Accelerometer Controlled Labyrinth Game

EECS 3216

Luca Filippelli, Paras Kumar, Thevagan Nagarajah, Tharuveen Raveendran

April 8, 2024

Abstract

The goal of this project was to create a program to highlight the logical modelling techniques and DE10-lite-specific features highlighted in EECS 3216. A maze game was created such that an accelerometer-controlled square must navigate through a VGA-generated maze to reach a finish line. For added immersion, multiple levels were generated. The team initially developed a state machine along with several sub-states to model the program's requirements and switching logic. Afterwards, the team used several external modules along with custom Verilog programs to implement the model thus creating the game. The successful system implementation was tested after which it was discovered that more resources were needed to effectively implement the controller movement. The successful implementation was recorded as a demonstration video.

Introduction

In the realm of computer science and engineering, gameplay has always been an optimal method of demonstrating a collection of relevant skills through an engaging interface. Digital Modelling, like its conceptual peers, has integral concepts to the realm of game-making and gameplay. For instance, games typically have a control system and an interface. Both subsystems are best designed through modelling and implemented with some embedded system.

The purpose of this project was to highlight the VGA and accelerometer features of the DE-10 Lite board by constructing a maze-like game. The game features 3 distinct levels which require a user to navigate a character through the maze without colliding with any obstacles. Given the problem, the goals of this project were to create three distinct levels using VGA, generate a movement system through the use of the accelerometer, provide collision functionality for increased difficulty, create a level swapping mechanism and generate a method of initiating and resetting gameplay on a level.

To achieve the project goals, a state machine was generated and external modules were employed. 3 states are respectively used to represent initiation, level selection and gameplay. A further four states are used within the gameplay state to represent the start of a game, movement, character stillness and completion state. The state machine was then implemented in a System Verilog program to create a functional program. External modules were used to assist with VGA control and clocking. The VGA controller and PLL clock modules from the fifth lab in this course were used to supplement the group's work. Each submodule is connected using a top-level module to drive the inputs and outputs to the board.

Explanation

To initiate the design process, the system states were modelled and the levels were designed. The requirements for the system were initially modelled as a state machine. The machine included three main states, *Initial*, *Level_Selection* and *Gameplay* (figure 1). The *Level_Selection* state is responsible for generating one of three levels based on the two-bit binary input of *SW[1:0]*. The level is locked by moving *SW[9]* into the ON position. This lock ensures that a user cannot accidentally change levels during their gameplay session. Level locking transitions the state to *Gameplay*. Within

Gameplay, the substates *Game_init* and *Move* exist. Game_init, the playable square is still and in the initial position. In Move, the square is being translated across the map. *Game_init* will transition to move upon input from the accelerometer. Move will transition to *Game_init* upon a reset signal from *KEY[0]* or the detection of a collision as modelled by the boolean variable, *withinanyrectangle*. Within *Move*, an axis locking mechanism was implemented through *SW[5]* such that a high value will enable vertical movement and a low value will enable horizontal movement.

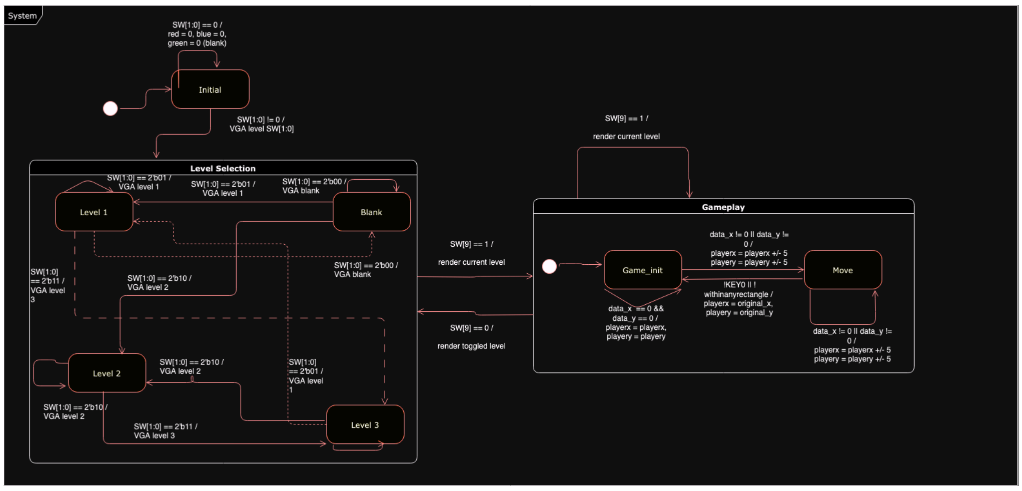
System
SW[1:0] == 0 / red = 0, blue = 0, green = 0 (blank)
Initial
SW[1:0] != 0 / VGA level SW[1:0]
Level Selection
SW[1:0] == 2'b01 / VGA level 1
SW[1:0] == 2'b01 / VGA level 1
SW[1:0] == 2'b00 / VGA blank
Level 1
Blank
SW[1:0] == 2'b11 / VGA level 3
SW[1:0] == 2'b10 / VGA level 2
SW[1:0] == 2'b00 / VGA blank
Level 2
SW[1:0] == 2'b10 / VGA level 2
SW[1:0] == 2'b01 / VGA level 1
SW[1:0] == 2'b10 / VGA level 2
SW[1:0] == 2'b11 / VGA level 3
Level 3
SW[9] == 1 / render current level
SW[9] == 0 / render toggled level
SW[9] == 1 / render current level
Gameplay
Game_init
Move
data_x != 0 || data_y != 0 / playerx = playerx +/- 5 playery = playery +/- 5
!KEY0 || ! withinanyrectangle / playerx = original_x, playery = original_y
data_x == 0 && data_y == 0 / playerx = playerx, playery = playery
data_x != 0 || data_y != 0 / playerx = playerx +/- 5 playery = playery +/- 5

Figure 1: A state diagram representation of the video game system

The level design consisted of creating three hand-drawn maps. Each map contains a series of boundaries with adjoining lengths, a starting position and a final position. These drawings were successfully presented in the project proposal. The levels were then generated through the use of combinational logic to drive the correct VGA values depending on the pixel area and desired level. To aid in visualizing character movement, the VGA values in the position of the character are overwritten based on the input of the accelerometer.

The implementation of the maze navigation game required various hardware and software components and modules. The *vga_controller* module was used to establish communication with the display by setting the horizontal and vertical VGA synchronization signals. A central controller module was created to create the map and character based on user input from the DE10-lite board. The *controller* module takes the switch, button, and accelerometer positions as inputs and assigns the LEDs and VGA as outputs. The two accelerometer modules were used to aid in reading the sensor's input. The *spi_control* module is responsible for reading the accelerometer input and generating two cartesian signals for horizontal and vertical reaction whereas the *spi_serdes* module establishes a connection between the accelerometer and the rest of the board.

Results

After implementation and testing, it was determined that the system conformed to the requirements and resources outlined in the proposal. Switches 0 and 1 were used to perform level selection while switch 9 was used to accurately initiate gameplay. The accelerometer accurately moved a character through a maze based on orientation. Additional functionality was required for movement control as a result of the peripheral's output sensitivity. Additionally, the accelerometer acts as an LED driver to aid in visualizing the output as a binary figure in which each LED represents one bit. Furthermore, a reset was successfully implemented using button 0.

The project required the use of several resources including the DE10 Lite board, and many premade modules. The DE10 Lite board was used as the main hardware controller of the system. The built-in buttons and switches provided a user interface for system reset, locking and level selection. Furthermore, the VGA component allowed the system to generate a series of accurate visuals. The built-in accelerometer provided an effective input mechanism to control the character. A PLL clock VGA controller and accelerometer modules were used to provide support to the author-designed modules. The PLL clock is an auto-generated Quartus module which is used to divide the onboard 50 MHz clock to 25 MHz to support the pixel rendering of the VGA system with a 60hz refresh rate. The premade lab 5 VGA controller module was utilized to help provide horizontal and vertical stabilization on the VGA system [1]. The module assists in visual stabilization through the generation of horizontal and vertical synchronization signals. The *spi_serdes* and *spi_control* modules were used to output a readable value from the sensor. These modules were taken from UC Davis's EEC180 accelerometer tutorial [2]. In its totality, the system utilized 2% of the board's available logic elements and 51% of the DE-10 Lite's total pins.

Two key challenges faced during the design of the project were the implementation of the accelerometer and the generation of collisions. The accelerometer was challenging as it was never used during the EECS 3216 or EECS 3201 labs. This novelty required the authors to perform external research to understand the general characteristics of the sensor. After initial research, a set of modules was found to effectively read sensor input from the accelerometer. These modules were then connected to the system and manually tested by orienting the board such that different accelerometer values were read. Furthermore, collisions proved to be a difficult concept as it required complex logic to determine when the character encountered an illegal boundary. To overcome this, the initial levels were designed using precise lengths. After generating the levels, the character position relative to the map was determined through simple addition and subtraction relative to the total pixel area of the screen. These positions were used to determine thresholds for a collision. In other words, if the character's location overlapped with the border of a level, a collision would occur. Upon collision, the character would return to its initial gameplay state. Through research and planning, the accelerometer and collision challenges were effectively overcome.

Works Cited

[1] A. Amirsoleimani, “Lab 5,” in *EECS 3216*, Mar. 4, 2024

[2] M. Hildebrand and B. Bass, “EEC180, Digital Systems II,” EEC180 tutorial: Using the accelerometer on the DE10-Lite Board, https://www.ece.ucdavis.edu/~bbaas/180/tutorials/accelerometer.html (accessed Apr. 9, 2024).