Digital Systems Project Report (ECE241)

Project Name: The Maze

Teaching Assistant: Abdelrahman Abbas

Team Members:	**Student Number:**
Ehsan Nasiri	995935065
Rafat Rashid	996096111

Date: December 3rd, 2008

1. Introduction

For the final project in the Digital Systems course, we designed a circuit that implemented a maze onto the display and have an object traverse the maze and complete it. The user that is playing the game utilizes four keys on the Altera DE2 Board to move this object and the goal is to guide it through the maze to a final point (a diamond).

In the course, we learned how to compute, store, and control signals and data in hardware. This motivated us to create a real project in hardware using digital logic concepts. We used QuartusII to implement our design on the FPGA and used Verilog to describe our circuit.

We had several goals in mind when we started the project. These milestones are listed in order of completion below:

- Draw a background picture (picture of a maze onto the screen)
- Draw an object (person) in the maze
- Move the object in the maze
- Detect the walls of the maze (the object should not overwrite the walls)
- A timer that measures how long it takes for the user to complete the maze.

In the sections that follow, we describe the circuit design and provide a report on the performance and success of the project.

2. The Design

This block diagram illustrates all the parts of our project:

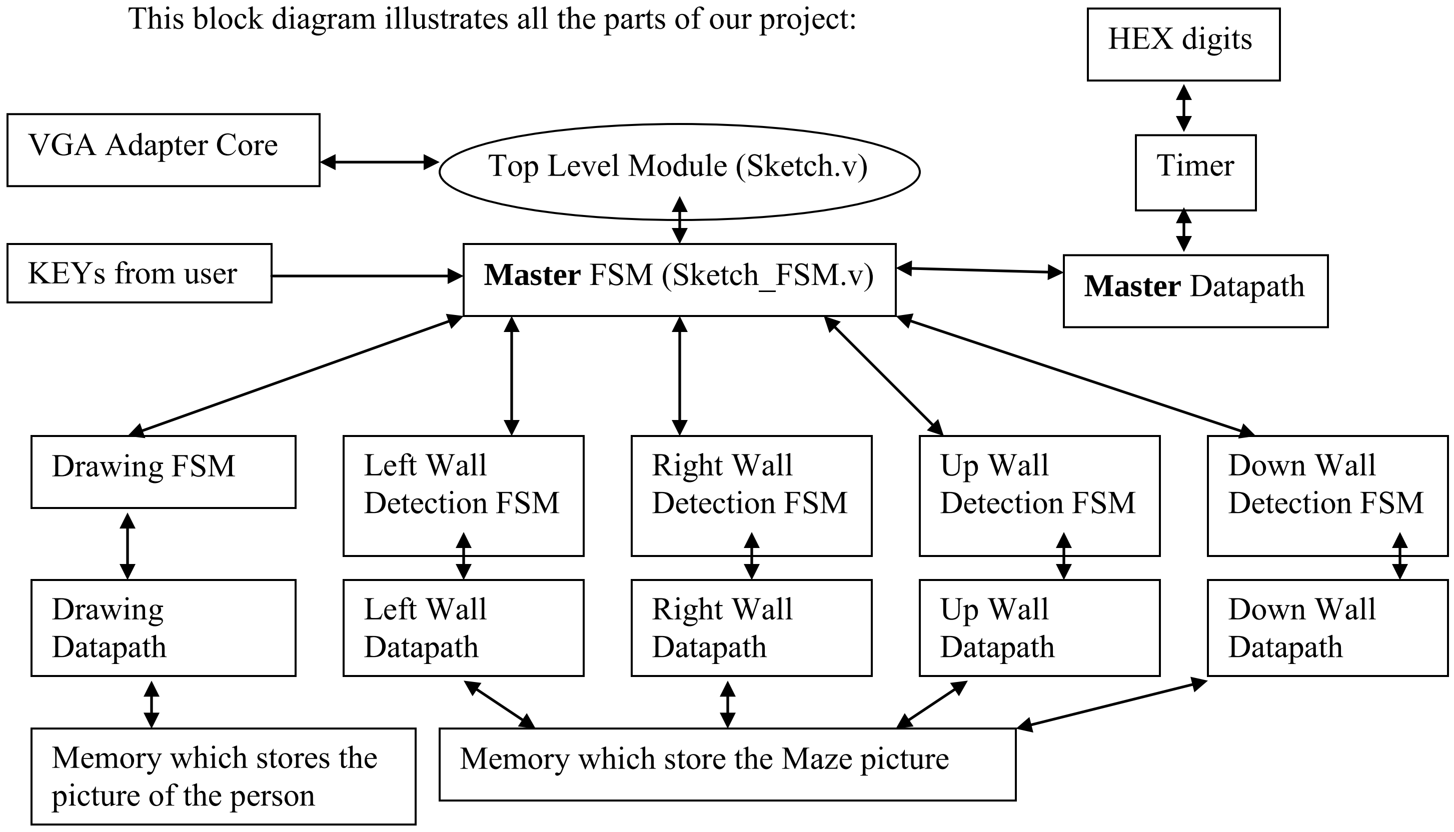


Figure 1. Design Block Diagram

Finite State Machines (FSM)

The master FSM gets the direction from the user using the keys. It then calls the corresponding direction's wall detection FSM. Wall detection FSM, together with its datapath, looks at the memory which stores the maze picture, and identifies if the movement is going to result in a collision with a wall. The way it detects the wall is by checking the 3-bit pixel color of the appropriate side, and if the color is black (3'b000), then it means it is a wall.

The wall detection FSM tells the Master FSM if we will hit a wall or not. If we have hit a wall, the Master FSM will go to a "Wait" state and wait for the user to press another key. If we have not hit a wall, Master FSM calls the Drawing FSM to start working. Drawing FSM gets the current coordinates of the picture(person) from the Master Datapath, and cleans it from that position (it draws a 16x16 pixel white box on that position). Then, using the Drawing Datapath, it increments or decrements the appropriate coordinate. Then, it reads the memory which contains the person, and draws it in the new position. This process repeats until the person reaches the diamond.

Memory Modules (lpm_ram_dq)

The maze that is seen on the screen by the user is a 160x120 pixels picture initialized to the VGA memory. The same picture is initialized to a chip memory that we can read for detecting the walls. The object (person) is a 16x16 pixel picture that is initialized to another memory in the chip. We have (Width x Height) number of pixels and a 3-bit color for each pixel, so, each memory is (Width x Height x 3) bits.

Master Datapath (datapath.v)

This datapath has two registers that remember the current X and Y position of the person on the screen. It also has two adder/subtractors that increment or decrement either X or Y each time the object moves.

Drawing Datapath (drawingDatapath.v)

This datapath contains three counters. One counter for counting the address in the memory, one for counting the X coordinate, and one for counting the Y coordinate. The enable signals for these counters come from the Drawing FSM. It also has a multiplexer that chooses to read the pixel colors either from the memory (for drawing) or a constant White color (for erasing).

Wall Detection Datapaths

Each of the wall detection datapaths contains a multiplier and an adder. It gets the X and Y coordinate from the Master Datapath, and find the corresponding memory bit that contains the color. (bit address# = Y * 160 + X). Each of the datapaths also has a counter that counts up to 16 (because the length of the picture is 16 pixels and we have to check all of the pixels on the sides).

So, each time we check the color, and if it is not black, we keep incrementing the address of the memory and add one to the counter. When the counter reaches 16, if no black pixels are read, the datapath tells the FSM that we have not hit a wall. Otherwise, it tells the FSM that we have hit a wall and the picture should not move.

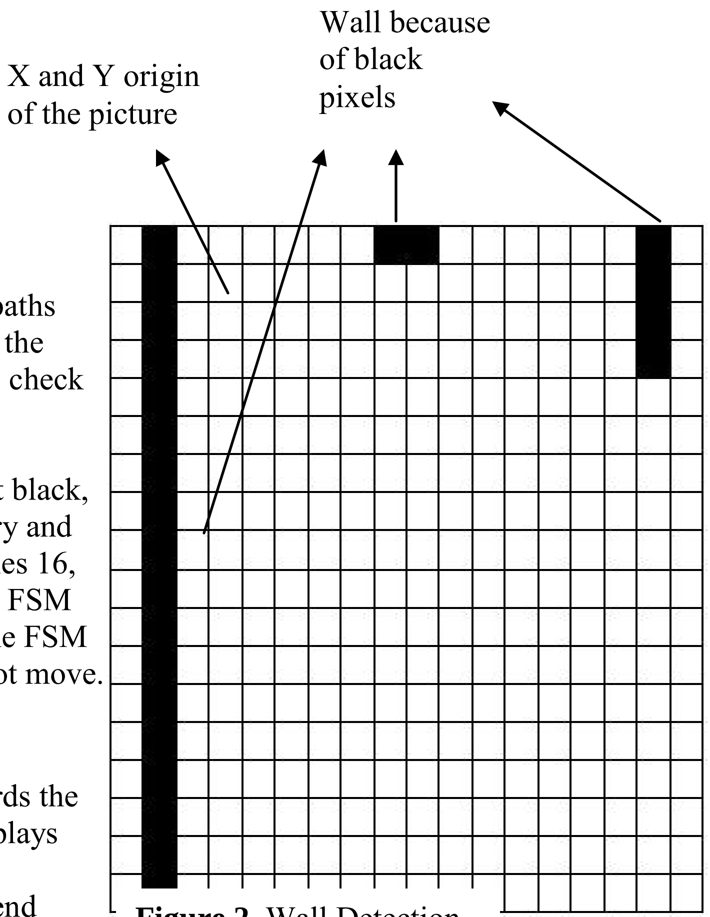


Figure 2. Wall Detection

Timer Module (with the hex_digits decoder)

This module implements the circuit which records the time elapsed since the start of the game and displays it on the HEX displays in real time using the hex_digits decoder. When the user reaches the end of the maze, the timer will stop but will continue to display the recorded time until the circuit is reset.

VGA Adapter

The VGA adapter core takes the X and Y coordinates from the Master Datapath, and a three-bit color from Drawing Datapath and plots a pixel with that color in that position on the screen. Drawing FSM instructs the core to draw a pixel (using the WriteEn signal).

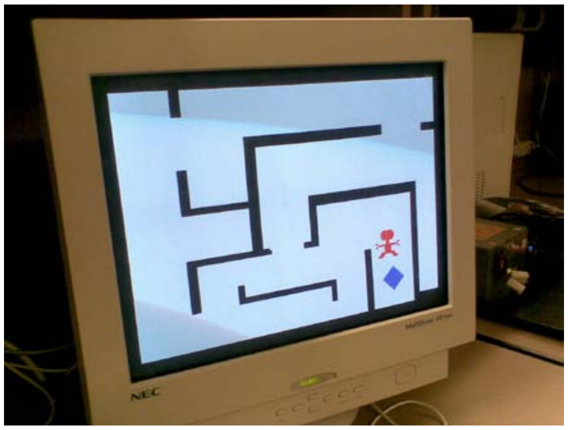
3. Report on Success

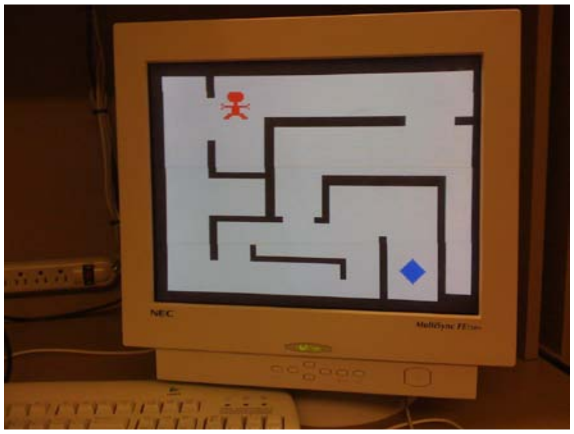
- **Did it work?**

Our project, "The Maze", performed as expected during the lab demo shown to the TA. The object moved across the screen and properly stopped moving upon reaching a wall. For instance, given there was a wall to the left of the object, it did not move left when the user pressed the "left" directional button. However, it was still capable of moving in the other three directions, provided there were no walls present in those directions. Moreover, the timer, which recorded the elapsed time since the beginning of the maze, stopped when the object reached the end of the maze and overwrote the blue diamond present there. In summary, we were able to achieve all of our primary goals, stated in the introduction of this report. The following pictures show the working project demo.

- **What would you do differently?**

We ran into many problems during this project. There were parts that were more complicated than we expected, and therefore, we had to put much more time to meet our milestones. Also, when we designed each part of the project separately, we assumed that it would not take much time to put the modules together. However, we ran into several additional problems such as the communication of the Finite State Machines with one another. So, although we managed to complete the project successfully, if we had to do the project again, we would assign much more time for debugging the circuit and putting the modules together.





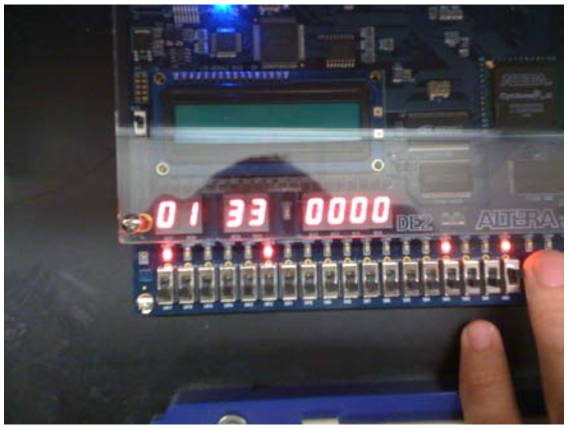


Figure 3. Project Demo