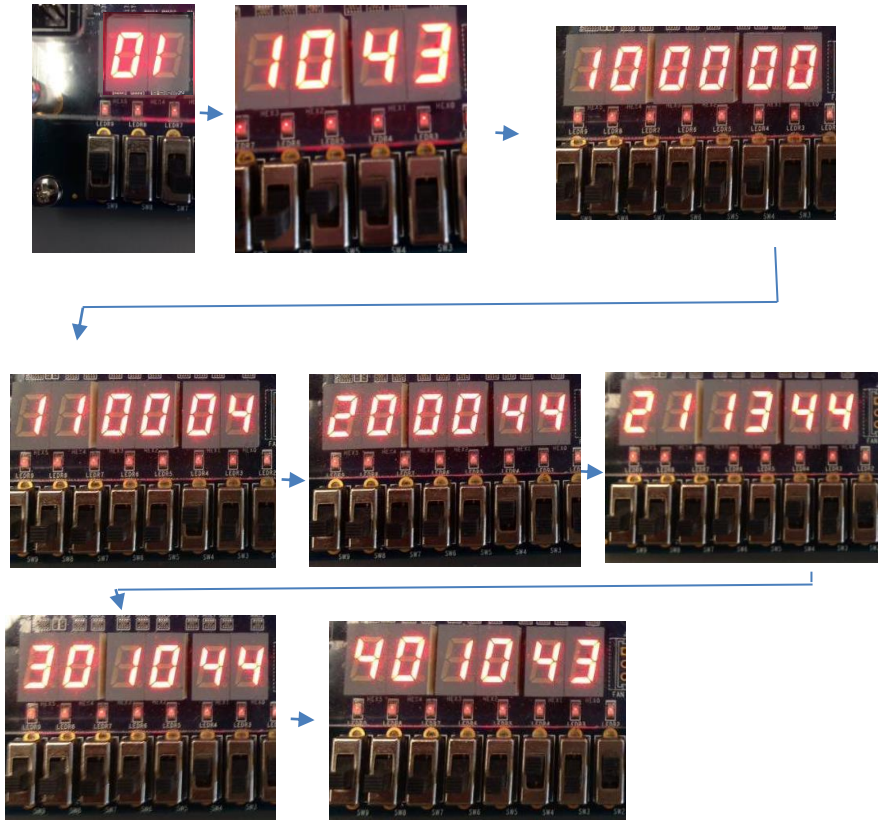
In the System guessing shown in Figure 10, we made the score “0521”. After we selected the mode and gave the ready signal, the system starts with a default “0011”. There are two exact matches, thus a score of (2,0) is inputted. Similarly, the system generates guesses based n the input, and gets the final answer, after guesses. The guesses and corresponding input are shown in the table below:

System Guess	Input
0011	2,0
0000	1,0
0200	1,1
0312	1,1
0412	3,0
0520	3,0
0521	4,0

User Guessing Test

For User guessing, the mode is set to 0, and after the ready signal is set to high, the system generates a random pattern using the possibility table. There is a cheat button, which allows us to check the score, which was randomly generated as “1043”. We start with an initial guess of “0000”. The system, correctly shows that there is one exact matches. We then pick different values for different locations to figure out where the 0 is located. In this process, we also manage to find some of the other hidden bits. Using only 6 tries, we can guess the patter that the system generated, the pictures below with a step by step walk through of how the user mode was played.

Figure 11



FPGA Resource Utilization

Flow Summary	
Flow Status	Successful - Mon Dec 07 20:37:38 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	g04_lab5
Top-level Entity Name	g04_user_input
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	33 / 32,070 (< 1 %)
Total registers	16
Total pins	47 / 457 (10 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 12

Conclusion

Problems or Issues during Design

1. Debugging
    a. Mainly singling out the main bug as it could have been within several components in VHDL, or with the pin assignments
2. System Integration
    a. It was hard to determine which component had the bug when the code didn't compile
3. Pin assignment
4. Timing issues between components
5. FPGA Board
    a. Easily mistaken high and low for buttons on click

Possible Enhancements or Extensions

1. Increase compile time
2. Increase clock speed
3. More displays to cater to more outputs
4. More input methods
    a. For instance one of our input methods was clicking a button as it increments the 7-segment 1 at a time. This is a slow process. If the guess can be directly entered, the system would be more ideal