Design and Implementation of Snake Game on Integrated Circuit

[Placeholder for Author List]

[Placeholder for Affiliations]

Abstract - This research article describes the design and implementation of the classic snake game on an integrated circuit. The game was first designed in verilog then implemented using an FPGA on an actual screen with a VGA connection. After the game was confirmed working it was synthesized and then placed and routed onto a 2mm x 2mm integrated circuit layout.

The motivation for this project was to use VLSI to design an Application Specific Integrated Circuit (ASIC); to implement the classic snake game. The project was chosen out of a desire to create a physical chip that displays a game that can be visually experienced and enjoyed. This chip has GPIO input pins which allow a user to communicate with the ASIC controller. The controller holds the logic and is the main brains of the snake game. The output is through VGA which can be connected to most screens. VGA was chosen because it is a simple protocol and there are adapters from VGA to almost all other display protocols. A Cyclone V FPGA board was used to test the controller. This FPGA board was really handy because it allowed the use of GPIO buttons as well as a VGA adapter. Using Cadence tools, a GDSII file was created to manufacture the physical integrated circuit.

Previous works in the field of integrated circuit games use a similar setup as this project exhibits. They start with a hardware description language then synthesize it onto an FPGA. In “The Design and

Implementation of Space Shoot Game" [1], a similar data path architecture is implemented. The game is placed on an FPGA with its output displayed visually on a screen through of a VGA connection. The game player uses an external device to change the game state with the game memory saving the world and controlling effects of world changes. The only minor difference is that the Space Shoot Game uses a mouse to change game states while this Snake Game just uses four directional buttons. The VGA implementation used in the referenced Minesweeper design is more specific example of how to implement a VGA connection.[2] This design sends the screen updates of a minesweeper game as it's world is altered. This Snake Game specifically simplifies the data flow process by only changing the world when the user presses a direction button, only saves the world when something changes, and outputs to VGA based on a significantly slower clock.

The designed chip takes input in the form of four GPIO pins, which are used by the snake controller. The controller drives the game which includes a snake that moves around the screen eating one randomly placed food at a time and then growing in size after each meal. The game ends when the snake intersects with itself or crashes into a wall. The world, food, and snake are stored in memory. The memory also records and controls the entire world of both the snake and food. Memory is accessed by the Snake Controller and VGA Controller by referencing an X and Y location in the world.

Snake memory was limited to just allow 10 additional food absorptions because of memory and chip area constraints. Food is randomly placed throughout the world after each has been eaten. This randomizer is

simply an X and Y counter based off the clock. The final integrated circuit is designed to output to a screen through VGA connection. Fig. 1 shows the block diagram.

The VGA output displays a simple three color world with a green snake, red food, and blue background. This world is a 15 by 15 block screen with each block being 32 by 32 pixels. Each block is a single color, with no further graphical implementations. This was all chosen because of memory and area constraints. Fig. 3 shows and example of the game display.

The snake moves forward based on a fraction of the input clock, more specifically 6Hz. This slow snake clock was required so game users could see the snake slowly moving across the screen, otherwise it would be too fast to play. It continues to move one block forward in the direction that was last inputted by the user on the four available directional buttons. After every meal the snake grows one block at its head.

After synthesis, the total cell area was 130182.890015 as shown in Fig. 5. The reported slack with a 50Mhz clock input was 17.21 μs as shown in Fig. 6. The reported total power usage was 0.4046mW as shown in Fig. 7. After designing the chip, the measured layout size was 1.28mm by 1.45mm for a total area of 1.856mm². When fabricating integrated circuits, a seal ring, poly fill, and metal fill are additionally required and that adds extra area. The area requirements for this chip is 2mm by 2mm, therefore this chip has plenty of room for its extra fillers and fits its' minimum size.

Each designed module was tested through modelsim, as shown in Fig. 4, and the final design was tested using the FPGA board. The output result was verified using push buttons on the fpga as the directional inputs, and by viewing the game on a monitor through vga output.

The final Snake design met size, timing and power constraints. In order to do so, the original design had to be scaled back with certain features removed or changed. State of the art snake games have many more additional features and graphical upgrades than the one that was implemented. The game was confirmed to be functional both through modelsim simulation and FPGA implementation. This VLSI integrated circuit layout was successfully implemented with no DRC or LVS errors. Fig. 2 shows the final layout. It is ready for fabrication and further testing.

[Placeholder for Acknowledgements]

References:

[1] Mishra, Kumar, and Parihar, “Design and FPGA Implementation of Space Shoot Game”, International Journal of Control Theory and Applications, Vol. 9, No. 44, 2016. Online. Available: http://www.serialsjournals.com/serialjournalmanager/pdf/1495713084.pdf

[2] Felt, Neal, Mudarapu, and Savage, “Final Project - Minesweeper”, April 2006. Online. Available: http://www.kdstevens.com/stevens/6712/minesweeper-report.pdf