CS M152A Lab 4 Report

Ray Hsiao, Sam Perrott

1. Introduction and requirement

In this final project, we are designing a simplified version of Pac-Man using Verilog in Vivado and implementing it on a Basys 3 board based on the Artix-7 FPGA. We are using the FPGA to drive a VGA display on the monitor. The four directional buttons on the board (UP, DOWN, LEFT, RIGHT) control Pac-Man's movement correspondingly. The center button pauses the game. The player's objective is to "eat" the dots in the maze without running into the ghost that chases Pac-Man throughout the maze – running into the ghost ends the game. Both Pac-Man's and the ghost's movement are restricted by the boundaries of the maze. The seven-segment display on the Basys 3 board shows the point total, which corresponds to the number of dots Pac-Man has eaten.

Our grading rubric consists of the following:

GUI (20%) – Maze should be enclosed on its perimeter and there should be a path from Pac-Man to all dots, Pac-Man should be reasonably represented, ghost should be distinguishable from Pac-Man, dots should be visible and in the maze, output to VGA display.

Ghost movement and interaction (20%) – The ghost will move around the maze, and the game ends if the ghost touches Pac-Man. The ghost must move in accordance with the boundaries set by the maze.

Eating dots (10%) – Whenever the Pac-Man runs into a dot, the dot should disappear and the score should be incremented.

Score display (10%) – The score should start at 0000 and increment by 1 whenever a dot is eaten by Pac-Man. The score will be displayed on the seven segment display on the FPGA board.

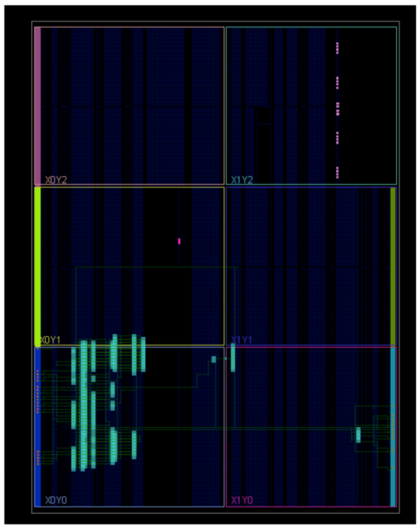
Pause button (10%) – When the center button is pressed, the Pac-Man and ghost should stop moving. Clicking the button again should unpause the game and allow the Pac-Man and ghost to move again.

2. Design description

The display module takes in the input wires of the clock signal and five buttons (UP, DOWN, LEFT, RIGHT, CENTER/PAUSE). The module outputs registers for where the VGA output displays (hsync and vsync are digital timing synchronization signals for the monitor) and what colors are drawn to the screen (red, green, and blue are analog signals for pixel data), along with wires to the seven-segment display.

The pixels on the screen start at (x,y) = (0,0) in the top left corner, with x rightwards and y increasing downwards. The positions of the maze walls and dots are hardcoded by coloring in the screen with the desired color within the desired vertical/horizontal range. Pac-Man and the ghost are colored based on constantly changing vertical/horizontal range of where the pixels are while moving. Pac-Man's movement is based on the button input from the board, and the ghost's movement is based on Pac-Man's position. Collisions are detected by checking for overlaps between the respective characters/objects by comparing their positions and widths. Each dot has a register representing whether it has been eaten, starting at 0 and being flipped to 1 once they are eaten. The sum of the dot registers is the displayed score.

For the score display on the Basys 3 board, we utilized our Clock module and the stopwatch display implementation from Lab 3. The 500 Hz fast clock is needed to cycle through the four anodes faster than the human eye can see to give the illusion that all four are simultaneously illuminated. The fast clock output is fed into our main display module, which displays a digit on the seven-segment display depending on how many dots have been eaten.



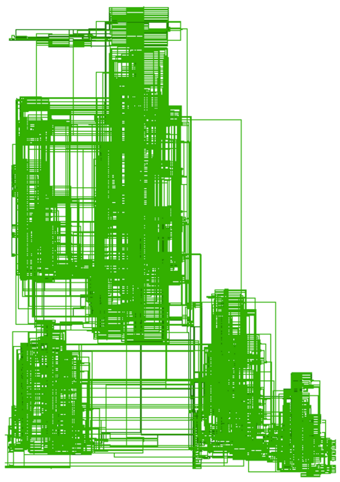


Figure 1. Synthesized design (left) and implemented design (right) generated by Vivado

3. Simulation

We developed incrementally to make sure each part was working before moving on to the next part. We started with ensuring that the VGA display was working at all. Then, we made the outer perimeter of the maze and a moving square to represent Pac-Man. At first, we did see colors move, but not in the square shape we desired. It turned out this was because the refresh rate was too high, so we used a different clock. After that, we implemented collision detection between Pac-Man and the maze. Because the coordinates are not the traditional Cartesian coordinates we are used to, we made some logic errors in the initial version of collision detection, leading to incorrect collision detection. This was fixed later on. Because we used magic numbers to hardcode the position of the maze walls, this led to some logic errors due to typos, which we also fixed, but were hard to identify until after seeing the display. We also needed to ensure that the points incremented correctly on the seven-segment display.

One confusing issue we ran into was "Poor placement for routing between an I/O pin and BUFG" during implementation. Some ideas for why this was an error was that there were too many clock signals, but since the actual circuit that our code represented was complicated and difficult to debug, we went with the (hesitant) recommendation by the compiler of adding `set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_IBUF]` to the constraint file.

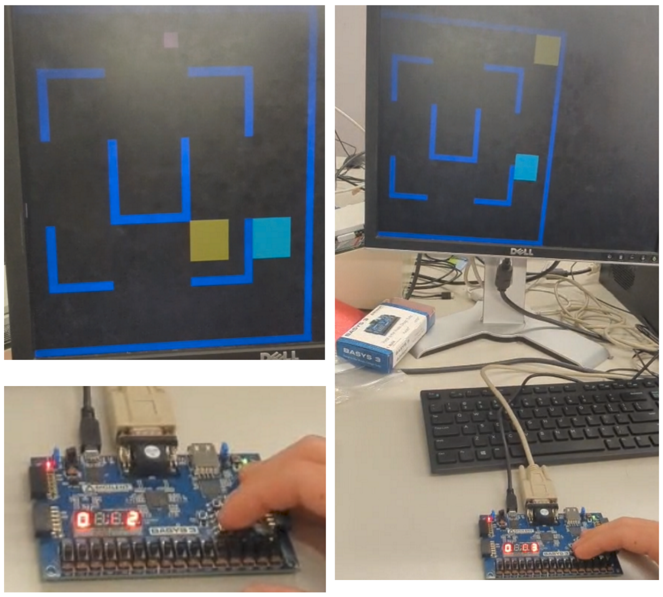
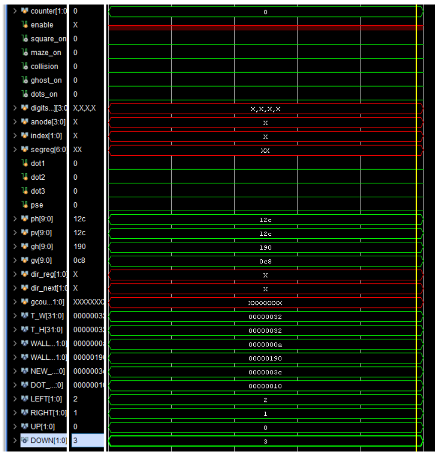


Figure 2. Pac-Man on the monitor and the seven-segment display on the board

The simulation waveforms were not as useful as looking at the VGA output for testing.



counter[1:0	0	0
enable	X	
square_on	0	
maze_on	0	
collision	0	
ghost_on	0	
dots_on	0	
digits...][3:0	X,X,X,X	X,X,X,X
anode[3:0]	X	X
index[1:0]	X	X
segreg[6:0]	XX	XX
dot1	0	
dot2	0	
dot3	0	
pse	0	
ph[9:0]	12c	12c
pv[9:0]	12c	12c
gh[9:0]	190	190
gv[9:0]	0c8	0c8
dir_reg[1:0]	X	X
dir_next[1:0	X	X
gcou...1:0]	XXXXXXX	XXXXXXXX
T_W[31:0]	0000003	00000032
T_H[31:0]	0000003	00000032
WALL...1:0]	0000000	0000000a
WALL...1:0]	0000019	00000190
NEW_...:0]	0000003	0000003c
DOT_...:0]	0000001	00000010
LEFT[1:0]	2	2
RIGHT[1:0]	1	1
UP[1:0]	0	0
DOWN[1:0]	3	3

Figure 3. Simulation waveforms

4. Conclusion

Summary of design

The buttons on the Basys 3 board control Pac-Man's movement and whether the game is paused. The display module drives VGA output to the monitor, and the pixels are colored based on the positions of objects on the screen. The widths and positions of the objects are used to detect whether they are colliding. Upon collision with a dot, Pac-Man eats the dot, and the seven-segment display on the Basys 3 board increments by 1. Upon collision with the ghost, Pac-Man turns red and can no longer move, ending the game.

Difficulties encountered

It was difficult to initially set up VGA output from the Basys 3 board. The first online guide we found was too complicated for our needs, and the one we ended up following was for a different board.

Future improvements

The maze was shifted to the left off the screen for some reason, but since that was not part of the rubric, we did not fix it. Ideally, we would have the maze centered properly. We also did not place the dots, maze, ghost, and Pac-Man exactly as the original Pac-Man game is set up because we could fulfill our rubric without doing so, but if we had more time, it would have been cool to better simulate Pac-Man. The ghost behavior is very simple, and it can get stuck in walls, so that would be another thing to change with more time.

We followed a guide (https://numato.com/kb/simple-vga-design-example-for-telesto/) for simple VGA design for the Telesto board, which is why we used 9-bit RGB when the Basys 3 board has 12-bit RGB capabilities. If we had more time, we could have modified the code to take advantage of the Basys 3's wider range of color.