GameCube to Nintendo 64 Controller Converter

by

Andrew Lee

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2012

TABLE OF CONTENTS

Section *Page*

LIST OF TABLES AND FIGURES

Tables

Figures

Acknowledgements

My greatest recognition goes to Bryan Mealy, my senior project advisor, for offering complete freedom and support on my project. Though he was on sabbatical for the two quarters, he provided support and advice on my project. With his knowledge and understanding, he has not only helped me on this project, but with other circumstances that occurred during the course of my senior project. Thanks to him, this project has achieved my current goal and has the potential to keep on growing.

Additional recognition goes out to my parents, friends, and family. They planted the initial seeds for this project and beyond that. Without their support, I would have not made it this far in my life.

Abstract

This project aimed to create a Nintendo GameCube Controller to Nintendo 64 [N64] Controller converter using a Nexys2 FPGA Development Board. Nintendo halted production on the N64 controller after the introduction of the GameCube. However, the controller design, especially the analog stick, is prone to wear and tear issues under normal use. The converter aims to solve these issues by converting a superior controller for use on the N64 console, the GameCube controller.

The converter is designed based on proprietary protocol information discovered by reverse engineering via logic analyzer and oscilloscope. Additional protocol information was also obtained from available online documentation. The converter breaks down into three separate modules: GameCube interface, instruction translator, and N64 interface. In the interest of time and resources, this project only emulates the basic controller inputs and the rumble motor.

Since both devices used a bidirectional serial signaling line, the converter needed to tri-state a signal line to operate correctly. Finite state machines emulate the tri-state controller protocol, by effectively polling, receiving, and waiting. VHDL code is used to synthesize the circuit on a Spartan FPGA. This effectively develops a basic GameCube to Nintendo 64 converter. The GameCube controller works well for most N64 games and provides an improved and accurate gaming experience by removing the shortcomings of the N64 controller. Future projects can take the knowledge gained from this project to expand this adapter to multiple controllers and systems.

Introduction

Inspiration for this project came from the desire to provide a better gaming experience on the Nintendo 64. The controller for the N64 uses encoder wheels and a spring for re-centering the analog stick, making the design prone to wear and tear issues, as shown in Figure 1 and Figure 2. Over time, the wear and tear causes the analog stick to re-center improperly, and creates a dead zone. This causes problems in gameplay requiring fine movement and quick response.

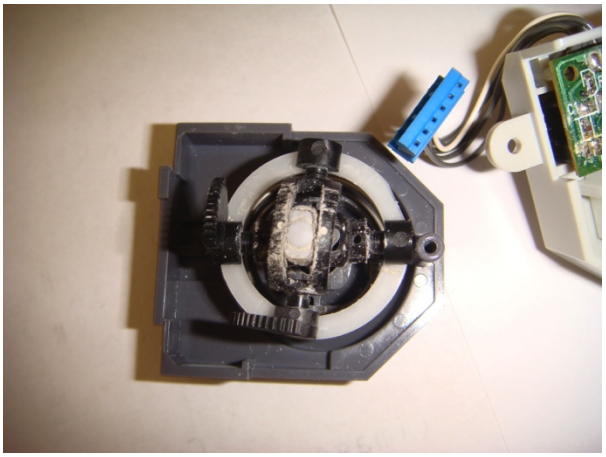

Figure 1: N64 Analog Stick Internals

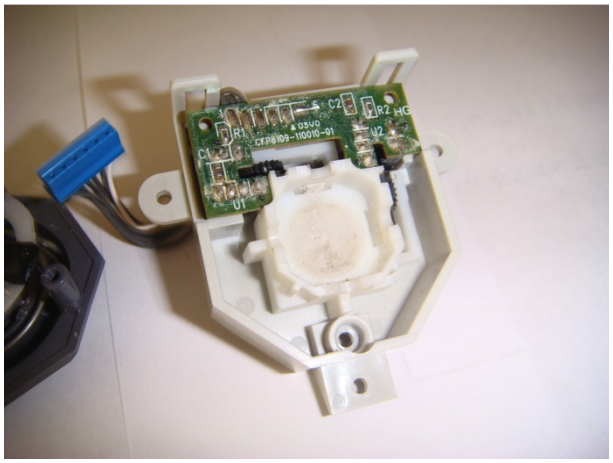

Figure 2: N64 Analog Stick Circuitry

Nintendo eventually fixed most of these problems with the introduction of the GameCube and it's controller. The newer design uses two potentiometers for the X and Y axis readings and an improved spring for longer durability, shown in Figure 3. This design continues to be used in all modern analog sticks, including those from other manufacturers.

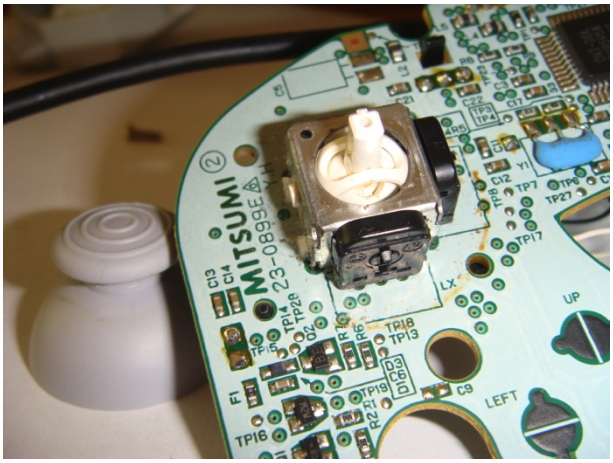

Figure 3: GameCube Analog Stick

Rather than designing a controller from scratch, this project aimed to create a converter using an FPGA in VHDL. This converter takes GameCube controller signals and converts them for use on a Nintendo 64. It also provides customizable configurations using the switches provided on the Nexys2.

Background

Figure 4: Nintendo 64 System

The Nintendo 64, shown in Figure 4, is a fifth generation video game console designed by Nintendo. The console was released in September 1996 in North America. This system greatly pushed the three-dimensional console gaming market. As such, the controller featured an analog stick, a multidirectional input device that has varying levels of sensitivity, as shown Figure 5. This was a significant departure from the traditional digital-pad, or D-Pad, which only provided 8 distinct directions and a neutral position. However, being one of the first analog sticks designed for gaming use, the analog stick is prone to wearing out from light use. The analog stick uses mechanical gears and encoder wheels, similar to a mouse, to track its shift in position. The controller will eventually stop centering correctly, as the plastic mechanism will wear out due to friction between the moving plastic parts.

Figure 5: Nintendo 64 Controller

While the controller is ergonomic, the three-way layout creates problems in accessing all available inputs on the controller. Most games are designed with this in mind, and rarely expect the user to change the position of their hands. This creates unused inputs for most games, resulting in an inefficient controller design.

Figure 6: Nintendo GameCube and Controller

The Nintendo GameCube is a sixth generation video game console designed by Nintendo. The console was released in North America in November 2001. The

console featured several departures from the Nintendo 64 design, such as disk based media, and a completely redesigned controller. The redesigned layout allows easy access to all the possible inputs of the GameCube controller, as opposed to the three-way layout present on the N64 Controller. However, this redesign also creates differences in the layout and button mappings. Some games will favor certain button mappings over others, thus a method of switching configurations must be provided to ensure usability.

Requirements

Prototyping Hardware Requirements

GameCube controller: Analyze the protocol used to communicate with the GameCube game system. These signals will be translated to equivalent Nintendo 64 signals. The GameCube controller, shown in Figure 7, has 6 discrete digital buttons, 2 analog slider pads with a digital button, a directional pad, and 2 analog joysticks. It also contains a vibration motor to provide force feedback to the user. Its U-shaped design allows both hands to easily reach any input.

Figure 7: GameCube Controller

Nintendo 64 controller: Analyze the protocol used to communicate with the Nintendo 64 game system. These will be emulated using an FPGA to interface the GameCube controller to the Nintendo 64. The Nintendo 64 controller, shown in

Figure 5, has 10 discrete digital buttons, a directional pad, and an analog joystick. It also has an expansion slot for multiple devices. The M-shaped design limits the available input depending on the holding position.

Nintendo 64 Game System: Testing and analyzing the usability and functionality of the converter. The Nintendo 64, shown in Figure 4, allows for four simultaneous controllers to be used. It is also a cartridge-based game system.

Wii Game System: Testing in analyzing the usability and functionality of the converter. The Wii, shown in Figure 8, also allows for four simultaneous GameCube or Wii controllers to be used, depending on the type of game being played. The Wii system uses disk-based media, as opposed to cartridge-based. Since the Wii features backwards compatibility with certain GameCube hardware and software, I will use this feature to perform tests on the GameCube hardware.

Figure 8: Wii Game System

Nexys2: Interpreting and emulating protocols. Originally, the ATtiny was chosen as the microcontroller to provide signal interpretations and emulation. However, a decision was made to switch to the Nexys2, as it provides several prebuilt resources that the ATtiny does not have, such as switches, leds, oscillator, etc. The Nexys2 also contains a 50Mhz clock, which is fast enough for the 4us clock cycles that an ATtiny would push. The Nexys2 also allows easier scaling to 4 controllers.

Figure 9: Nexys2 Development Board

5V Power Supply: Provide power to the Nexys2 and controllers. The GameCube controllers and Nexys2 require a 5V power supply to operate correctly, this is provided by a 5V wall wart power supply.

Completed Design Requirements

The converter should emulate N64 commands perfectly, and provide a connection port for a Nintendo GameCube controller and a connection port for the Nintendo 64 console. If possible, the design should include provisions for changing button configurations on the fly and four-player ability.

Specifications

Reverse engineered documentation for both controllers exist online, but several referred websites have expired and thus additional analysis using an oscilloscope was necessary to fill in gaps of the protocol.

Signaling:

The protocol is particular to Nintendo (Buels) (Thompson) (Cube64-DX). A line is idle high at 3.3V. A zero is sent using a 3us low followed by a 1us high. A one is sent using a 1us low followed by a 3us high. Entire communication packets are ended with a stop bit of one. Significant jitter exists during each transition. Figure 10 shows the ideal timing for a logic high and logic low. Signals are sent most significant bit first, illustrated in Figure 11.

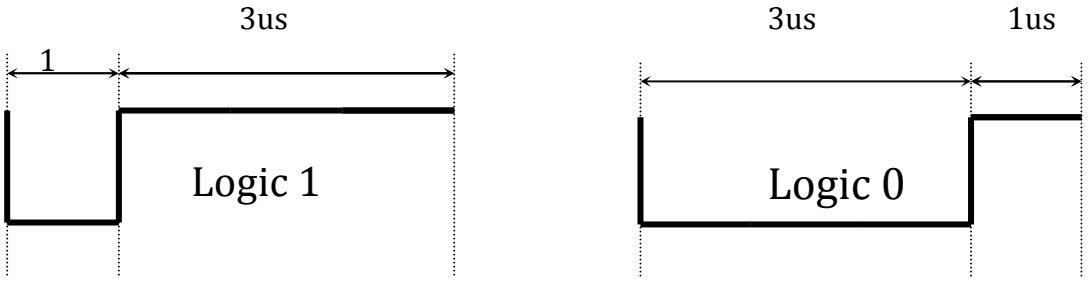

Figure 10: Logic 1 and 0 Signals

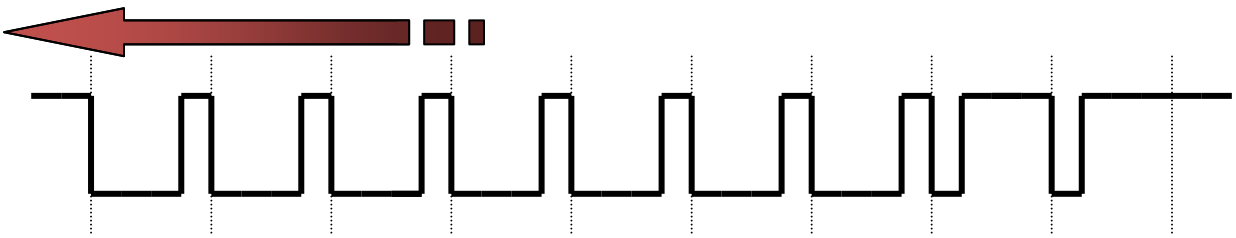

Figure 11: Sample Packet: N64 state Poll 0x01

Nintendo 64 Controller Interface:

The N64 controller pinout interface, shown in Figure 12 (Brown), consists of a 3.3V rail on pin 1, bidirectional data signaling line on pin 2, and ground on pin 3. The console provides a poll packet, and the controller responds accordingly. Different polls signify different commands. Due to the addition of the controller pak slot, the interface can become quite complicated. The scope of this project only expects to emulate the rumble pak functionality. Therefore, only a reduced set of commands needs to be implemented on the Nexys2.

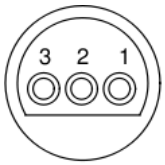

Figure 12: Nintendo 64 Pinout for Male End

N64 Controller Packet Information:

The console has been observed to send several different types of polls. The controller responds differently depending on the poll sent. Table I lists the documented and observed poll commands.

Poll	Request	Description
0x00	Identify	Requests the controller's Identity and expansion slot state. Controller responds with 0x050002 with nothing in the expansion slot or 0x050001 with the presence of a device.
0x01	Status	Requests the controller's buttons and input status
0x02	Read	Followed by a 2 Byte address, Reads 32 Bytes of data from expansion slot + CRC Byte
0x03	Write	Followed by a 2 Byte address and 32 Bytes of data, Controller responds with a CRC Byte
0xFF	Reset	Resets the controller, controller responds like an Identify

Table I: Nintendo 64 Controller Poll Packets

For a status poll, the controller responds with a four-byte packet that contains all the states of each input on the controller. Table II shows the corresponding input to bit.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A	B	Z	Start	D-Up	D-Down	D-Right	D-Left
1	0	0	L	R	C-Up	C-Down	C-Left	C-Right
2	Analog Stick X Value (8-bit Range -128 to 127)							
3	Analog Stick Y Value (8-bit Range -128 to 127)							

Table II: Nintendo 64 Controller Status Packet

N64 Controller Expansion Slot Behavior:

The N64 Controller features an expansion slot on the bottom for various peripherals. These peripherals provide increased memory for game data storage, or enhanced gameplay through force feedback motors. If the console detects a connected accessory through the Identify poll, it then initializes the attached pak by writing 32 bytes of 0xFE to address 0x8000. The console then requests a read to the same address. The controller returns all 32 bytes as 0x00 for an attached controller pak, or 0x08 for a rumble pak. Both reads and writes are followed by a CRC byte and an extra long stop bit of 2us, as opposed to the normal 1us. The rumble pak motor uses a memory latch at 0xC000, and activates on a 0x01 write, and deactivates on a 0x00 write.

GameCube Controller Interface:

The GameCube controller pinout interface, shown in Figure 13, consists of a 5V power supply for the rumble motor on pin 1, bidirectional data line 3.43V idle on pin 2, grounds on pins 3, 4, and 7, unconnected pin 5, and a 3.43V logic supply on pin 6. Various controllers and adapters have been made for the GameCube and for the scope of the project, only a standard official GameCube controller, and it's wireless equivalent the WaveBird in Figure 14, will be analyzed.

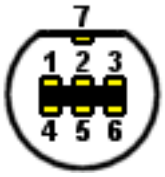

Figure 13: GameCube Controller Pinout Female End

Figure 14: GameCube WaveBird Wireless Controller and Receiver

GameCube Controller Packet Information:

The GameCube has only been observed to use two poll commands with standard and wireless controllers, as shown in Table III (James). Due to the range and variety of controllers that exist, ranging from guitars to microphones, undocumented polls exist, but will not be analyzed to the scope of this project.

Poll	Request	Description
0x00	Identify	Requests the controller's Identity, Necessary for Wavebird
0x40 03 02	Status	Requests the controller's buttons and input status, last bit is rumble, 1 on, 0 off.

Table III: GameCube Controller Poll Packets

The GameCube controller follows a similar response to a Status poll, seen in Table IV. It returns an 8-byte long packet as opposed to just a 4-byte packet due to the increased number of analog inputs.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	Start	Y	X	B	A
1	1	L	R	Z	D-Up	D-Down	D-Right	D-Left
2	Analog Stick X Value (8-bit Range 0-255)							
3	Analog Stick Y Value (8-bit Range 0-255)							
4	C Stick X Value (8-bit Range 0-255)							
5	C Stick Y Value (8-bit Range 0-255)							
6	Analog L Shoulder Value (8-bit Range 0-255)							
7	Analog R Shoulder Value (8-bit Range 0-255)							

Table IV: GameCube Controller Status Packet

Design

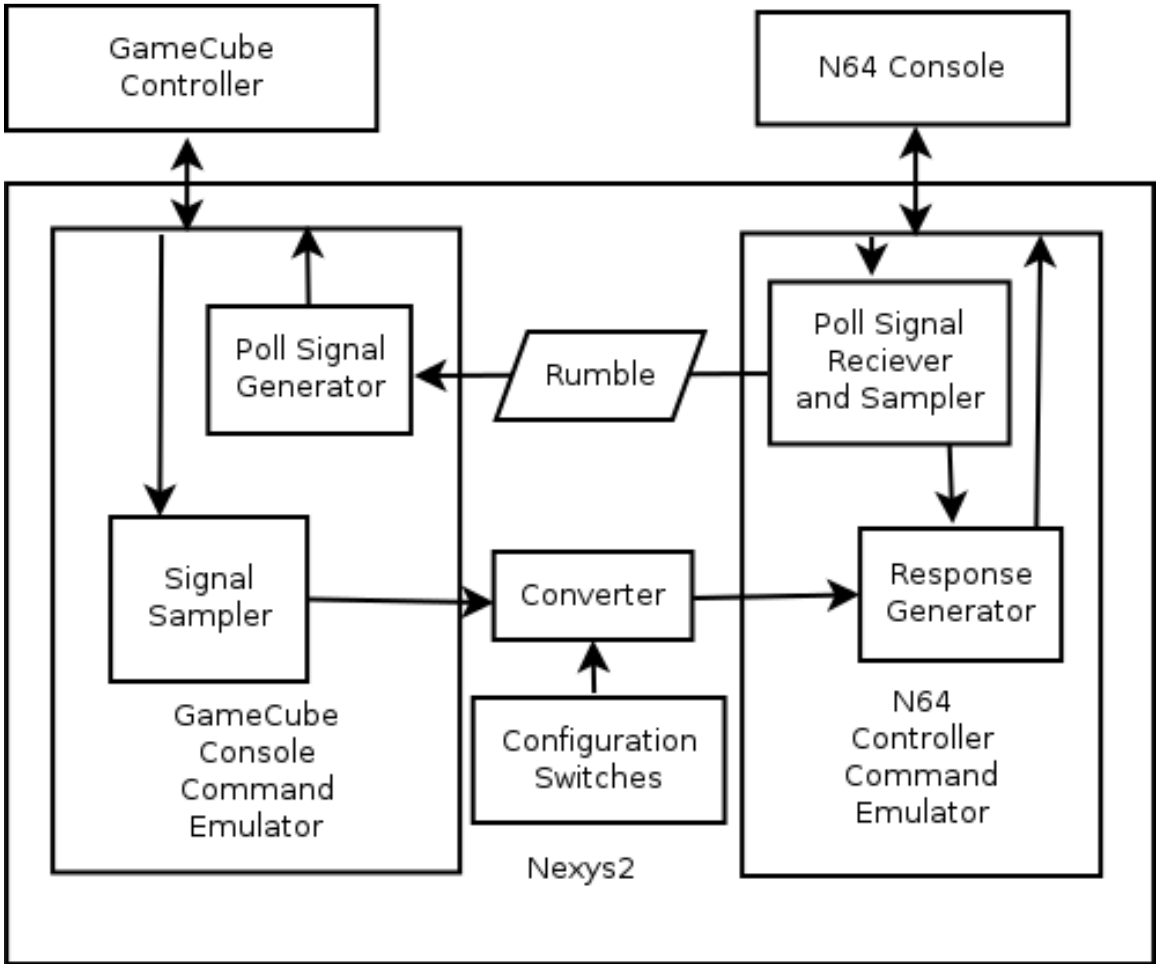

Figure 15: Converter Design Block Diagram

The design of the controller converter breaks down into 3 subsystem blocks, seen in Figure 15. One block handles communication with the GameCube controller, while the other block handles communication with the N64 Console. A command converter provides the translation between the GameCube input status and the N64 input status. A rumble signal between the emulator blocks provides the command

for initializing the rumble motor on the GameCube controller. Configuration switches allow the converter to interpret GameCube inputs differently depending on the optimal layout for various games.

Development, and Construction

Development and construction of the converter consisted of making two modules, a signal generator and signal sampler. Due to Nintendo's unique signaling protocol, a pulse-width modulation module was developed to handle outbound communication on both emulators. To handle incoming signals, a sampler, which used the falling edge as a trigger and sampled the signal in the center of the bit to provide the correct command as seen in Figure 16. A shift register organizes and prepares a parallel output signal for each poll byte. Parallel signals provide internal communication between the modules on the Nexys2 for simplicity and speed. Each component contains a finite state machine to manage the tri-state line.

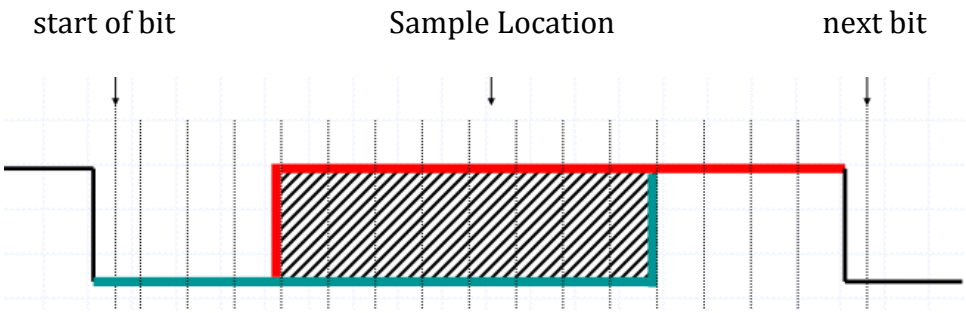

Figure 16: Bit Sample Location

The instruction converter handles the mapping between GameCube and N64 inputs. This includes a shift and a lookup table for the analog stick values, which apply a slight curve, shown in Figure 17, due to the non-linear nature of the original N64 controller's gears and encoder wheels. It also provides various button configurations, such as mapping the X and Y buttons to difference C buttons for

different games. Thresholds handle converting the C-stick position to C buttons and the analog shoulder triggers to digital should button presses. Table Vshows the button and input mappings of the controller and it's equivalent function.

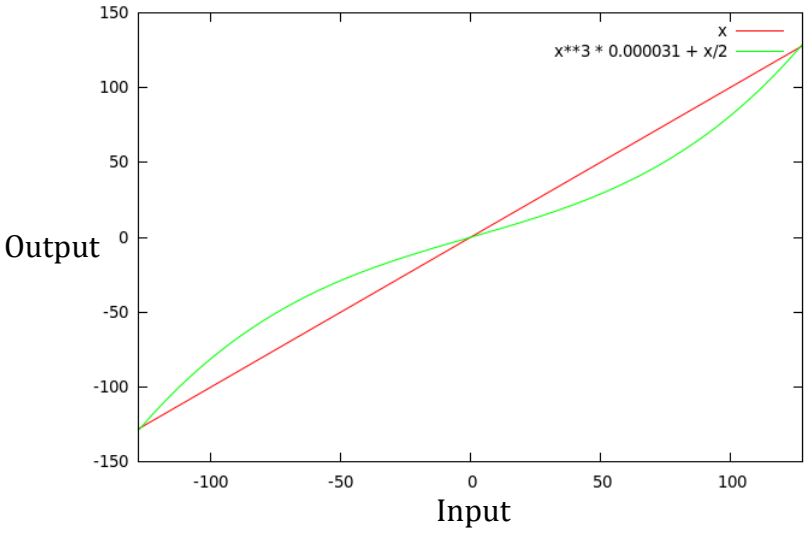

Figure 17: N64 Linear vs. Curve Analog Stick Input

GameCube Input	Default N64 Output	Alternative N64 Output
A Button	A Button	-
B Button	B Button	-
Start Button	Start Button	-
X Button	C Down	C Right
Y Button	C Left	-
Z Button	C Right	C Down
L Trigger	Z Button	L Button
R Trigger	R Button	Z Button
L Button	-	-
R Button	-	-
D Pad	D-pad with D-Up as L	D-pad
Analog Stick	Analog Stick	-
C-Stick	C Buttons	-

Table V: Button and Input Mappings

Testing and Test Results

Testing

The converter was testing using the setup shown in Figure 18. On power up, the Nintendo 64 checks for a controller on each port. Most games only check for controllers upon boot up, and thus unplugging or plugging in any controllers does not change the ones recognized by the system. If no controllers are found, the game will display the error message “Not Controllers” or similar, and be unplayable.

The converter tests simply consist of hooking up the Nexys2’s IO port onto both a GameCube controller and N64 console’s bidirectional data line. The protoboard uses spliced GameCube and N64 extension cables to provide the pinout interface required by the controller. The correct implementation will result in the console recognizing the converter as an official controller with the rumble pak attached. The GameCube controller will then function identically to a N64 controller with a rumble pak inserted.

Figure 18: Converter Test Setup

Test Results

The converter replicates N64 commands perfectly. Games respond as though a normal rumble pak equipped controller is attached. Use other play testers, some lag exists due to the converter requiring a little bit of time to translate commands and signal propagation. Figure 19, Figure 20, and Figure 21 show the system running multiple games and adequately working.

Figure 19: Super Mario 64 Running with Converter

Figure 20: Mario Kart 64 Running with Converter

Figure 21: Super SMASH Bros. Running with Converter

Conclusion and Recommendations

Conclusion

The controller converter works fairly well. The Nexys2 provides enough resources to produce four internal prototype designs, allowing four players to use one Nexys2 as a converter for four controllers. This project greatly increased knowledge of finite state machines, FPGA circuit design, and VHDL coding, as timing was crucial to each element of the design. However, there was not enough time nor resources to send and manufacture the FPGA design on a microchip. An actual system on a chip design would make integration with hardware much more elegant, compared to this prototype.

Recommendations

For future projects, this concept should be expanded to multiple controller devices and consoles. As gaming systems age, replacement parts become harder to find, especially with moving parts in controllers. Thus, suitable substitutes and replacements can be produced to provide this market. The design can be shrunk down to a smaller profile, allowing users to easily switch between multiple systems and devices.

Bibliography

Brown, Andrew. GitHub: Gamecube-N64-Controller. 10 Dec 2009. 15 May 2012 <https://github.com/brownan/Gamecube-N64-Controller/blob/master/curve.png>.

Buels, Joseph Weaver and Rob. Program and Hardware Design. 15 May 2012 <https://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2002/jew17/lld.html>.

Cube64-DX. Jacques Gagnon. 15 May 2012 <http://svn.navi.cx/misc/trunk/wasabi/devices/gchub/firmware/>.

James. Nintendo Gamecube Controller Protocol. 11 Dec 2002. 15 May 2012 <http://www.int03.co.uk/crema/hardware/gamecube/gc-control.html>.

Mealy, Bryan. CPE 169: Digital Design Laboratory. 15 May 2012 <http://courseware.ee.calpoly.edu/cpe-169/>.

Thompson, Kirren. micro-64-controller. 15 May 2012 <http://code.google.com/p/micro-64-controller/wiki/Protocol>.

Appendices

Appendix A: Senior Project Analysis

GameCube to Nintendo 64 Controller Converter:

Andrew Lee

Bryan Mealy:

• Summary of Functional Requirements

The design of the project allows the usage of four GameCube Controllers to be used with a Nintendo 64 game system. This emulates a GameCube controller port and a Nintendo 64 controller to produce this functionality.

• Primary Constraints

The primary constraint for this project was cost and processing speed. Some considerations were made using an Arduino, but this limited operations to a single controller, yet was $40 per device, making it expensive for a four-person setup. The Nexys2 allowed extension to four players, yet at the significantly reduced cost of $100

• Economic

This product has very small economic impact. The purpose of this project was to produce a small converter for game systems, thus the market for such device would be very small. Manufacturer costs would be outsourced and minimal.

• **If manufactured on a commercial basis**

Estimated Number of Devices Sold Per Year: 1000

Estimated Manufacture Cost for each Device: $50

Estimated Purchase Price for Each Device: $80

Estimated Profit Per Year: $30000

Estimated Cost for User to Operate Device, Per Unit Time: $1/month

• **Environmental**

Environmental impacts should be minimal related to manufacturing, as this small product will only use minimal amounts of metal, plastic, and silicon. There should be very little waste produced due to the low volume of production this product would have.

• **Manufacturability**

Since production would be very small scale, finding a company to produce this product on an assembly line may be difficult for cost reasons.

• **Sustainability**

With the lack of moving parts and unexposed circuits, this device should last several years as long as the user is willing to keep their game consoles. The prototype is very large, but should be shrinkable to a smaller footprint. Expanding this project to multiple game consoles would increase the footprint of the device, due to the number of unique connectors on each device.

• **Ethical**

No ethical issues should result in the production, usage, manufacture, or misuse of the project.

• **Health and Safety**

No health and safety issues should result in the production, usage, manufacture, or misuse of the project.

• **Social and Political**

No health and safety issues should result in the production, usage, manufacture, or misuse of the project. This project should benefit those that desire alternative input options to various gaming systems. Thus, this should prevent systems from being discarded.

• **Development**

Extensive VHDL, FPGA, and finite state machine design knowledge was gained over the course of this project. Additional techniques were developed for oscilloscopes and logic analyzers.

Appendix B: Parts List, Cost, and Time Schedule Allocation

Nexys2: $100

Arduino: $40

N64 Extension Cables: $20

GameCube Extension Cables: $30

Protoboard: $10

Total Project Cost: $200