Habib University

Dhanani School of Science and Engineering

Digital Logic and Design

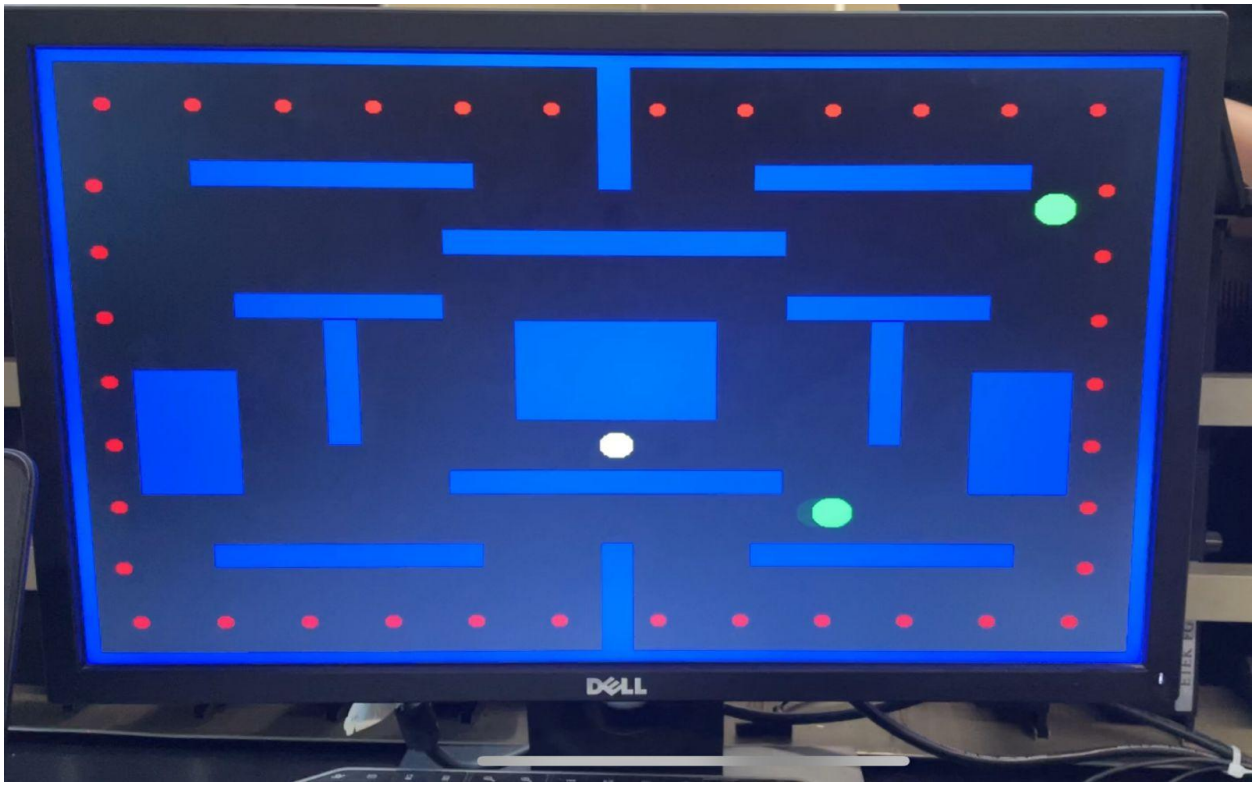
(EE-172L/CS-130L)

Project Documentation

PACMAN

Bushraa Yousuf
Dua Batool
Hijab Fatima
Zainab Haider

Table of Contents



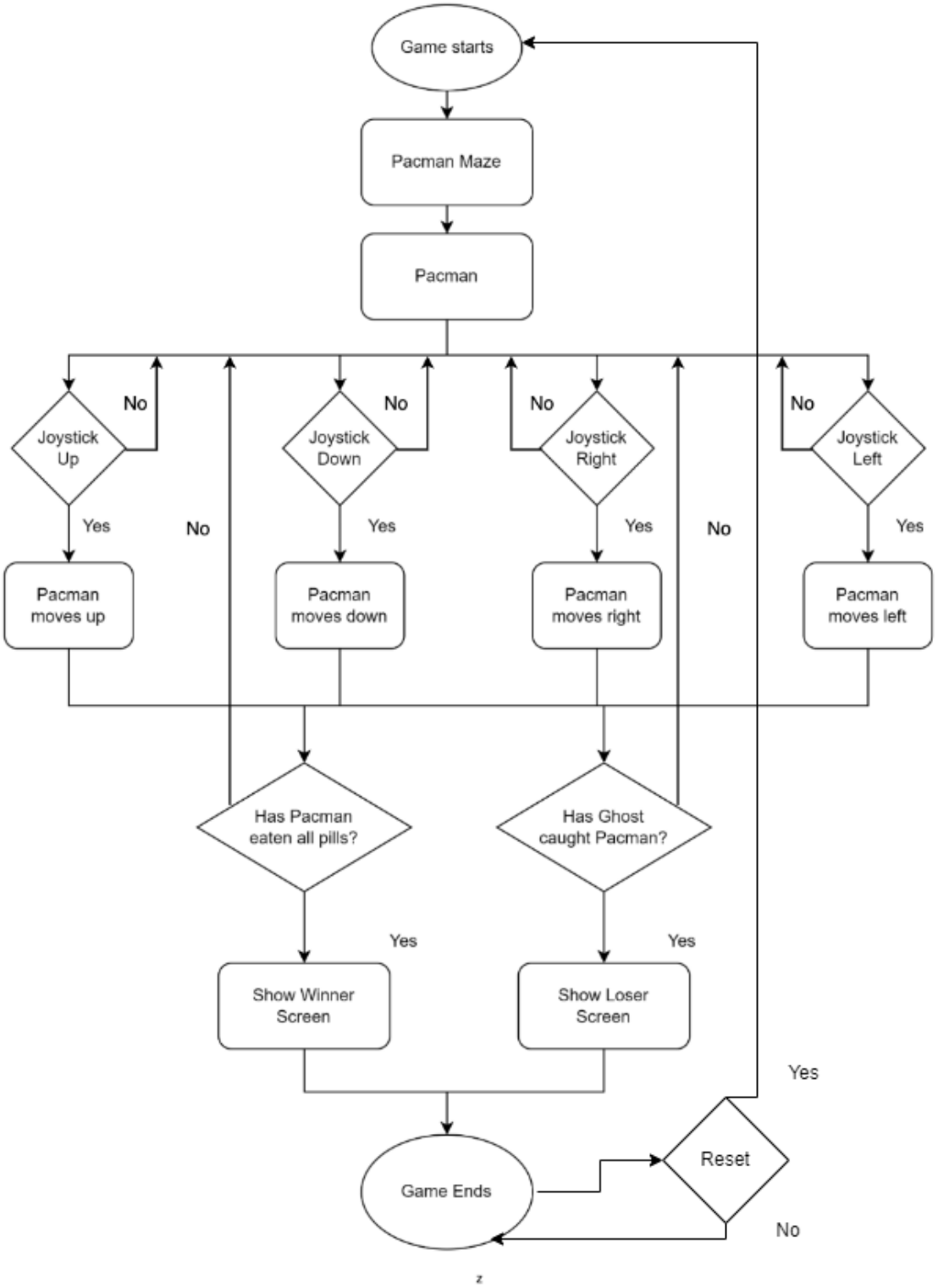
Game Features:

The objective to win this game, Pacman, is to eat all the red pills displayed on the screen while avoiding any collision with two ghosts, who would be moving constantly. Pacman would not be able to pass through the walls, as it would result in a collision. If the ghost manages to collide with Pacman, the game will be lost. But if Pacman eats all the red pills, the game will be won. After losing or winning the game, player can reset it and replay the game.

Playing Instructions:

The player can use the joystick to move Pacman in four directions across the screen: up, down, left, and right.
Player can use the switch to reset the game.

User Flow Diagram:



Input Block:

The input block will manage the input stream from the joystick and will store the direction in which the joystick has been moved and also the directions in which the joystick has not been moved.

Control Block:

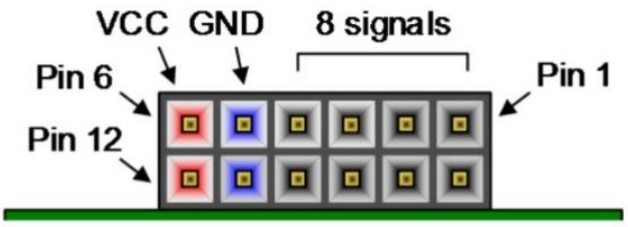
The control block will administer the display on the screen, the movements of Pacman, and the ghost. A module will be keeping track of the red pills that Pacman eats in order to remove them from the screen and decide if the game is won yet. Collisions module will deal with the collisions of Pacman with the maze walls to allow only the valid moves. In addition to this, Control block will also have a module that checks the collision of pacman and the ghost to decide if the game is lost or not. Movements of the ghosts are predefined. Furthermore, The integration of the output signals from the joystick to the display will be catered to in the control block.

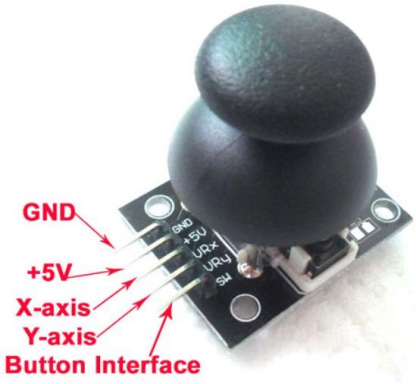
Output Block:

The output block will use the control block data to display the maze, the Pacman, the red pills, and the ghosts. It will consist of modules h_sync, v_sync, pixel_generation, and vga_controller.

Input Block Implementation:

The Pmod ports from the BASYS-3 board are used to read the outputs from the Joystick.





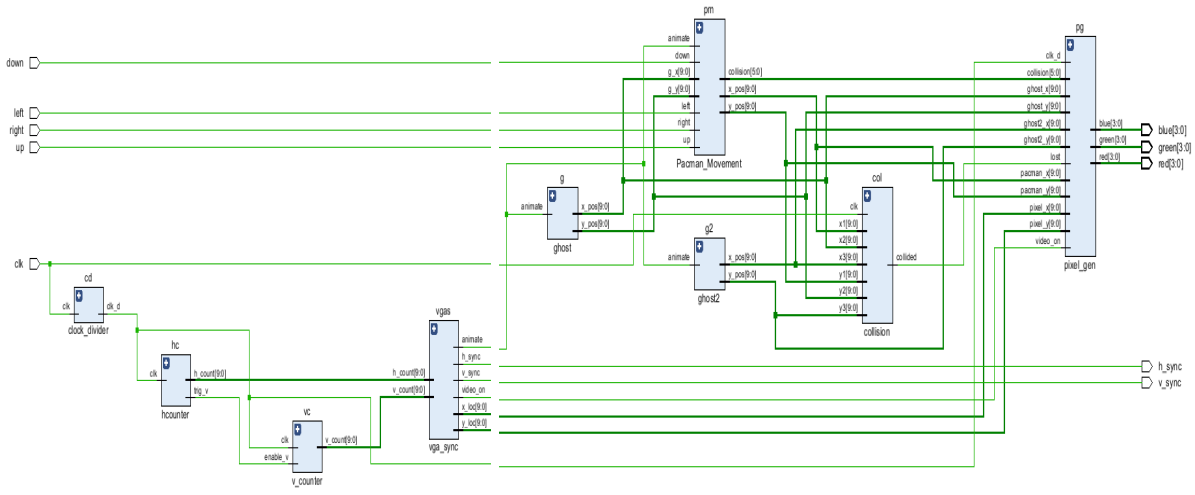
A voltage of 3.3V is supplied by Pin 6 and Pin 12 on the Pmod. We have connected Pin 6 of Pmod to +5V pin on the joystick after applying a resistance of 15.6 k ohms. This resistance is achieved through two resistors of 6.8 k ohms and two resistors of 1 ohm in series. Ground pin of the joystick is connected to Pin 5 (ground) on the Pmod. Vrx (X-axis) with Pin 2 and Vry (Y-axis) with Pin 1. The pins below Vrx and Vry (Pin 7 and Pin 8) are provided a common ground.

Output Block Implementation:

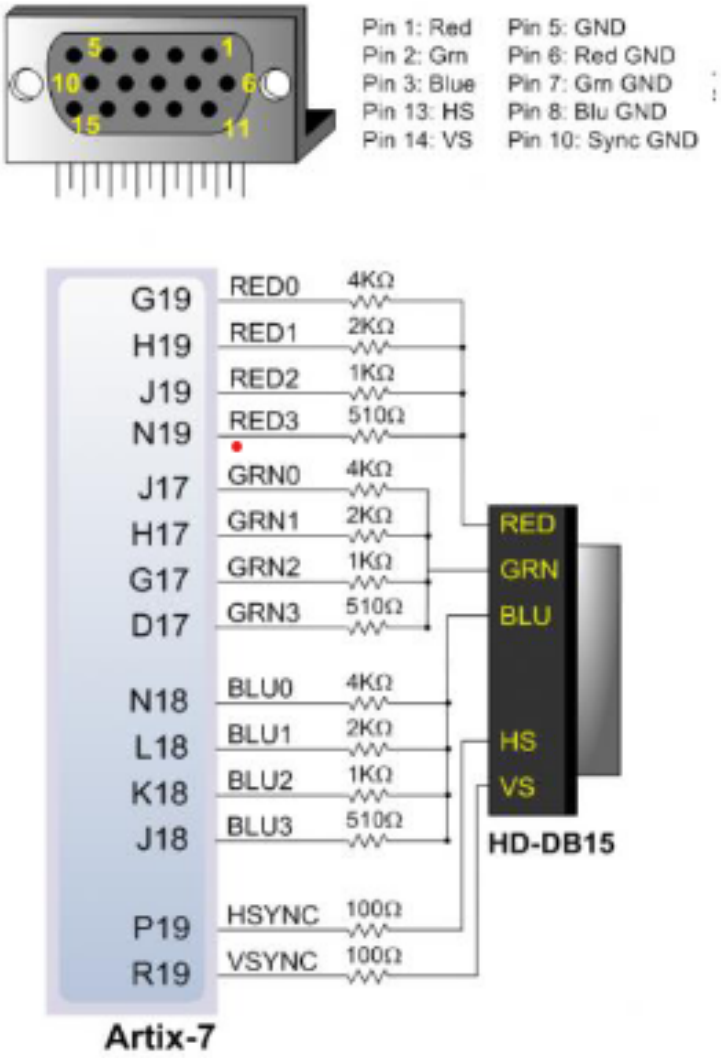
The h_counter and v_counter modules receive a 25MHz clock generated by the clock divider module. The h_ counter module increases the h_count which corresponds to the pixel_x (the horizontal pixel location) on the screen and v_counter module increases the v_count when h_count reaches its maximum value (right most corner of the screen). This is how all the pixels of the screen are accessed. The pixel_gen module assigns an RGB value to every pixel of the screen which is called by the top module (the main display module) which generates and updates the output of the screen according to the inputs coming from the Control Block.

The game screen contains a predefined maze,ghosts and pills which are fixed. The winner screen displays “WON” and the loser screen displays “LOST”. All of these screens are drawn by changing the RGB values of individual pixels.

Top Level Module Block Diagram:



Pin Configuration:

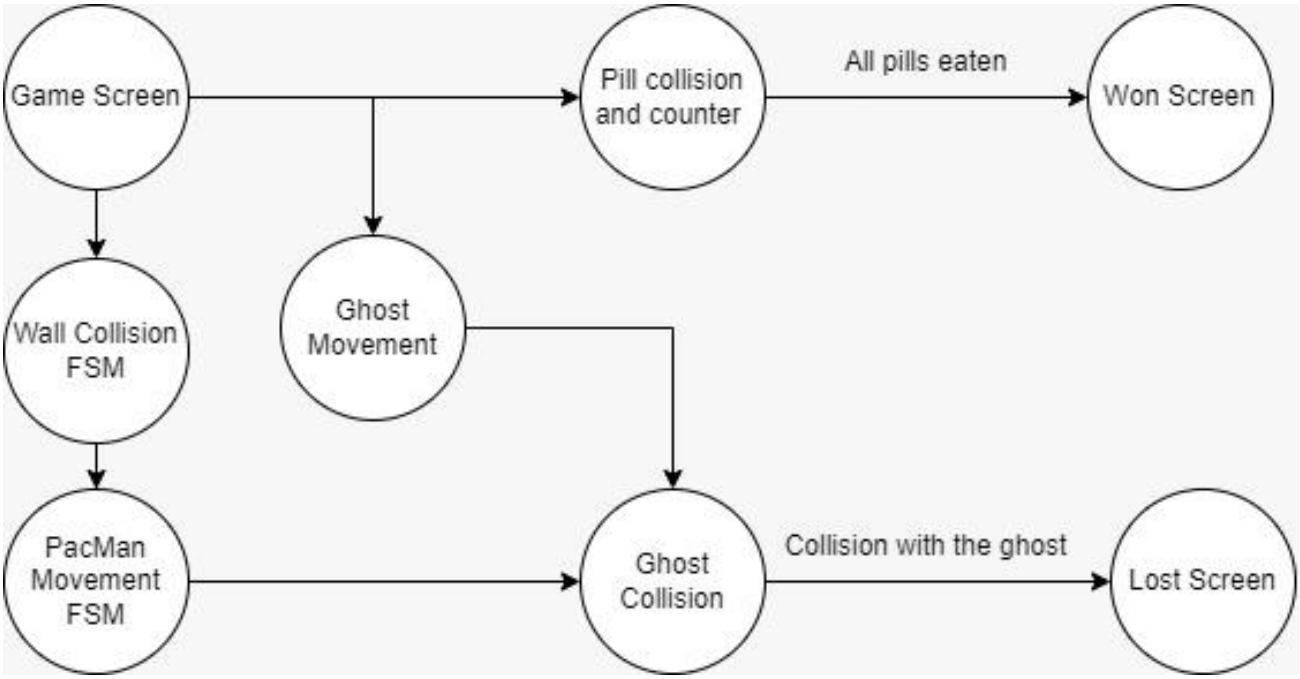


Control Block Implementation:

The main game has three displays:

- The Pacman maze display
- Won Screen
- Lost Screen

Top Level Control Block:



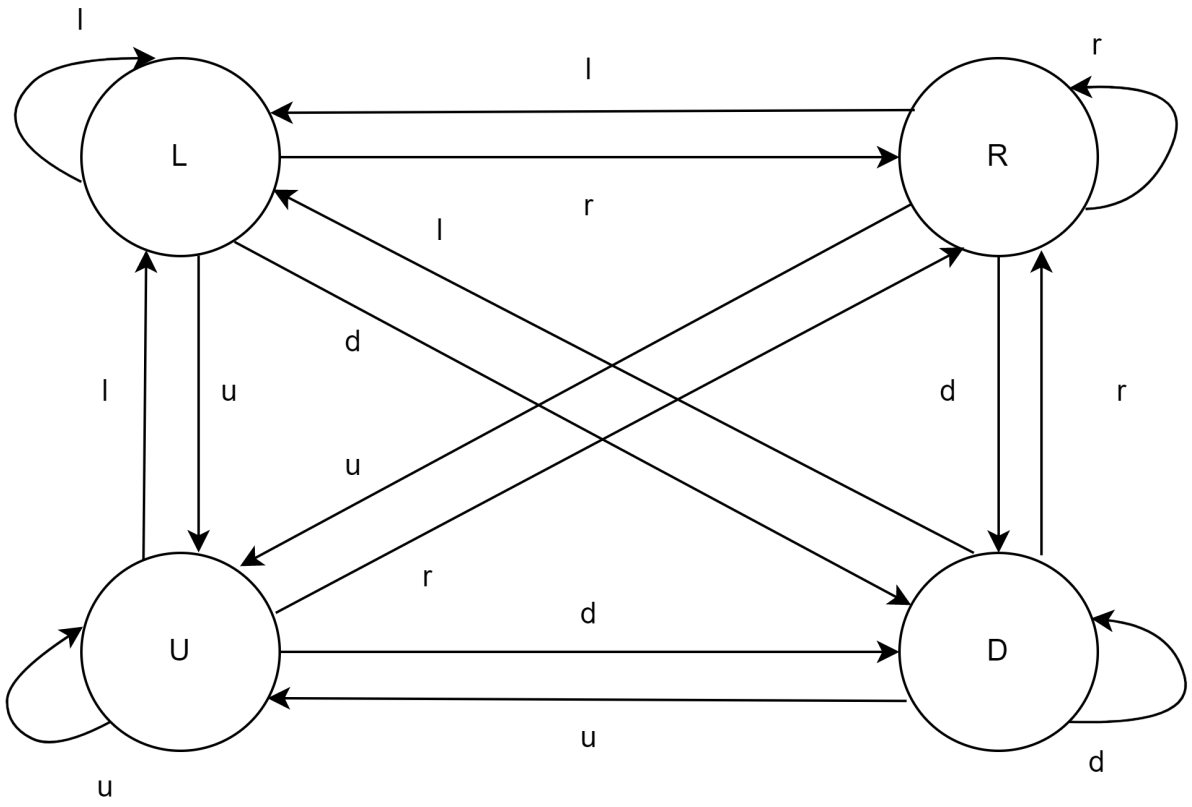
Pacman Movement Module:

In this module we are using Pacman Movement FSM to implement movement of the pacman. We have four states; up, down, right and left. Depending on the input given, the condition for the specific state is implemented and x and y coordinates are changed to implement the movement. For example, if the state is down then we are incrementing the y position of the pacman. Similarly the y position and x position are changed depending on the direction.

Movement Finite State Machine:

Implementation for the movement of the Pacman is carried out by the movement FSM.

State Diagram:



States		Inputs	
L	Left Movement	l	Joystick moving left
R	Right Movement	r	Joystick moving right
U	Up Movement	u	Joystick moving up
D	Down Movement	d	Joystick moving down

State Table:

State	Representation

L	00
R	01
U	10
D	11

Current State	**Present State**		**Input**		**Next State**	
	A	**B**	**C**	**D**	**E**	**F**
L	0	0	0	0	0	0
	0	0	0	1	0	1
	0	0	1	0	1	0
	0	0	1	1	1	1
R	0	1	0	0	0	0
	0	1	0	1	0	1
	0	1	1	0	1	0
	0	1	1	1	1	1
U	1	0	0	0	0	0
	1	0	0	1	0	1
	1	0	1	0	1	0
	1	0	1	1	1	1
D	1	1	0	0	0	0
	1	1	0	1	0	1
	1	1	1	0	1	0
	1	1	1	1	1	1

State Equations:

E = C
F = D

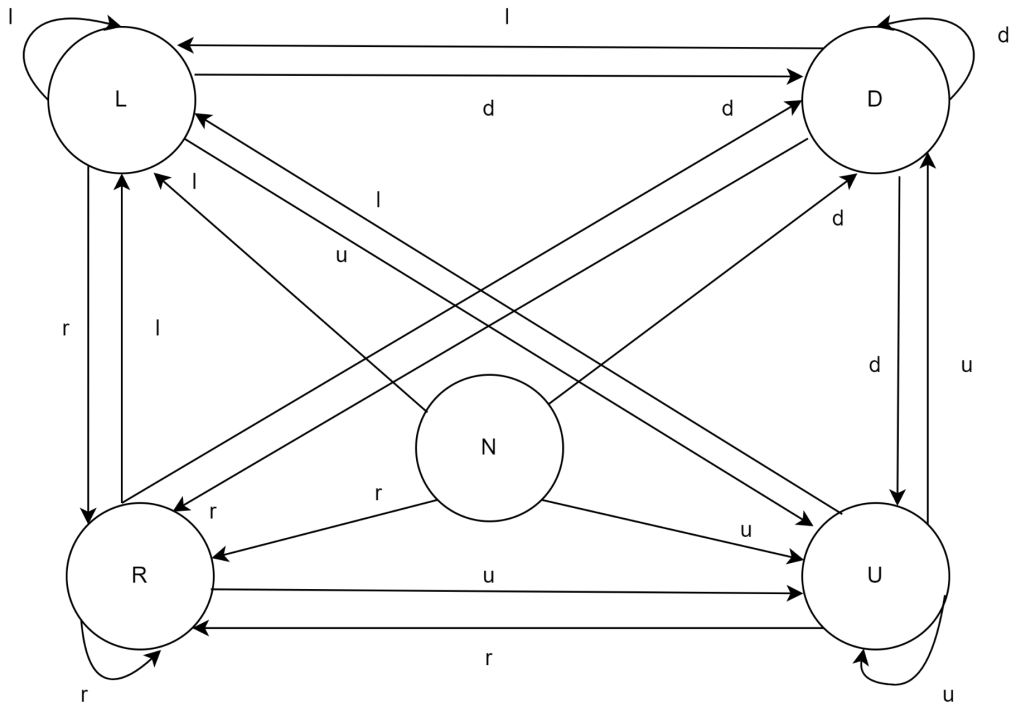
Walls Collision Module:

Since values of the walls are predefined, we are using conditions to limit the movement of pacman on the walls by specifying the range of values of x and y at which the movement of pacman is restricted. If the position of the pacman is in range of coordinates of the wall then the pacman is restricted from moving further and has to take another direction. The collision is implemented using the FSM which has five states are; left collision, right collision, up collision, down collision and no collision.

Movement Collision Finite State Machine:

Implementation of collision with walls is carried out by the Movement Collision FSM.

State Diagram:



States		Inputs	
L	Left Collision	l	Invalid Left movement
R	Right Collision	r	Invalid Right movement
U	Up Collision	u	Invalid Up movement
D	Down Collision	d	Invalid Down movement
N	No Collision		

Invalid movement implies that the coordinates of Pacman have become equal to the coordinates of any wall of the maze.

State Table:

State	Representation
N	000
L	001
R	010
U	011
D	100

Input	Representation
l	00
r	01
u	10
d	11

Current State	Present State			Input		Next State		
	A	**B**	**C**	**D**	**E**	**G**	**H**	**I**
L	0	0	1	0	0	0	0	1
	0	0	1	0	1	0	1	0
	0	0	1	1	0	0	1	1
	0	0	1	1	1	1	0	0
R	0	1	0	0	0	0	0	1
	0	1	0	0	1	0	1	0
	0	1	0	1	0	0	1	1
	0	1	0	1	1	1	0	0
U	0	1	1	0	0	0	0	1
	0	1	1	0	1	0	1	0
	0	1	1	1	0	0	1	1
	0	1	1	1	1	1	0	0
D	1	0	0	0	0	0	0	1
	1	0	0	0	1	0	1	0
	1	0	0	1	0	0	1	1
	1	0	0	1	1	1	0	0
N	0	0	0	0	0	0	0	1
	0	0	0	0	1	0	1	0
	0	0	0	1	0	0	1	1
	0	0	0	1	1	1	0	0

State Equations:

G = D.E

$H = D \oplus E$

$I = E'$

Pill Collision Module:

We have a total of 38 red pills that pacman is supposed to eat in order to win the game. Since the position of these pills are fixed, we have used conditions to check the distance between each pill and pacman in order to see if there is a collision and the pill is being eaten. If the condition is true then the pill is removed or undrawn from that position of the screen.

Ghost Collison Module:

This module is receiving as input the x and y location of pacman and the two ghosts. The distance between pacman and each ghost is calculated and checked for collision. If they are colliding then an output is generated which is sent to the pixel generation module where it is used to display the lost screen.

Won Screen Module:

To check if all the pills are eaten, we check if collision has occurred with each of the 38 pills on the screen. If the condition is yes then an output for won is generated which is used in the pixel generation module to display the won screen.

Depending on the direction in which pacman is traveling, the condition for the specific state is implemented. This logic is implemented for all the predefined walls in the maze.

Ghost Movement Module:

A predefined set of movements of the ghost are used to manage the ghost movements around the maze. The x coordinate will keep changing while the y coordinate is kept fixed at a position that allows the ghost to continuously roam in the maze in a predefined direction. Both of the ghosts have opposite directions. The coordinates of the

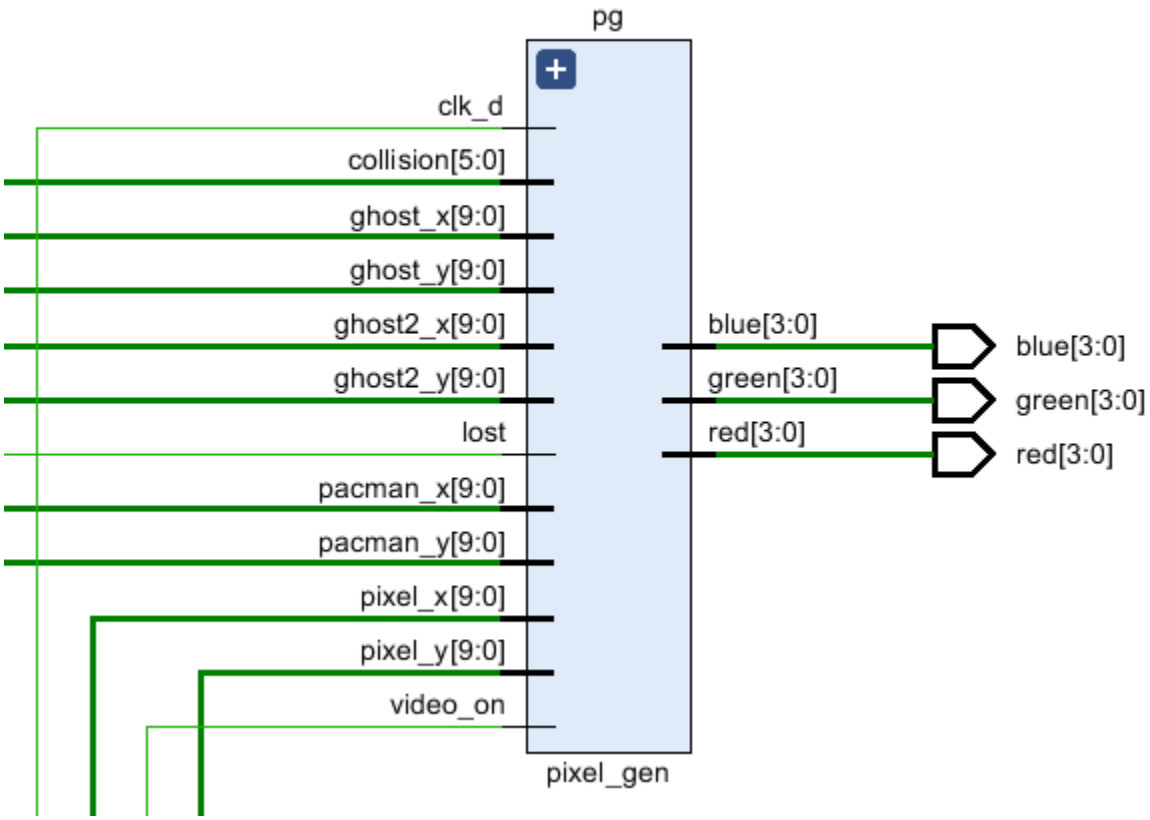
ghost generated through this are sent to the ghost collision module to continuously check for any collision with the pacman.

Pixel Generation:

The pixel_gen module assigns RGB value to each pixel according to the inputs it receives by the control block.

The pixel_gen module receives the current coordinates of pacman and the ghost which are used by pixel_gen to draw pacman and ghost in their current position accordingly.

The pixel_gen module also receives the input for won and lost, generated by the control block which allows the pixel_gen to display one of three screens accordingly. If the won input is high, winner screen will be displayed, if the lost input is high then the loser screen will be displayed and if both won and lost are low then then the game screen will be displayed.



Results:

Screen displayed when the player wins:

Screen displayed when the player loses:

Footnotes:

Source Code for Joystick was taken from a github repository:
https://github.com/Digilent/Basys3

Github Link for Project:
https://github.com/bushraay/DLD-project