LE/EECS 3216 Z - Digital Systems Engineering

(Winter 2024-2025)

FPGA-Based Multi-Board Pacman Game with Integrated AI - Project Report

Lassonde School of Engineering

23rd of April, 2025

Team Members:

Minhyeok An 217078668

Kevin Nguyen 217228255

Prithviraj Keswani 215649817

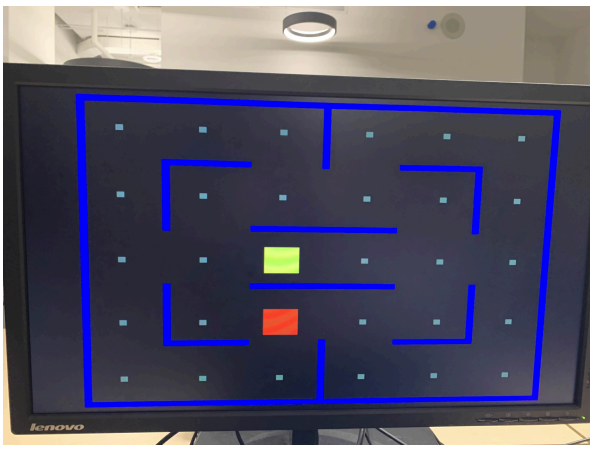
Vincent Mai 217990136

Humza Inam 218144865

Table of Contents:

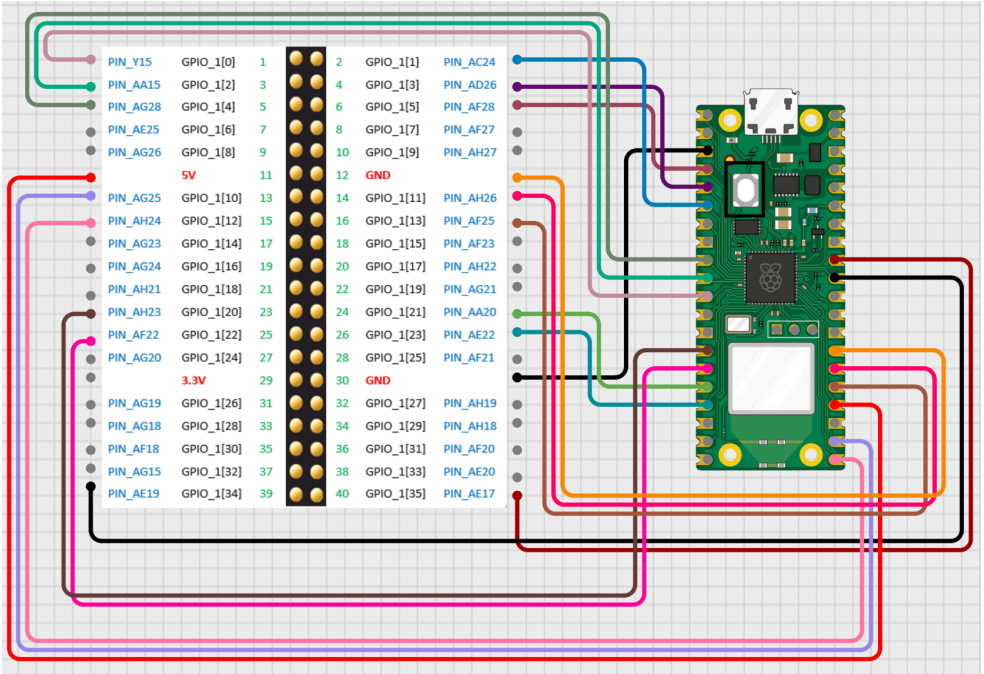
Introduction

In this project, a Pacman-style game was developed using VHDL on DE10 Lite boards and AI. As part of the course, the goal was to create an interactive game system that combines real-time graphics, player controls, and smart AI behavior. The game will have the DE10 Lite connected to the VGA port, which will display the game on the monitor. It is controlled using KEY0 and KEY1 for up/down when SW0 is on, and then switches to left/right when SW0 is off. The Raspberry Pi will be used for the AI controlled ghost, using pathfinding and Euclidean distance algorithm to adapt to the player's actions. In this report, the documentation of how the boards communicate, how the game works, and the difficulties faced when developing the project will be covered.



Implementation Details

The DE10 Lite board was used to display the maze and the Pac-Man itself. To output the maze walls on the VGA, individual walls were placed to form the overall map grid. User inputs can be inputted on the onboard keys of the DE-10 Lite (KEY0 and KEY1), with SW0 being used to toggle between left/right and up/down. SW9 will also toggle the start screen, with SW8 being utilized as the reset switch in the system. The FPGA will be connected to the Raspberry Pi Pico 2W using 19 GPIO pins with UART protocol. The x-y positions of the ghost and pac man will be given in binary values, using 3 input pins for pac man x value, 3 input pins for pac man y value, 3 input pins for ghost x value, and 3 input pins for y value. 4 input pins are also used for the wall values for up, down, left, and right. '1' would indicate that there is a wall in that position, and '0' for an open path. Finally 2 output pins from the Raspberry Pi Pico 2W will be used to output the calculated direction that the ghost Euclidean distance algorithm decided, will one more pin used as a common ground.



The winning condition of this game is to collect all 30 pellets scattered across the map, before the ghost is able to catch up with you. If the ghost is able to reach the same position as the user, the positions of pac man and ghost will reset to a default value, where the user can choose to continue their game progress or reset the game.

Hardware & Software Configuration

In order to work on this project the following hardware tools were used:

- DE10 Lite board
- Raspberry Pico Pi 2W
- Breadboard
- VGA wire
- Jumper wires
- Micro USB

The following software tools were used:

- Quartus Prime
- Micropython

The DE10 Lite was connected to the Raspberry Pico Pi using jumper wires on the breadboard connecting the wires to the GPIO pins and assigning each pin to their respective parts.

Code Explanation

Image_gen.vhd

This is the main display generator that renders the game elements to the VGA screen, it draws the maze and the pacman character which is a yellow cube and the ghosts which is a

red cube. We use component instances to handle game elements and combine output to create final pixel color for the VGA display.

Maze_rom.vhd

This file defines the maze layout for the Pac-Man game. It contains the logic for determining what's in each grid position (walls, regular pellets, power pellets, or empty spaces). The maze is created using conditional statements which define various segment and wall mapping and place pellets using the mapping coordinates.

Start_screen.vhd

This generates the start/title screen that appears before gameplay begins.

wall_mapping.vhd.bak:

A utility package that defines arrays for mapping rows and columns to specific positions in the game grid.

Pacman_top.vhd

This is the top-level entity that ties everything together. It serves as the main controller for the entire system with these key functions:

1. **Port Definitions**: Connects to the physical I/O of the DE10-Lite board (switches, keys, LEDs, 7-segment displays, VGA outputs, and GPIO pins for Pico communication).
2. **Component Instantiation**: Integrates all major modules including the VGA controller, PLL for clock generation, start screen, and game logic.
3. **Mode Selection**: Uses SW(9) to toggle between start screen and gameplay.
4. **GPIO Mapping**: Establishes the bidirectional communication with the Pico by:
    - Sending Pac-Man coordinates (pacman_row_bin, pacman_col_bin)
    - Sending ghost coordinates (ghost_row_bin, ghost_col_bin)
    - Sending wall sensor data (up_wall, down_wall, left_wall, right_wall)
    - Receiving ghost movement commands (ghost_bin0, ghost_bin1)
5. **Score Tracking**: Converts the score to BCD format and displays it on the 7-segment displays.
6. **Game State Management**: Routes the appropriate display signals based on game state and handles the frame update pulse generation.

Pacman_common.vhd

This is a package file that contains shared constants and type definitions used throughout the project:

1. Grid Configuration: Defines the fundamental dimensions of the game:

TILE_SIZE (8 pixels)
GRID dimensions (80×60 tiles)
SCREEN dimensions (640×480 pixels)

2. Color Definitions: Provides standard 12-bit RGB color constants for game elements:

   Black for background
   Blue for walls
   White for pellets
   Yellow for Pac-Man
   Red for ghosts

PACMANGHOST.py(Running on PICO)

The code implements a reinforcement learning algorithm (Q-Learning), which essentially controls a ghost in the Pacman game. The ghost essentially learns over time and chases the pacman more effectively.

The main Q-Learning States

1. State Representation
   a. The ghost's state is encoded as: (relative_x, relative_y, direction_x, direction_y, wall_up, wall_down, wall_left, wall_right), which captures the pacmans position relative to the ghost a
2. Q-Table
   a. Stored as a dictionary mapping state-action pairs to expected rewards
   b. Slowly builds up knowledge of which actions are best in each situation

3. Action Selection
   a. Uses epsilon greedy strategy, which means it sometimes random actions and exploration rate which decreases over time as the ghost learns.
4. Reward System
5. Experience Replay

The way the code Relearning works is that it reads the position and ghost from GPIO pins. It then calculates the current state and valid possible moves, then updates the r

The code interfaces with the DE10-Lite through GPIO pins on the Pico:

1. Pins 2-4 and 6-8: Read Pacman's X and Y coordinates
2. Pins 16-21: Read ghost's X and Y coordinates
3. Pins 10-13: Read wall sensors (up, down, left, right)
4. Pins 26-27: Output pins that control ghost movement

The code uses dictionaries instead of arrays which allows memory optimization in the sparse Q-table. The main game loop runs continuously, with a 0.7-second delay between moves to make gameplay responsive while still allowing time for processing.

Design Overview

A system with a split processing approach was designed, where the FPGA handles graphics, game mechanics and core timing and offloading AI decision making from the PICO. The FPGA provides parallel processing capabilities that enable smooth, real-time graphics rendering and game state management.

Instead of hard coding the ghost we have implemented Q-learning on the Pico to create ghosts that improve their strategy over time. As the ghost learns the game will become more challenging.

The bidirectional communication between FPGA and Pico through GPIO pins enables real-time decision making. The system transmits data using UART to enable intelligent decisions.

Testing & Debugging

User testing method was used to test if the wiring of the connection between the DE10-lite board and Raspberry Pi board connections were stable and giving out right outputs depending on the input. It was also used to test the mapping of the Pac-Man for the boundaries of the borders and walls, and movements of Pac-Man and ghosts. Gameplay testing was used to find what errors that might have occurred while running the code and the game, and if the Pac-Man and the ghosts meets all the desired outputs that were implemented in the code. The console output of the Raspberry Pi Pico 2W was also observed to verify if the pin inputs and outputs were being sent/delivered correctly

Challenges

One of the challenges along the process was the display using the VGA port and finding a connection to it. It took some time to figure out what the cause of the issue was and how to use them to display the required information. Another problem was with the movement of the Pac-Man. Originally there were 4 separate buttons on the breadboard that would be connected to the DE10 Lite board that controlled the pacman but there were difficulties with integrating the debouncer with the physical buttons on the breadboard. Due to time constraints it was a better decision to use the two buttons KEY0 and KEY1 to control the pacman's direction with SW0 to change the movement from up and down to left and right. After that, the process went smoothly, but we faced another challenge with the connection of the Raspberry Pi and the DE10-lite board. The ports that were assigned caused some problems, which caused some issues with the code, but in the end, the issue was solved,

which made it possible to finish the project. The main issue with them was routing the values to the functions for input and output, pin assignments were necessary to figure this problem out.

Concluding Remarks

In conclusion, this project aimed to cover the important aspects taught in EECS 3216. Previous knowledge from courses such as EECS 3201 was used to develop the FPGA VHDL code, with new knowledge acquired such as the GPIO connections between the Raspberry Pi Pico 2W and DE-10 Lite FPGA. Many challenges were encountered when creating the core functionalities of the project, and were successfully solved as a team.

Resources

"VGA Controller (VHDL)": https://forum.digikey.com/t/vga-controller-vhdl/12794