Pac-Man Game Design Proposal

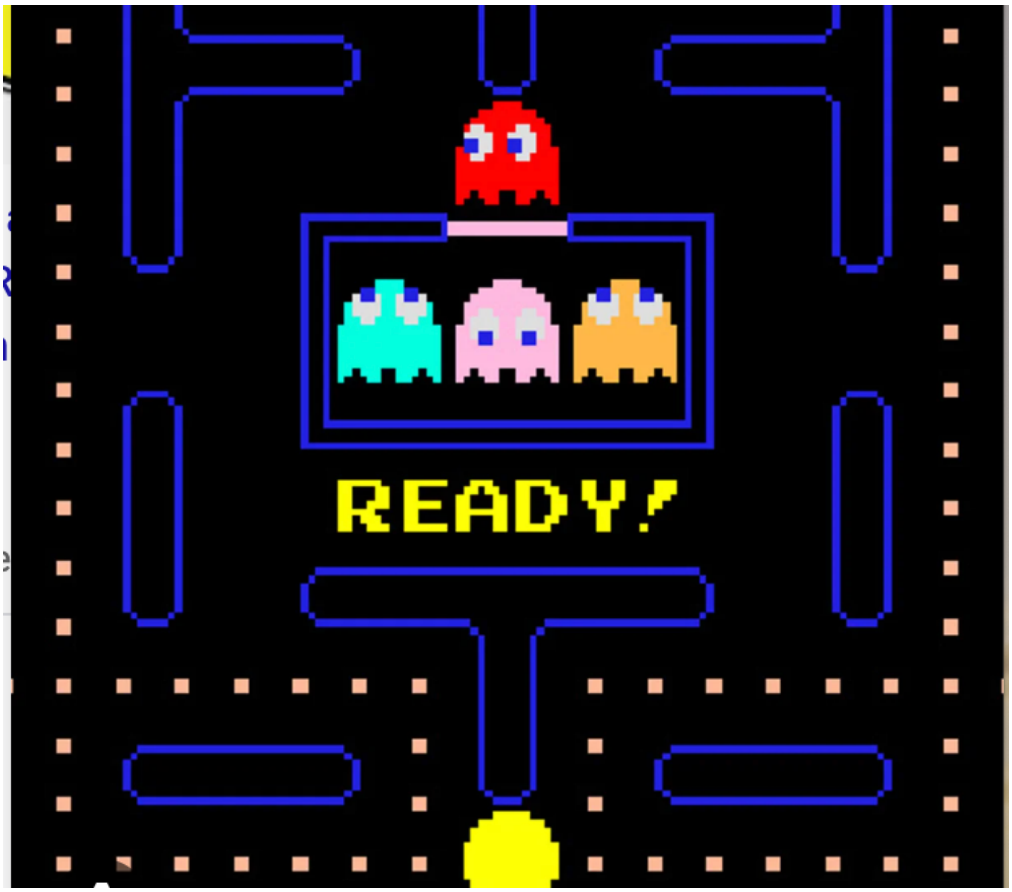
By: Anastasiia Merkudanova(am6754), Austin Gnecco(asg2290), Shishir Sharma(sks2266), Connor Marvin(cm4662)

Project Overview

The goal of this project is to design and implement a Pac-Man-style arcade game on the DE1-SoC board. The game will display a maze on a VGA monitor in which the player controls Pac-Man and navigates through paths to collect pellets while avoiding enemy ghosts. The player wins a round by collecting all pellets in the maze and loses a life upon colliding with a ghost. The game will track score, remaining lives, and game state, and will provide a simple but interactive arcade-style experience using the DE1-SoC platform.

The project will leverage both the FPGA and embedded software capabilities of the DE1-SoC. The hardware portion will focus on VGA output, sprite rendering, timing, and efficient display updates. The software portion will implement the game rules, including Pac-Man movement, ghost movement, pellet collection, collision detection, scoring, and menu/game-over behavior. This hardware/software co-design approach allows real-time graphics output while keeping the game logic flexible and easier to modify.

Hardware generates VGA timing and renders the maze using a tile-based graphics engine. Software maintains game stats and updates tile/object data. It also handle the ghost and Pac-Man UI.



Hardware Design

- **VGA Display-** The FPGA will generate VGA timing signals and display the maze, Pac-Man, ghosts, pellets, score, and status information in real time.
- **Framerate-** The system will target a smooth display refresh suitable for arcade gameplay. 60 frames per sec is fairly common for arcade style game.
- **Tile-Based Graphics Engine-** The graphics for this project will be implemented using a tile-based VGA architecture rather than a framebuffer. Instead of storing a color value for every screen pixel, the design will store:
    - A tile map that specifies which tile appears at each screen location.
    - A tile set containing the pixel patterns for each 8x8 tile.
    - A palette that maps color codes top the RGB values
    - During display, the hardware will use the current VGA pixel coordinates to determine the tile row/column, fetch the corresponding tile number from tile map memory, fetch the pixel color code from tile set memory using the local pixel position within the tile, and then use the palette to output the final RGB color.

- **Input interface-** The design will receive player inputs from onbard buttons, a keyboard, or another supported controller interface.

Software Design

- **Game Logic:** The software will manage all game rules, including Pac-Man movement, legal movement constraints based on maze walls, pellet consumption, ghost behavior, score updates.
- **Collision Detection:** The software will determine when Pac-Man collides with a pellet, wall boundary, or ghost.
- **Ghost and Pac-man UI:** The software will implement ghost movement patterns, which may initially be simple rule-based or semi-random behavior. It will also make the UI for both the ghost and the Pac Man
- **Game State Management:** The software will manage states such as start screen, gameplay, pause, win, and game over.

Hardware-Software Interface

The software will communicate with the VGA tile engine through a memory-mapped interface. The FPGA hardware will expose separate addressable regions for the **tile map**, **tile set**, and **palette** memories. The software will initialize these memories at startup and update them during gameplay as pellets are collected, score/status text changes, or screens transition between menu, gameplay, win, and game-over states. The VGA hardware will continuously read these memories and generate RGB output in real tim

Development Tools

- **Hardware:** DE1-SoC board, VGA monitor, input device
- **Software Language:** C, System verilog
- **Hardware Description / Integration:** Quartus, Platform Designer if needed
- **Development Environment:** Linux environment / terminal tools / code editor
- **Testing:** On-board verification with VGA output and gameplay debugging

Milestones

Stage 1: Build VGA timing and verify static color/test pattern output.

Stage 2: Implement tile-based rendering with tile map, tile set, and palette memories; display a static Pac-Man maze.

Stage 3: Connect software to the tile memory interface so the HPS can initialize/update tiles and colors.

Stage 4: Add Pac-Man movement, legal wall checking, pellet removal, and score updates.

Stage 5: Add ghost movement, collision detection, lives, game-over screen, and final polish/optimization.