FINAL PROJECT: SNAKE GAME

University at Buffalo, The State University at New York

EE478 Fall 2019

The information enclosed in this document and the source code generated for this laboratory assignment were generated by the named students on this cover page. Detection of plagiarism in a submitted lab or project design document will result in an automatic failing grade in the course by all parties involved.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Written By** | Chitvan Chauhan |  | **Revision History** | | | |
| **Engineer** | Chitvan Chauhan |  | **Rev** | **Date** | **Session** | **Approved/Grade** |
| **Engineer** | Henry Morocho |  | Orig | 9/5/18 | Wed. 5PM |  |
| **Teaching Assistant** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | **EE478F19**  **LAB 0** |

# Table of Contents

[Table of Contents 2](#_Toc27148986)

[List of Figures 2](#_Toc27148987)

[1. Introduction 3](#_Toc27148988)

[1.1 Overview 3](#_Toc27148989)

[1.2 Document Scope 3](#_Toc27148990)

[1.3 Intended Audience 3](#_Toc27148991)

[2. System Design Overview 4](#_Toc27148992)

[2.1 System Block Diagram and Description 4](#_Toc27148993)

[2.2 State Machine/Process Module 5](#_Toc27148994)

[3. Testing and Verification 9](#_Toc27148995)

[3.1 Testbench and Simulation 9](#_Toc27148996)

[3.2 Objective Verification 9](#_Toc27148997)

[Glossary 9](#_Toc27148998)

[List of Abbreviations 9](#_Toc27148999)

[Hardware References 10](#_Toc27149000)

# List of Figures

**Figure 1: System Machine 4**

**Figure 2: System Machine 5**

# Introduction

## Overview

In the final project with the help of an existing codebase from lab 5 that is already set up to control the HDMI output on the Zybo board, we write the code in VHDL to create a snake game and display them at the PC using an HDMI cable. We use a 74.25MHz pixel clock is input to this component, and the hcount and vcount signals as the x-axis and y-axis lengths. We output a 720p signal to the monitor, with a resolution of 1280x720 pixels.

## Document Scope

The scope of this document is to provide with the explanation of working of different processes and state machines altogether that were created to create a snake game in VHDL. This document explains the functionality of a state machine and use of clock along with use of arrays to make increments in the length of snake every time it eats food.

## Intended Audience

This section mentions who the intended readers of this document are. For example: The audience for this document is the EE478 session TAs and professor, as well as the lab partners on team J3 in the Wednesday 7PM session.

# System Design Overview

## System Block Diagram and Description

**Process State\_proc:** In this process, the snake was assigned with initial positions and then assigned with a velocity. It was assigned to the push buttons on the Zybo board and was made to switch to different states whenever the push button was pressed.

The state machine had 5 states, all the states where assigned with a Select Value (S) which was in 4 bit Standard logic vector and whenever that value was called, there was a change in State

Idle- Has the starting position of the snake, S = “0000”

Movel- moves to the left, S = “0001”

Mover- moves to the right, S = “0010”

Moveu- moves up, S = “0100”

Moved- moves down, S = “1000”

**STATE MACHINE**

State 4

Moveu

S = “0100”

State 3

Mover

S = “0010”

State 2

Movel

S = “0001”

Blue

State 1

Idle

S = “0000”

BLUE

State 4

Moved

S = “1000”

Figure.1

Each state changes the position when the push button assigned to the Select is pressed on the zybo board in the constraints file

**Process: food\_loc**

In this process we assigned about 10 locations to the food. We used another state machine to implement the different locations of food and with the help of case statements the food was made to switch locations.

**type STATE\_FOOD is (l1,l2, l3, l4, l5, l6, l7, l8, l9);**

**signal food: state\_food := L1;**

l1 l2 l3 l4 l5 l6 l7 l8 l9

Figure. 2

The states l2, l3, l4, l5, l6, l7, l8, l9 all go back to the l1 (initial state) if the food\_idler =’1’ which ensures that if the snake collides with the border the food should go back to the first state and the game restarts

L9 is the “You won” state, the game ends after this state and the game does not restart.

## State Machine/Process Module

We created 3 different processes

1. **Clock Process**

Clock process implements pclk in the clock process.

1. **State\_ proc process**

In this process we used the state machines to make the snake move in different positions whenever a push button was pressed. We also used arrays to increment the length of the snake in the same process. At every rising edge of the clock the array index to the value of the previous array index (contained hmin, hmax, vmin and vmax) took the position of the previous array. This helped in incrementing the size of the snake and make it follow the snake’s had and corresponding tails at all times.

The flag commands were used to coordinate with the process Border\_detect in which every time, as seen in the movel state provided below, idler was the flag statement that made sure ‘idler = 1’ then the head had collided with the border and the snake should go back to the idle state and the game would restart else it should go to the corresponding states depending on the select statements and the commands provided

**WHEN  movel =>  
           if idler  = '1' t hen  
           state <= idle; else  
                        hmin <= hmin + left;  
                        hmax <= hmax + left;  
--                          
                 if S = "0100" then  
                    state <= moveu; end if;  
                 if S = "1000" then    
                    STATE <= moved; end if;  
            end if;**

All the array indexes were provided with the value of the previous index so that the tail increments and follows the head. As seen in the code provided below the ha(1), hb(1), va(1) and vb(1) are taking the previous array index value.

**ha(1) <= ha(0);  
                           va(1) <= va(0);  
                           hb(1) <= hb(0);  
                           vb(1) <= vb(0);**

\*\*ha, va, hb, vb are the hmin, hmax, vmin, vmax position coordinates of the head and tail.

1. **Object Process:**

We made a process named object which assigned colors to the snake (head and tail) and food. This process contained a set of if statements that assigned the color to the head, tail and food in the present location.

The snake’s head was assigned with a Blue color and was implemented with the help of the following if statements:

--Here ha(0) is the hmin array index for snake’s head

-- Here hb(0) is the hmax array index for snake’s head

--Here va(0) is the vmin array index for snake’s head

-- Here vb(0) is the vmax array index for snake’s head

**if (h > ha(0) and h < hb(0) and v > va(0) and v < vb(0))  then  
        red\_data <= color4\_red;  
        green\_data <= color4\_green;  
        blue\_data <= color4\_blue;**

The snake tail was assigned to the corresponding array indexes and was implemented with the help of flag statements.

-- tail   
**elsif flag(0) = '1' and h> ha(2) and h < hb(2)and v > va(2) and v <vb(2) then  
          red\_data <= color2\_red;  
          green\_data <= color2\_green;  
          blue\_data <= color2\_blue;**

**In this the the array index 2 is being assigned a color and displays as a tail**

The flag(0)= ‘1’ is the flag command that makes sure that all tails are printed and get incremented every time the snake eats the food. The flag was assigned the value of flagger in the start of the process.

We used flags to coordinate the if statements to coordinate with the collisions that were implemented in a different state.

1. **food\_loc Process**

The food was assigned with 9 different state locations and was implemented with the help of the following if statement, the below given if statement tested if the head of the snake was at the same hmin, vmin, hmax, vmax values as the food, there will be a change in state and the snake gets assigned with new h and v values.

**CASE food is when l1 =>  
if food\_idler = '1' then  
food <= l1;  
else  
 hfmin <= to\_signed(800, 11);  
 hfmax <= to\_signed(810,11) ;  
 vfmin <=  to\_signed(250, 11);  
 vfmax <=  to\_signed(260,11);  
 --(ha(0) = hfmax and va(0) = vmin)or (vb(0) = vfmin and ha(0) = hmin) or (va(0)= vfmax and ha(0) = hmin  
 if (hb(0) = hfmax and ha(0) = hfmin and va(0) = vfmin and vb(0) =  vfmax ) then  
        food <= l2;  
       flagger <= "000000001";  
        end if;**

Here food\_idler is the flag that makes sure that the food goes back to the idle position after collision with the wall and flagger is the flag command that ensures that food gets printed in the object process.

Flagger is another flag statement that is used to print the tail in object process, every time the snake eats the food.

The final state l9 made sure that once the snake has eaten all the food items the screen goes pink stating that the player won the game.

1. **Process- Border\_detect**

We created the process called border detect to keep track of the snake’s head and to make sure that whenever it collides with the borders on the screen it goes back to the idle state. This process ensured that the snake went back to the idle state and position and the food went back to the first food location. This process basically made sure that every time the snake’s head hits the border, the game starts all over again.

The use of fag statements was made in this process and was implemented in process state\_proc and food\_loc to make sure that the food and snake goes back to the initial position after collision with the border.

**if ha(0) = to\_signed(0,11) or ha(0) = to\_signed(1279,11) or  va(0) = to\_signed(0,11) or vb(0) = to\_signed(719,11) then  
          idle\_state <= '1';  
          food\_idle <= '1';  
         else  
          idle\_state <='0';  
          food\_idle<='0';  
    end if;**

The if statement checks for the border and the idle\_state and food\_idle are flag commands that we call in the state\_proc and food\_loc respectively to ensure that the snake and food goes back to the idle state upon collision.

# Testing and Verification

In this section, you will discuss the verification that your design meets the requirements. Generally, this will involve a discussion of your simulations and testbenches, as well as discussion of your physical verification on the board.

## Testbench and Simulation

## Objective Verification

# Glossary

This section includes helpful supplemental information. The subsections will depend on the lab, but may include references to online datasheets for hardware modules, supplemental information about communication interfaces, etc.

## List of Abbreviations

Engineering documents are often full of abbreviations and acronyms. Design documents generally include a list of these abbreviations alongside their full text. For example

* VHDL: VHSIC Hardware Description Language
* VHISC: Very High Speed Integrated Circuit
* FPGA: Field Programmable Gate Array
* LED: Light Emitting Diode
* CAD: Computer Aided Design

## Hardware References

* Xilinx Spartan 6 FPGA <https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf>
* Digilent Atlys development board <https://reference.digilentinc.com/atlys/atlys/refmanual>
* Xilinx ISE WebPack CAD tool, etc.