Project Hardware Exact Rough Clone Unto Nintendo Entertainment System (Hercu-NES)

Cody Anderson
Ben Nollan
Morgan Skrabut
Ryan Price

November 20, 2017

Jeremy Thomas
Dept. of Electrical Computer Engineering
DigiPen INSTITUTE OF TECHNOLOGY

ECE 310L, ECE410L, Conjoined 3^{rd} and 4^{th} Year CE Project

Abstract

Project Hardware Exact Rough Clone Unto Nintendo Entertainment System (Hercu-NES) aims to create a Nintendo Entertainment System (NES) clone in SystemVerilog, which will be instantiated on an FPGA. The design of the NES clone will be nearly identical to the design of the actual NES console. In order to create a very similar clone, the discrete IC's on the NES will be modeled by discrete modules in SystemVerilog. Extensive validation and testing will go into checking the correctness of the timing and results of CPU and Picture Processing Unit (PPU) instructions. If we succeed in creating this clone, the Hercu-NES will be capable of running original NES games.

1 Introduction

Remembered and loved by many individuals today, the Nintendo Entertainment System released in North America in 1985 [1] is a gaming console icon of nostalgia and joyful memories. This console has been brought back to life by few individuals on the FPGA and we propose to achieve the same endeavor this year. We are motivated to recreate the NES ourselves because it is a very rare and highly sought after console with its roots deep in the hearts of many gamers including ours and those of our friends, families, and academic colleagues. We will be referencing two individuals who completed this project themselves (Dan Strother [2] and Jonathan Ganyer [3]). Our success in completing this project will be defined by our ability to run the game Super Mario Bros. on our FPGA embedded console by the end of the Spring semester.

2 Methods, Techniques, and Design

We are going to use an FPGA for this project, an FPGA was chosen over any other platform because it is the most accurate way to replicate the hardware of an NES. It also is much more capable of the task than a similar grade of microcontroller and gives us experience in defining an entire computer system in a hardware description language. We chose to use an Altera DE1 development board because it has enough general I/O accessible for this project. Also, there are already enough of them owned by the college that all the team members can have one. Project Hercu-NES is structured such that the 6502 CPU, the integrated APU, and the PPU each have their own SystemVerilog module and all of these modules will be instantiated on the same FPGA, as a larger NES module. These modules will be connected as though they were all their constituent physical packages. This structure can be seen in Figure 1.

2.1 Block Diagram

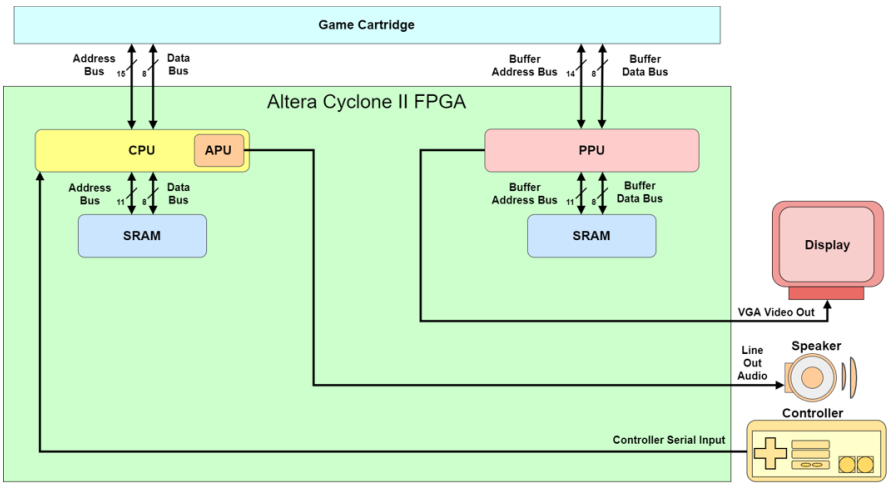

Figure 1: Project Block Diagram showing simplified flow of data and physical structure of components and modules

2.2 Central Processing Unit (CPU)

The CPU in the NES is based on the 6502 CPU, but with a few changes. The CPU operates at a frequency of 1.79 MHz and the Audio Processing Unit (APU) is built into the same package. In terms of I/O, the CPU has a 16-bit address bus and a bi-directional 8-bit data bus.[4]

2.3 Picture Processing Unit (PPU)

The NES PPU is used for generating two-dimensional scenes by using composite video output. It operates at 21.48 Mhz and outputs one pixel per clock cycle. It also has a dedicated 2K SRAM Chip for storing data which is shown above in Figure 1 below the PPU. This chip communicates with the CPU by using the CPU data bus. It also is connected to three of the CPU address lines, these are used to select the PPU register.[4] Sprites on the NES are limited to either 8x8 and 8x16 pixels which makes things fairly simple for our purposes. The PPU implements a type of scrolling in order to show larger worlds than the display's resolution alone can show. It does so using PPU internal registers which are 1-15 bits long that allows the PU to read and write to memory in order to draw the background and update the data of what is being drawn.[4] This process will be more complicated in a splitscreen setting but that is not yet in our list of goals.

2.4 Chip Interconnects

In the NES, all of the devices share the data lines and take turns outputting to them. This method won't work when using SystemVerilog to describe the

hardware as it doesn't have the capability to resolve multiple driving nets. The current solution that we intend to use is the method of separating the inputting and outputting of each device. Then all the data buses will be passed through a multiplexer. This system should mitigate some of the issues with multi driver nets and shouldn't pose any problems with data propagation.

2.5 Hardware Interface

We are going to connect a cartridge interface and two controllers to the FPGA using the two 40-pin expansion ports on the FPGA. The cartridge is a 72-pin socket, most of which are used. Each controller connector has 7 pins, four of which connect to the CPU data bus.[4]

A NES cartridge has 72 pins, but not all will be needed for this implementation. The CIC (Checking Integrated Circuit) system occupies four pins on the cartridge: two data buses, a clock line, and a reset line. Since the CIC system will not be implemented in this design, all four of those pins can be excluded. Ten expansion port pins are also present on a NES cartridge. These expansion port pins would have been used to connect to third party accessories, however no commercial products were ever developed that utilized the port (with the exception of the many accessories made for the Japanese version of the NES, the Famicom)[5]. These expansion port pins will also be excluded from this implementation, reducing the total amount of pins utilized by the cartridge to 58.

A standard NES controller has five pins: +5v, data clock, data latch, serial data, and ground [6]. The ground and power pins can be shared between the two controller inputs, bringing the total number of pins for two controllers down to seven pins. This means that a NES cartridge and two controllers can interface to the FPGA using a total of 65 pins, which is less than the 80 pins we have available through the 80 pins on the Altera DE2's expansion ports.

2.6 Audio Processing Unit (APU)

The APU is made up of five output channels. These channels include:

- Two (2) square wave generators
- One (1) triangle wave generator
- One (1) sample generator
- One (1) noise generator

Each of these channels are fed into their own DAC. Each channel is then combined in the APU Mixer. The APU Mixer is an analog circuit, but can be approximated easily with digital logic.[4]

3 Schedule and Task Breakdown

As you can see in Figure 2, we each have one distinct job in this project. The task distribution among team members will go as follows:

- Cody Anderson is in charge of overlooking the task of CPU creation
- Morgan Skrabut is in charge of overlooking the task of PPU creation
- Ryan Price is in charge of the cartridge and controller connectivity as well as the APU
- Ben Nollan is in charge of designing module framework and all testing and verification

First Half Semester Milestones (Week 7):

- Have Framework Completed.
- Implement internal registers of CPU.
- Basic video output of PPU.
- Cartridge to FPGA connectivity.

Second Half Semester Milestones (Week 12):

- Basic CPU functionality working (some opcodes execute correctly).
- Some audio outputting.
- Background Color and one or more sprite rendering.
- Complete controller and cartridge connectivity.

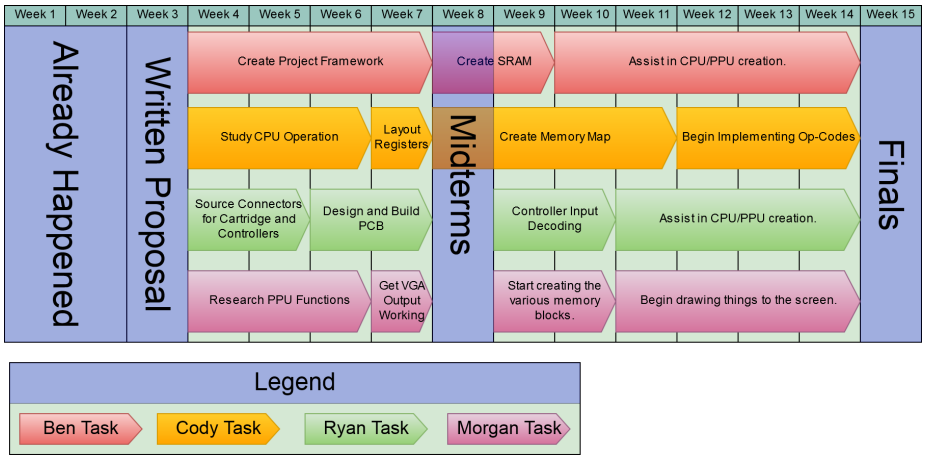

Figure 2: Project task breakdown for this semester.

4 Bill of Materials

As you can see below in Figure 3, our bill of materials includes the DE2 boards, a system and some of its official accessories, some connectors and other necessary pieces. First off, we already have some DE2's readily available to us so those don't need to be purchased. Then we decided we need an NES and games because we will be using the official system and its accessories to test, verify, and design our own version of the NES. Lastly, we'll need cables and connectors in order to be able to use the official cartridges on our own FPGA system which is part of our end-goal.

Bill of Materials							
Project:	Nock-off Entertainment System						
Team:	NoES						
Course:	ECE310L-ECE410L						
Semester/Year:	Fall 2017						
Item	**Qty.**	**Price (Each)**	**Price (Total)**	**Vendor**	**Part # (Vendor)**	**Part # (Manufacturer)**	**To Be Purchased**
Altera DE2 Education Board	4	495	1980	N/A	N/A		
Nintendo Entertainment Syster	1	86.29	86.29	Amazon	B0006TNCTG	NES-001	X
NES Cartridge Connector	4	9.99	39.96	Amazon	B01DM33TMM		X
Controller Port (female)	2	8.65	17.3	Amazon	B01H9CCDYW		X
ROM Cart	1	169.99	169.99	Amazon	B01GF0J3RO		X
Official Cart - Mario	1	22.37	22.37	Amazon	B00004SVV7		X
Official Cart - Zelda	1	22.6	22.6	Amazon	B00004SVXW		X
40-pin Male cable plugs	1	9.26	9.26	Amazon	B00VGDUFK0		X
40-pin female cables	8	0	0	N/A	N/A		
NES Controllers	4	3.37	13.48	Amazon	B07122FHM4		X
Nintendo Bit Set	1	4.98	4.98	Amazon	B009I6FWOQ		X
Card Slot Connector	4	4.89	19.56	DigiKey	A31721-ND	5530843-8	X
			0				
			0				
			0				
			0				
			0				
			0				
			0				
			0				
			0				
			0				
			0				
		Total:	2385.79				

Figure 3: Project Bill of Materials

References

[1] Nintendo. Nintendo entertainment system. (Date last accessed 9-October-2017). [Online]. Available: https://www.nintendo.com/nes-classic/

[2] D. Strother. (2010) Fpga nes. (Date last accessed 8-October-2017). [Online]. Available: https://danstrother.com/fpga-nes/

[3] J. Ganyer. Nes fpga. (Date last accessed 8-October-2017). [Online]. Available: https://jonathanganyer.wordpress.com/nes-fpga/

[4] (2017) (Date last accessed 8-October-2017). [Online]. Available: http://wiki.nesdev.com/w/index.php/Nesdev_Wiki

[5] (2017) Nintendo's expansion ports. (Date last accessed 8-October-2017). [Online]. Available: http://www.nintendoworldreport.com/feature/27664/nintendos-expansion-ports

[6] "The nes controller handler," 2017, (Date last accessed 8-October-2017). [Online]. Available: https://tresi.github.io/nes/