ECEN 5863

Programmable Logic Embedded System Design

FPGA-BASED SNAKE GAME PROTOTYPE

By Eric Percin, Isha Sharma, Matt Hartnett

Presentation Roadmap

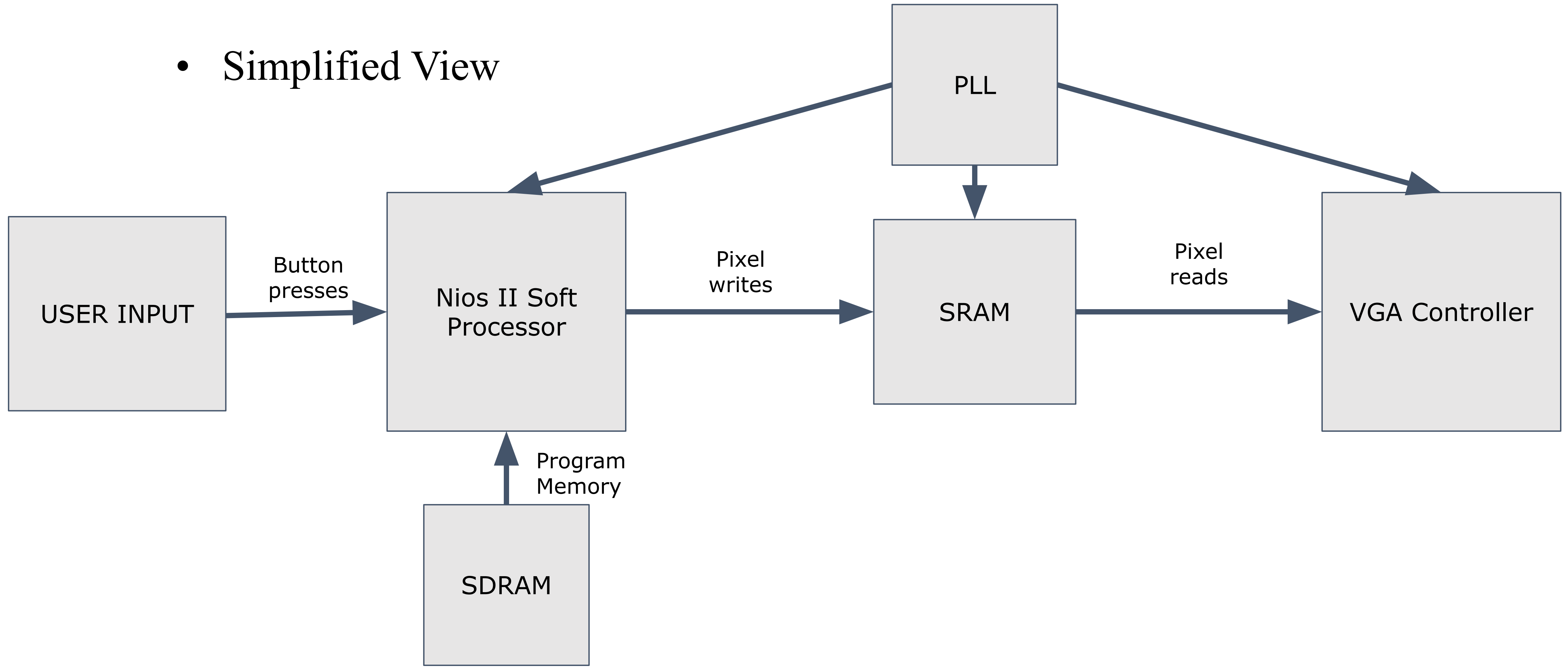
- Project Description
- System Block Diagram
- Programmable Logic Design
- Software Design
- Proof of Concept Implementation
- Demo
- Summary

Project Description

- A prototype that implements the basic functionality of the Snake arcade game using the DE1-SoC FPGA.
- VGA signals were utilized to draw and update the game graphics on the monitor at a 320x240 pixel resolution.
- Using the on-board key inputs to control the snake, the SoC outputs the state of the game to VGA monitor.
- The features of the game begin fairly basic, including the movement of the player character, the generation/collection of food, growth of the player and win/loss.
- Game logic and display control was implemented using C code running on the Nios II processor.

System Block Diagram

- Simplified View



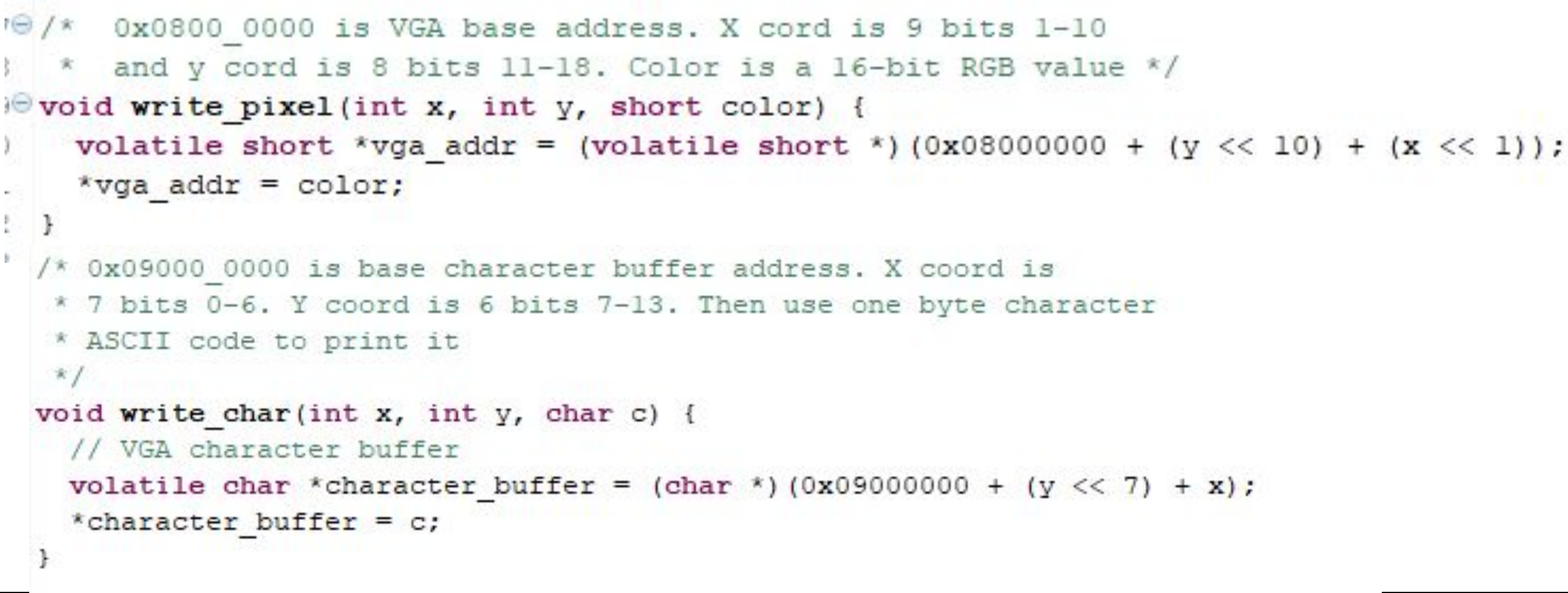
Programmable Logic Design

- The hardware was created primarily in Qsys
- Included a hard ARM processor and a NIOS II
- Connected to the VGA IP
- SRAM holds visual array that is drawn onto the screen
- SDRAM holds program memory
- Processor writes to SRAM to draw, VGA IP reads from SRAM to output video signal

Software Design

First, need to interface with VGA port

- Use base address of VGA and character buffer to write to display



```
/*  0x0800_0000 is VGA base address. X cord is 9 bits 1-10
 *  and y cord is 8 bits 11-18. Color is a 16-bit RGB value */
void write_pixel(int x, int y, short color) {
  volatile short *vga_addr = (volatile short *)(0x08000000 + (y << 10) + (x << 1));
  *vga_addr = color;
}
/* 0x09000_0000 is base character buffer address. X coord is
 * 7 bits 0-6. Y coord is 6 bits 7-13. Then use one byte character
 * ASCII code to print it
 */
void write_char(int x, int y, char c) {
  // VGA character buffer
  volatile char *character_buffer = (char *)(0x09000000 + (y << 7) + x);
  *character_buffer = c;
}
```

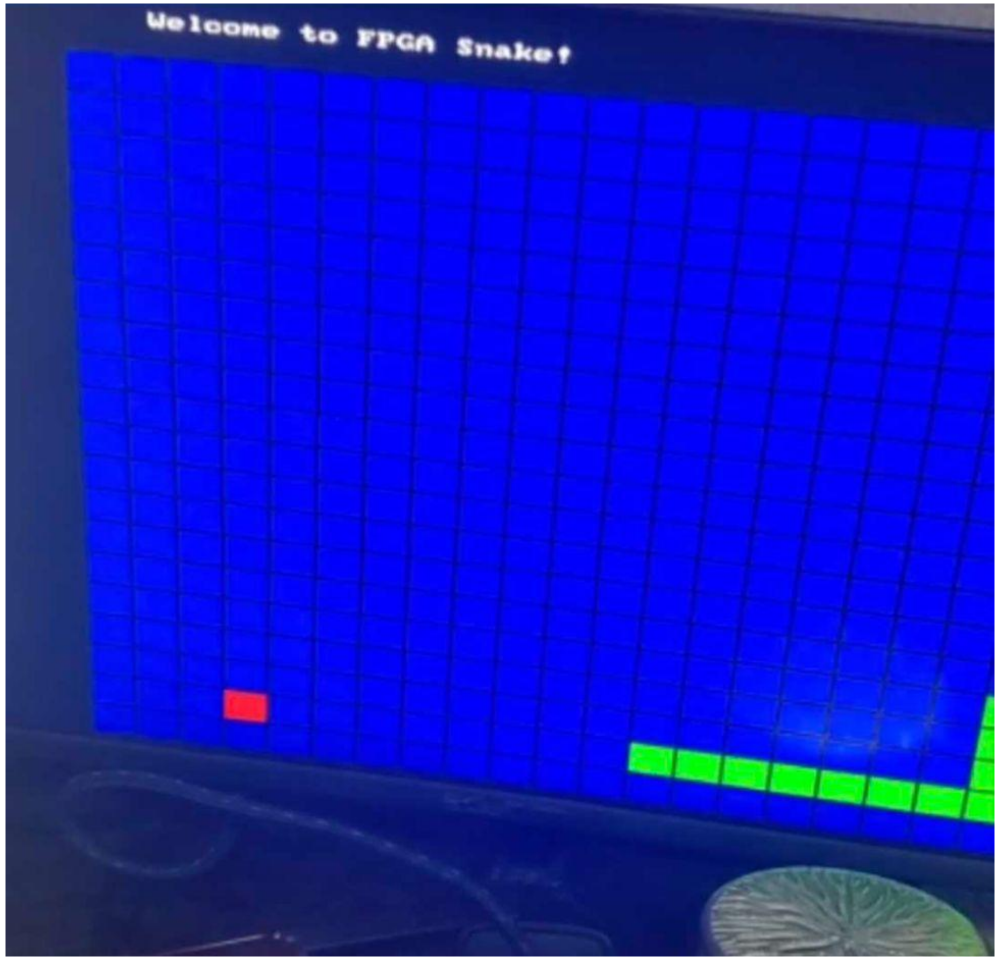
Software Design

Game logic:

- Program operates in a while(1) loop
- Snake body position tracked in an array
- First checks for collision with food
- Loops through playing field, coloring blocks based on block's status (snake, food, background)
- Checks for head/body collision, updates snake body
- Runs on with a for loop delay to provide pacing and check for user input

Proof-of-concept Realization

- A functional Snake prototype
- Move in four directions to collect food
- Grow upon collection
- Game resets upon collision with borders or snake body



Let’s play the game!

(if we can get it set it up properly)

Summary

Although there were a lot of initial challenges, we ended with a functional snake game that we could play

Thank you for listening to our presentation!

Q/A