Vic Vector: FPGA Game Emulator Based on *Tempest* by Atari©

Robert Baker, Tony Camarano, Drew Hanson, Robert Higginbotham

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Vic Vector is intended to re-create the classic arcade game *Tempest* using a modern design. It preserves the original functionality and quality of the game using a Field Programmable Gate Array (FPGA) and updated analog circuitry. The digital components such as the microcontroller and vector-graphics processor are modeled using Verilog HDL and programmed to the FPGA. The digital output from the FPGA is processed using analog components to produce vector-based voltage changes. These changes represent XY coordinates used to generate vector graphics, a common characteristic of many arcade games created in the 1980s. Rather than using a traditional display, the analog voltages are fed into a galvanometer laser which then draws the game images to be displayed on any surface.**

***Index Terms* — FPGA, Verilog, Vector Graphics, Game Emulation, Galvanometer Laser.**

I. Introduction

Vic Vector is a game emulation project which preserves the classic arcade game *Tempest* by Atari© using modern technology. Using the original schematics and specifications for *Tempest* as a foundation, the digital and analog components are re-created with Verilog HDL and an updated hardware design. The primary objective was to maintain the game characteristics, qualities, and flaws while implementing a more efficient, effective, and robust design. The project combines a Xilinx Spartan 3E 500K FPGA – a moderately small-scale chip – with a prototyping board containing the analog components such as the Digital-to-Analog Converters (DAC) and analog op-amp integrators. The FPGA contains all of the Verilog HDL modules which model the digital components of the game. These include the 6502 Microprocessor, Atari© Math Box, Vector Generator, and game memory. The analog circuitry processes the digital output from the FPGA to generate voltages changes in X-Y coordinates. The Laser Galvanometer Scanner CW20 receives these voltage levels and the game images are produced. The project uses a game controller similar to the original. It is a spinner wheel controller that yields a data and clock input, which dictate a player's position in-game.

II. System Overview

The project is comprised of three main components: input circuitry, Verilog modules, and output circuitry. A high-level block diagram of the design components and interfacing is shown in Fig. 1.

A. Input Circuitry

The inputs include a spinner wheel controller and four push buttons. The spinner wheel controller allows the user to navigate through the sixteen different positions on any given level of the game. The rotational direction is binary in nature and is determined using the data and clock lines. Once this signal has been processed, it is read into the FPGA. The push buttons are active-high and are tied to game controls such as "Fire" and "Zap." These signals are sent to the same input module as the spinner wheel controller.

B. Verilog Modules

The FPGA is programmed with several Verilog modules as shown in Fig. 1. The Pokey Control module receives input from the push buttons and spinner wheel controller. This information is relayed to the 6502 Microprocessor module. Both the Pokey and 6502 modules are open-source VHDL files. The 6502 module accesses the Math Box, Vector Generator, and Memory components via a pre-defined memory map and address decoder. It is responsible for executing the game ROM instructions. This includes controlling Math Box operations and writing instructions to the Vector Generator. The Math Box consists of a set of ROMs and ALUs which perform the mathematical operations necessary to generate the vector graphics. This component is a unique feature of Atari© games. Using pre-defined memory, it generates the appropriate mathematical data for the game. The 6502 uses this data to write instructions to the Vector Generator. The Vector Generator is the vector-graphics driver for the system. It is essentially a simple microprocessor containing a small set of microinstructions. These instructions, generated by the 6502 microprocessor, dictate the vector drawing, scaling, and timing parameters for the digital output of the FPGA. Additionally, the Vector

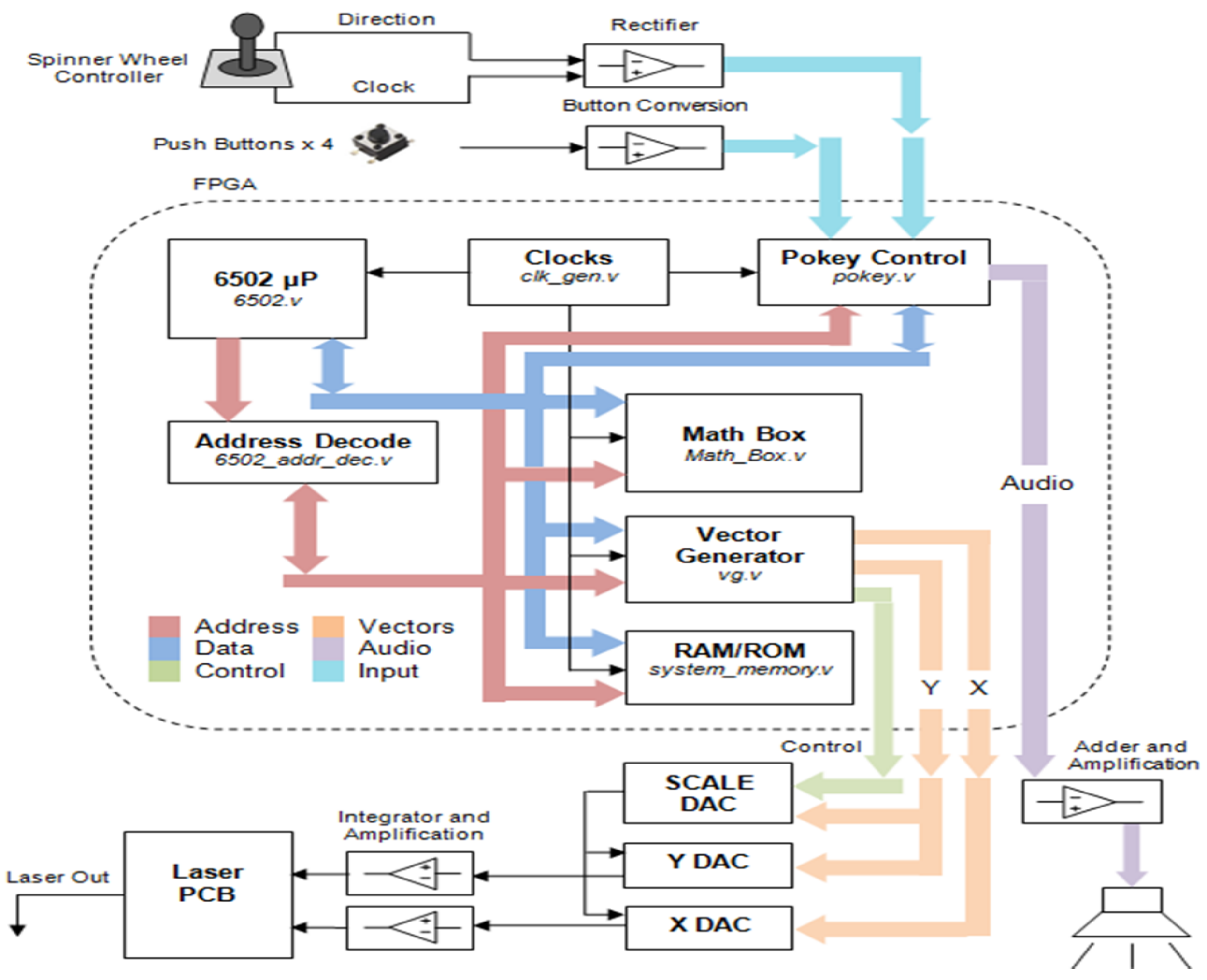

Fig. 1. High-level block diagram of the input circuitry, Verilog modules, and output circuitry.

Generator has a small ROM which contains frequently used drawings and shapes, such as letters and numbers.

C. Output Circuitry

The output circuitry consists of three DACs, several op-amps, and two integrators. Two DACs handle the 2's complement X and Y digital output from the FPGA. The third DAC is responsible for scaling. Based on a pre-determined digital scaling value, the output of the scaling DAC controls the value of the reference voltage for the X and Y DACs. Once the digital values have been converted to their respective analog voltages, they are tied to an X and Y integrator. The integrator keeps track of the previous X-Y positions and integrates the new value representing a change in said positions. A digitally controlled switch is installed across the capacitor to occasionally clear the voltage stored and reset the position to the middle of the display. The output of the integrators is fed into the laser galvanometer scanner which produces a laser display based on the changing X-Y values.

III. 6502 MICROPROCESSOR

The 6502 Microprocessor controlled all of the operations for *Tempest*. It was developed in 1975 for MOS Technology and was most notably used in *Apple I* and *Apple II* computers [1]. The 6502 is an 8-bit microprocessor that supports 56 instructions. There are nine different addressing modes that are supported, such as immediate, absolute, and relative. The 6502 has three main buses: Bi-Directional Data bus, Address-High bus, and Address-Low bus. In addition to these three busses, the 6502 also supports the following six registers:

(1) A – 8-bit accumulator register
(2) X and Y – 8-bit index registers
(3) SR – 8-bit process status register
(4) SP – 8-bit stack pointer
(5) PC – 16-bit program counter split into two 8-bit register (PCL and PCH)

A. Clock Signals

The 6502 operates using a dual-phase clocking system with clock signals Φ_1 and Φ_2. The Φ_1 and Φ_2 clock signals are generated in such a way that they are the inverses of each other. This allows one cycle for internal instruction execution and a complimentary signal used to access external components such as memory. When the Φ_1 clock signal is high, data is latched into the appropriate locations within the 6502. When the Φ_2 clock signal is high, the internal components carry out their respective functions. Fig. 2 illustrates the timing waveforms for the dual-phase system.

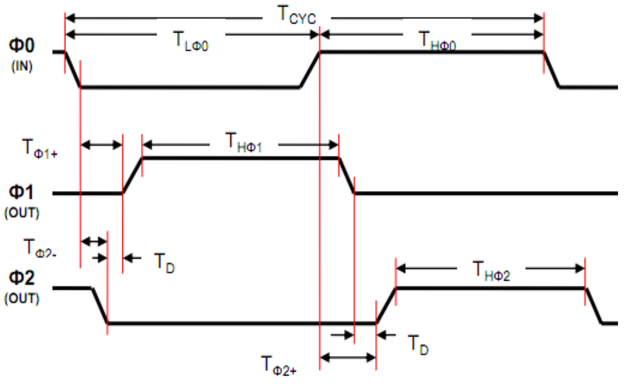

Fig. 2. Timing waveforms for the 6502 dual-phase clocks.

B. VHDL Source Code and Adaptation

The source code for the 6502 that is being utilized for this project was initially written in VHDL by Daniel Wallner and published for use on OpenCores.org. [2] This code was written in such a way that it would be compatible for the 6502, 65C02, and 65C816 chips. Wallner's code has been used in similar emulator project for the *Asteroids* by Atari©. The VHDL code has three sub-modules. The first module is T65_MCode.vhd, which handles decoding the address to perform the operation of the 56 instructions with respect to the desired addressing mode. The next module is T65_ALU.vhd, which performs all arithmetic and logical operations. It is also capable of executing Binary-Coded Decimal operations. The final module, T65.vhd, handles checking the enable and clock signals to verify that the system is really ready for operation, while also handling the execution of interrupt signals.

The 6502 VHDL code only executes op-codes when the enable and clock signal are both logic-high. In order to attain an effective 1.5MHz execution cycle necessary for *Tempest* specifications, the clock signal is given a 6MHz pulse. The enable signal will run for two clock cycles before going low, which results in the effective dual-phase clocking system at 1.5MHz. This timing waveform is shown in Fig. 3.

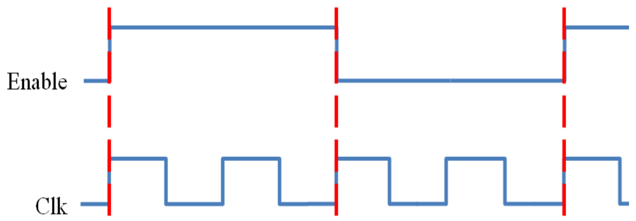

Fig. 3. Timing waveforms of the clock and enable signals.

C. Memory Map

The 6502 operates using a memory mapped Input-Output (I/O) scheme. *Tempest* had a variety of I/O devices mapped to pre-determined address locations. For example, if the 6502 needs to access RAM, it will use address locations 16'h0000-16'h07FF. If the 6502 accesses a memory location within this range of memory addresses, the result would be activating the RAM so that data may be read or stored. This memory map determines the control and access signals for the other modules such as the Math Box and Vector Generator as well.

D. Address Decoder and Access Procedures

The address decoder for the 6502 is instantiated as a separate module that will utilize a series of case statements to enable a set of control lines for accessing various I/O devices. Based on the values of these control lines, data will then be latched to and from the 6502 between the Math Box, Vector Generator, ROM, and RAM. These four components receive the 6502 output address, directly passed to their respective modules for their own internal operations.

IV. MATH BOX

The Math Box handles all computations needed within the game. These operations include tracking the game's high-scores and performing the trigonometric calculations needed to rotate some of the game's graphics as they move throughout the screen. The 6502 interfaces the Math Box as a memory-mapped I/O device. Various addresses correspond to different Math Box operations. For example, writing to addresses 6080-609F corresponds to "Math Box Start," which will start the Math Box clock and begin performing the specified subroutine. The Math Box is given instructions from the 6502 via the External Address Bus (EAB). The top four bits of this bus are sent to the address decoder, which activates the high-score module, the POKEY module or the Math Box, depending on the desired operation. This address decoder consists mostly of 2-to-4 decoders and some basic combinational logic.

A. Basic Functionality Overview

In the Math Box, the bottom five bits from the EAB are fed to ROM A1 which contains the first operation to be performed. Since ROM A1 allows five bits for addressing, there are 32 instructions that are supported by the Math Box. This address is essentially a sub-routine call, as all Math Box functions must begin with one of these 32 functions. This initial address selects an 8-bit ROM address which is then forwarded to a program counter. This program counter is controlled by the PCEN signal and a Math Box clock signal, both of which are controlled by the EAB and the clock signal from the 6502. When the Math Box is not active, the Math Box clock and PCEN are set to logical-low, thus disabling the counter. When the counter is active, the address from ROM A1 is forwarded to six 1 KB ROMs. All six ROMs have 256 memory addresses and output a 4-bit signal. Two of the ROMs (ROMS K and L) forward their signals back to a flip-flop and are then fed to the program counter. Another ROM sets control bits such as the overflow flag or the Math Box STOP signal. A summary of this data flow is shown in Fig. 4.

Due to the fact that FPGAs provide limited space for on-chip memory, some small changes were made to the original Math Box schematic in order to reduce the amount of memory that our project would require. For example, the A1 ROM is coded as a case statement with 32 different scenarios. This reduced the amount of required ROM by 256 bits.

All of the coding and simulation for the Math Box was done using Xilinx© *ISE Project Navigator* CAD Software. While this software was critical to our project, it did cause some issues. Most notably, Xilinx does not allow instantiated FPGA memory in logical simulation. To bypass this problem, we created long case statements which contained the contents of the ROMs used. A special extension called *Coregen* was required to instantiate the ROMs for our FPGA.

B. AM2901 ALU Module

The Math Box ROMs K and L combine signals with three of the other ROMs, selecting the function of the four AM2901 Bit-slice ALUs. The AM2901 supports thousands of mathematical operations. These ALUs can handle 8-bit word lengths as operands because they are configured in parallel. Various control lines allow for a parallel configuration. These lines include the carry-out bit and the carry-in bit. The input data and output data reside on the External Data Bus (EDB). The original game schematics call for this bus to be an 8-bit bi-directional data bus. However, our implementation employs two separate buses for proper 6502 interfacing. Incoming data is written to EDB-in by the 6502 and sent to the various other modules. The output of these modules is written to EDB-out. The 6502 will then latch data from the EDB-out to the EDB-in as needed. Permissions were attained in order to use pre-existing open-source VHDL code for the AM2901. [3]

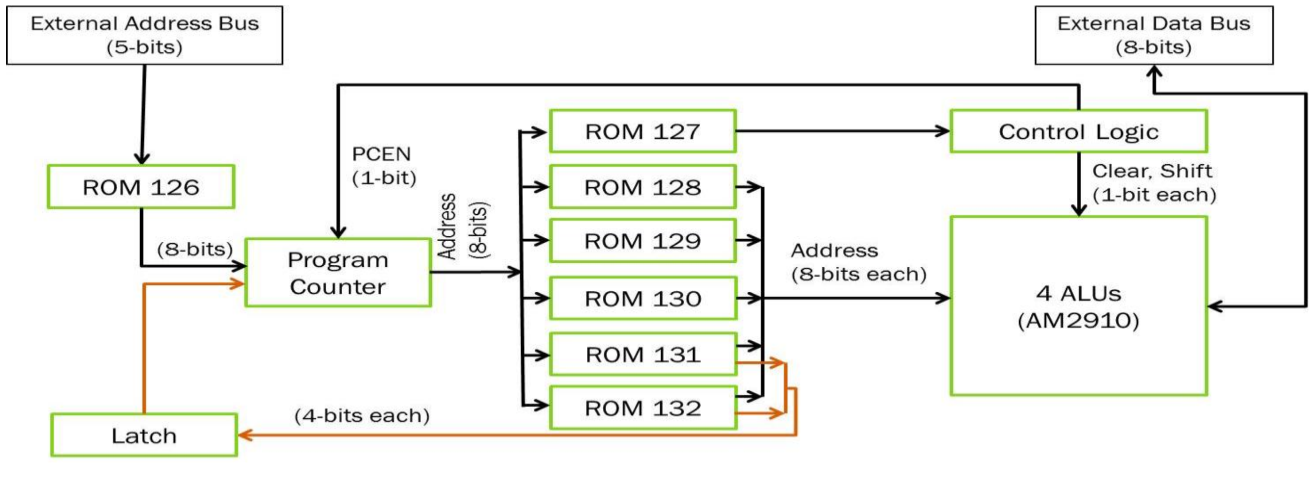

Fig. 4. High-level block diagram of the Math Box.

C. High-Score Memory

Tempest includes a basic high-score memory. This memory contains 32 memory addresses, which translates to a maximum of 32 high-scores. The High-Score Memory module is activated by the EAB and the data is read in through the EDB-in. When the scores need to be displayed, data is written to the EDB-out.

D. POKEY Module

The Atari© POKEY chip serves as the main interface between the user and the game, as well as generating audio output. Similar to the Math Box and High-Score modules, the POKEY module is activated by the EAB. The POKEY module contains two VHDL POKEY chip emulators. This is open-source and has been used in similar Atari© emulation projects. [4] One module detects when the player input buttons ("Zap" and "Fire") have been pressed. The other module processes input from the spinner wheel controller. The original game supported two sets of buttons, two spinner wheel controllers and the option to flip the image across the horizontal axis. The image flip was created to support a "Cocktail" style layout, in which opposing players sat facing each other. Only a single-player option is implemented in the design, which means the two-player capabilities were omitted. The POKEY modules read in signals provided by the spinner wheel controller and buttons. If the POKEYs are activated by the EAB, their data is written to the EDB-out bus.

V. Vector Generator

The Vector Generator controls the game's digital vector graphics output. The original Atari© *Tempest* Vector Generator was a very simple microprocessor built primarily from discrete TTL chips. The Verilog module written for this component combines a direct implementation of some of the original logic chips with a more efficient use of hardware. Many of the old TTL parts used were not used to their full capacity, e.g. a D flip-flop which contained a clear signal that was never used. Therefore, the original hardware could be optimized to only include the necessary functions for a given operation.

A. Microinstructions

The Vector Generator has the following microinstructions:

(1) VCTR – vector with a position range of ± 1024
(2) SVEC – short vector with a position range of ±16
(3) CNTR – centers the vector beam to position (0,0)
(4) HALT – halts all Vector Generator operations
(5) SCALE – changes binary and linear scaling parameters
(6) JMP – jumps to a given address
(7) JSR – stores current address and executes sub-routine
(8) RTS – returns from sub-routine

The VCTR and SVEC handle all the digital X-Y coordinate changes. The time it takes to draw the current vector is 2.73ms for a normal vector, and 21.6µs for a short vector. These timings are generated based on an internal counter which varies depending on the instruction being executed. The timing may also be altered given a binary scaling parameter. If the vector should be scaled by a factor of 2, the SCALE instruction may alter the binary scale register. This register controls the counter timing. For a larger binary scale, the timing will be decreased, producing a shorter vector without changing the X-Y digital output. Using this method, an image may start small and grow given an interval time. This was a common technique for *Tempest*, as game enemies started small and would progressively enlarge as they approached the player's ship.

The CNTR instruction allows the vector-drawing beam to be re-positioned to the center of the screen. The purpose of this instruction is clear the integrators' capacitors, as over a period of time, the noise and error in the analog system will be accumulated. Thus, CNTR controls the digital switches in parallel with the integrators' capacitors. HALT will clear these capacitors, but additionally prevents the Vector Generator from further operation until a subsequent start signal has been sent.

The Vector Generator employs a 4-word stack and 12-bit Program Counter (PC) which allows for the implementation of the JMP, JSR, and RTS instructions. These addressing instructions are used extensively by Vector Generator ROM in order to access frequently drawn shapes and characters.

B. Vector Generator ROM and RAM

The Vector Generator ROM contains preset sub-routines which were developed to make drawing frequently needed shapes more efficient. For example, the first twenty-six sub-routines in this memory are the instructions to draw the letters of the alphabet. Other sub-routines include the Atari© trademark symbol, the *Tempest* ship, and game level designs. The ROM was available as an open-source file and used in previous emulations of *Tempest*. The Vector Generator RAM stores the operating instructions of the module. At the start of operation, the Vector Generator executes from RAM, which will contain calls for ROM sub-routines as well as various instructions for scaling and centering.

The original Vector Generator used eight 10-bit address RAMs to achieve an effective 4096 KB of space. This was due to the limitation of only a 4-bit read width of the RAM component used at the time. Rather than implementing this inefficient design, the RAM module used in the project's Vector Generator combines all of the space into a single RAM component. This allows for a much cleaner access of data within memory.

C. 6502 Interfacing

The Vector Generator and the 6502 Microprocessor are able to communicate. All of the instructions written to Vector Generator RAM will come from the 6502. According to original design, the 6502 may only write to the Vector Generator during one half-cycle of a 1.5 MHz time period. This was due to the way the original game worked using the dual-phase clocking of the 6502 Microprocessor. Since the open-source 6502 VHDL does not use a dual-phase system, some adaptation was necessary in order to achieve the same functionality as the original game. When the 6502 wants to access the Vector Generator, a control VMEM will activate. This signal is fed back to the 6502, causing the 6502 to execute the break sub-routine. This will effectively postpone the 6502 from executing its next actual instruction for five cycles. During this time interval, the enable clock for the 6502 module will become low, literally freezing its operation until is once again high. Since this enable clock is tied to a 1.5 MHz clock, it is also synced with the valid access interval of the Vector Generator. Therefore, the 6502 may write to the Vector Generator since VMEM is active and the enable clock is low. Normal operation will resume on the next positive edge of the enable clock for both modules.

D. Digital Output

The Vector Generator outputs the following information from the FPGA to the analog prototyping board:

(1) 10-bit 2's complement change in X coordinate
(2) 10-bit 2's complement change in Y coordinate
(3) 8-bit linear scaling register
(4) VCTR
(5) CENTER

The X and Y coordinate changes are sent to 12-bit DACs which are configured to represent values from ±1024. The 8-bit linear scale is processed by a DAC which controls the amount of reference voltage given to the X and Y DACs. VCTR is used to enable drawing time intervals. It is affected by the length of the vector as well as binary scaling. It toggles the linear scaling DAC, which consequently toggles the reference voltage for the X and Y DACs. CENTER controls the digital switch across the integrators' capacitors. When it is active, it closes a line in parallel with each capacitor, draining the voltage stored and effectively centering the beam to (0,0).

VI. Analog components

The analog components for the project are made up of input and output circuitry. The input components are comprised of a spinner wheel controller and four arcade-style push buttons. The output components consist of three Digital-to-Analog Converters (DACs), two integrator op-amps, and several amplifying op-amps.

A. Input Circuitry

The first part of the input is the spinner wheel controller. This controller allows the player to navigate through the game interface. The controller chosen is the *Turbo Twist 2* from GroovyGameGear.com. [5] This particular controller uses an interface board that is powered by ± 5 VDC supplied by the FPGA board. It outputs a data and clock line which oscillates between 0 and 4 VDC. The FPGA input voltage is 3.3 VDC, so a 12 kΩ pull-down resistor performs a voltage division to drop the voltage to a useable level. The direction of motion is determined by the timing of the data and clock lines. For instance, if the if the controller knob is spun in the counter-clockwise direction, the data line is logic-high when clock is logic-high. The frequency of the data and clock lines varies between approximately 200 and 1000 Hz, based on the rate at which the spinner is rotated. At maximum frequency, the player's game position changes every 1 ms, which is too fast given that the game only has sixteen positions on any level. In order to actuate some delay of the input, a digital 11-bit counter is employed. The counter will increase every positive edge of the clock, but the position will not change until a certain bit of the counter has toggled. The lower 7 bits must be filled to increment the player's position by one. After implementing this method, the player changes position at maximum frequency every 0.128 s.

Four push buttons represent the following player input signals: *Fire, Zap, Coin-In, and Start.* Each button is connected to 3.3 VDC supplied by the FPGA. The buttons operate by using a three-terminal micro-switch. Once a button is pressed, the normally-open contact is closed which ties logical-high voltage to the FPGA.

B. Output Circuitry

The output circuitry requires three multiplying DACS: one DAC for the scaling of the object to be drawn and the other two for plotting the X and Y coordinates. There are several op-amps used throughout the circuit. The TL082 op-amp is used for all cases due to its high speed and low current requirement. [6] It is biased with ±15 VDC. An ADG201AKN digital switch is needed for use with the integrator op-amps. The ADG201AKN has a high switching speed, low current requirement, and low leakage current. [7] It used ± 15 VDC. The DAC08 is used as the scaling DAC in this design as it there are 8-bits of digital scaling input and has a low current requirement. [8] The DAC08 is powered with ±15 VDC. Based on the digital inputs from the FPGA, the scaling reference voltage can be incremented from 0 to 10 VDC determined by the 8-bit digital scaling input. When the scaling is increased, the amount of current that leaves the IO pin of the DAC08 increases. This current is sent through an op-amp with gain of 10 and is used as the reference voltage for the X-Y coordinate DACs.

The XY coordinate DACs are the LTC7541A. It has a low current requirement and a 12-bit digital input. [9] This DAC is powered by +15 VDC and takes 11 digital signals (LSB is grounded) from the FPGA output. The 11 digital inputs allow the LTC chip to output a digital swing from ±1024 positions which equate to ± 10 VDC. The output of the LTC7541A feeds into an integrator op-amp as shown in Fig. 5.

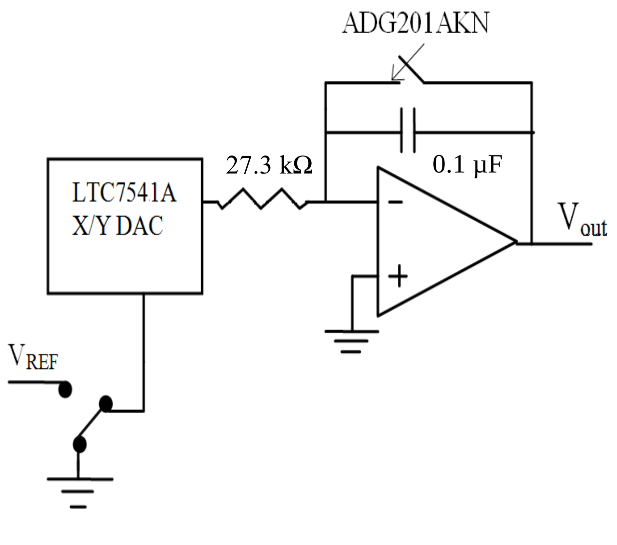

Fig. 5. Simplified Multiplying DAC and Integrator Op-Amp.

The circuits for the X and Y analog processing are identical. The integrator op-amp tracks the previous voltage compared to the voltage coming in and draws a straight-lined vector from its previous position to the new position specified. The new position is specified from the output of the LTC DAC. The integration timing chosen for the integrator is important because it determines the speed at which the lines are drawn. This timing is controlled by the RC value of the integrator. For this design, the RC value is set by (1).

$$\mathbf{2.73\ ms \approx 27.3k\Omega \cdot 0.1\ \mu F} \qquad (1)$$

This timing is based on the original timing parameters of the *Tempest* vector generator. The capacitor bridging from the $-V_{IN}$ and V_{OUT} pins of the integrator op-amp will store previous voltage values and periodically needs to be discharged. This is due to the gradual accumulation of noise and error which will lead to instability of vector images. This design uses the ADG201AKN digital switch which is controlled by the FPGA in order to discharge the capacitor. The output of the integrator op-amp is sent to an inverting op-amp to counteract the natural inversion of the integrator. The output of the inverting op-amp is the final X and Y voltage. It swings between ±10 VDC. Plotting these voltages versus each other will produce a vector-graphics image.

C. Power System

All silicon chips used in this design are biased with ± 15 VDC, with the exception of the LTC7541A. It only requires +15 VDC and ground. The ±15 VDC is supplied by a voltage regulator. The regulator used for this design is the IA0515D. It requires +5 VDC as power and generates a ± 15 VDC output. The regulator uses 1 W of power and outputs ±33 mA of current to the positive and negative voltage terminals, respectively. [10] The entire analog circuit draws approximately 26 mA of current. All of the silicon chips have a 0.1 µF decoupling capacitor tied to ground on their power terminals. Decoupling capacitors are necessary for maintaining signal stability.

D. PCB Design

All parts in the PCB design are through-hole components in order to simplify the testing process. A standard 4-layer, 5in. by 5in. PCB is large enough to create the entire circuit. The spinner wheel controller data/clock lines and four buttons have their inputs hardwired to the PCB. The FPGA board is mounted on top of the PCB via through-hole headers.

VII. Conclusion

Vic Vector is a game emulator that preserves the original hardware characteristics of the classic arcade game *Tempest* by Atari©. The digital hardware is modeled in Verilog HDL and programmed using an FPGA. The 6502 Microprocessor interfaces with the other modules and runs the game ROM. It is responsible for retrieving data from the Math Box and writing drawing instructions to the Vector Generator. The Vector Generator outputs digital vector data to the analog circuitry. The analog hardware has been updated and adapted to interface the FPGA and a laser galvanometer scanner used to display the game's vector graphics.

Acknowledgement

Our group wishes to recognize and thank Dr. Samuel Richie, Dr. Thomas Wu, Dr. Mingjie Lin, and Mr. Don Harper for their assistance and support throughout our project.

References

[1] "6502 Technology." *6502 Technology*. Web. 19 July 2011. <http://www.6502.buss.hk/6502>.

[2] "T65 CPU :: Overview :: OpenCores." *Home :: OpenCores*. Web. 19 July 2011. <http://opencores.org/project,t65>.

[3] "FPGA ARCADE - Asteroids Main Page." *FPGA ARCADE - Main Page*. MikeJ, 2003. Web. 22 July 2011. <http://www.fpgaarcade.com/ast_main.htm>.

[4] Nasr, Amr. "Microprocessor AM2901 4 Bit Microprocessor Slice." VHDL and Verilog Designer. 22 Dec. 2010. Web. 22 July 2011. <http://vhdldesign.blogspot.com/>.

[5] TurboTwist 2™ Arcade Spinner Control Product Information." *GroovyGameGear.com*. Web. 20 Apr. 2011. <http://groovygamegear.com/webstore/index.php?main_page=product_info&%20products_id=268&zenid=c2619c1529e3cb46c86264b212b42257>.

[6] "Wide Band Dual JFET Input Operational Amplifier" National.com. Web. 21 July 2011 <http://www.national.com/mpf/TL/TL082.html#Overview>

[7] "ADG201A: 60 Ohm, Quad SPST Switch" Analog.com. Web. 21 July 2011. <http://www.analog.com/en/other-products/militaryaerospace/adg201a/products/product.htm>

[8] "DAC08: 8-Bit, High Speed, Multiplying D/A Converter" Analog.com. Web. 21 July 2011. <http://www.analog.com/en/digital-to-analog-converters/da-converters/dac08/products/product.html>

[9] "Linear Technology Improved Industry Standard CMOS 12-Bit Multiplying DAC." Linear.com. Web. 21 July 2011. <http://www.linear.com/product/ltc7541a>

[10] "IA Series" Xppower.com. Web. 21 July 2011 <http://www.xppower.com/orderPriceList2.php?seriesid=100085&groupid=100059&catuid=2&lang=EN>

Group Members

Robert Baker is graduating from the University of Central Florida with a BS in Electrical Engineering. Upon graduation, he intends to pursue a career in electrical utility protection. In the future, he wishes to become a licensed Professional Engineer.

Tony Camarano is graduating from the University of Central Florida with a BS in Electrical Engineering. Tony plans to continue his education, pursuing a PhD in Electrical Engineering and studying under Dr. Thomas Wu at the University of Central Florida. He intends to focus on research in Modern Electric Machinery.

Drew Hanson is graduating from the University of Central Florida with a BS in Computer Engineering. He is currently a participant in the UCF/Lockheed Martin College Work Experience Program. Upon graduation, he intends to pursue full-time employment in the defense industry or another systems engineering field.

Robert Higginbotham is graduating from the University of Central Florida with a BS in Computer Engineering and Minors in Mathematics and Computer Science. Upon graduation, Robert intends to begin full-time employment in systems engineering, while also pursing his MBA.