EEL 4924 Senior Design, Spring 2012
Final Report
April 25, 2012

FPGA NES

Submitted by:
Art King
art.king@ufl.edu
352-263-1403

Table of Contents

TABLE OF FIGURES

ABSTRACT

The goal of this project is to design a stand-alone emulator of the Nintendo Entertainment System video gaming platform whose major architecture is based primarily around a modern programmable logic device in conjunction with additional external peripheral components. The design will include a FPGA core developed in VHDL, video and audio output, and human interface controller. A simple pong demo is written to demonstrate basic system functionality.

BACKGROUND

Emulation of the Nintendo Entertainment System is a popular pastime amongst hobbyists and hackers alike. However, the overwhelming bulk of community effort has been placed on high-level software approaches—without a doubt driven by the relative accessibility of software development tools vs. a hardware approach—relying on observations made from community reverse engineering efforts to incorporate "soft hacks" that replicate the many nuances and quirks of the original system, let alone basic functionality. Furthermore, the multiple layers of abstraction that software emulators must incorporate introduces real-time errors that are difficult to compensate for without direct register transfer level manipulation. The approach this project will take is to use existing reverse engineered documentation available throughout the web to synthesize a low-level emulator on a FPGA and to integrate all the necessary peripherals needed to support such a design.

PRIOR ART

As of writing, there are several known projects that have successfully emulated the NES using a FPGA as its core. Of these, three projects are noted to have given a fair general top-level description and are worth noting:

- Kevin Horton – developed on a custom 2-layer Altera Cyclone I platform [1].
- Dan Leach – developed in VHDL on a Altera UP3 development board as part of a Master's project at Bradley University [2].
- Dan Strother and Brent Allen – developed in VHDL as undergraduates at Washington State University; Dan has since expanded the project with numerous enhancements on a Digilent Nexys development board based on a Xilinx Spartan-3E and has ported a large section of his original code to Verilog [3].

This project will differ from the above known attempts in that 1) a modular prototype will be consolidated into a single multi-layer design whilst facilitating future expansion and development, and 2) a modern 60-nm technology node FPGA will serve as the system's programmable core, and 3) a Nintendo Wii controller will be integrated into the design as the primary user interface.

TECHNICAL OBJECTIVES

The intent of this project is to encapsulate multiple facets of electrical engineering in a single design, emphasizing techniques in FPGA development, PCB design for high-speed signal integrity, microcontroller integration, and analog design. Figure 1 depicts a basic system-level block diagram of interaction between major components. Since a modular design has already been prototyped and tested, the next goal will be to attend to previously unforeseen limitations and consolidating the segmented design into a single multi-layer PCB encased in a proper enclosure, with the only external devices needed being a standard AC/DC power adapter, game controllers, external display and speakers.

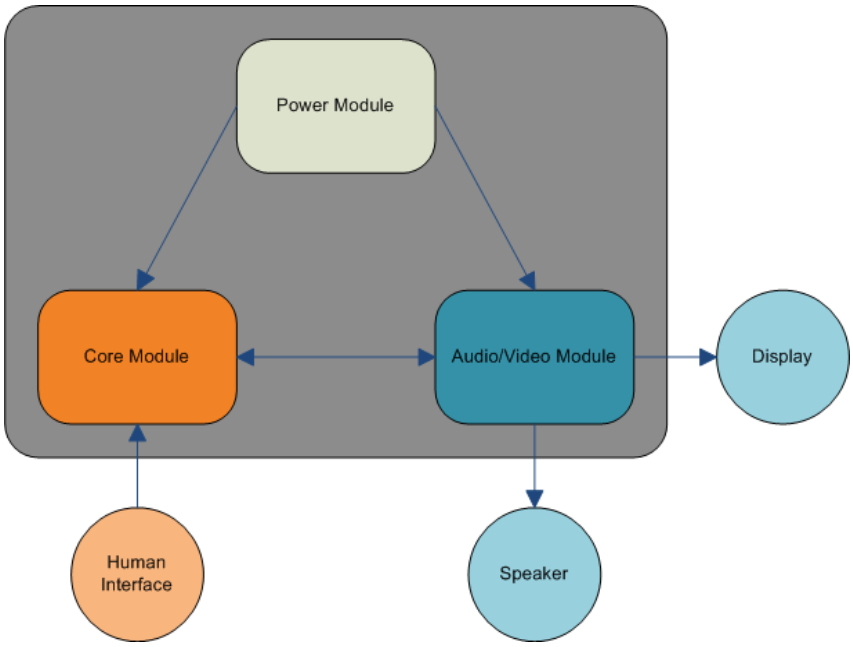


Figure 1 Abbreviated FPGA NES system-level block diagram.

CORE FUNCTIONAL DESCRIPTION

The Nintendo Entertainment System functions around two main chips: a Ricoh 2A03 and Ricoh 2C02 [5], as shown in Figure 2. The 2A03 can be thought of as the "brains" of the system, an ASIC containing a MOS Technology 6502 processing core with its binary coded decimal arithmetic functions disabled, and a custom pseudo-Audio Processing Unit integrated as a memory-mapped peripheral subsystem to the core. The 6502 is responsible for processing data stored in an external cartridge, interpreting controller input, and sending image data to the 2C02; the pAPU is responsible for producing five analog audio channels—two pulse waves, one

triangle wave, one white noise, and one differential PCM) which are mixed to produce all game sound.

The 2C02 is a custom chip developed by Ricoh dedicated to processing image data received from the 6502 and produces 256×240 pixel resolution video in NTSC format; the actual displayed image is slightly reduced as a byproduct of NTSC formatting, which eliminates the upper and lower 8 scanlines for an effective visible area of 256×224 pixels. Due to the limited address mapping capability of the 6502, most cartridges were equipped with on-board memory map controllers for bank switching in the case where the game being played required more than the 32KB of program ROM that the 6502 could map to at any given time.

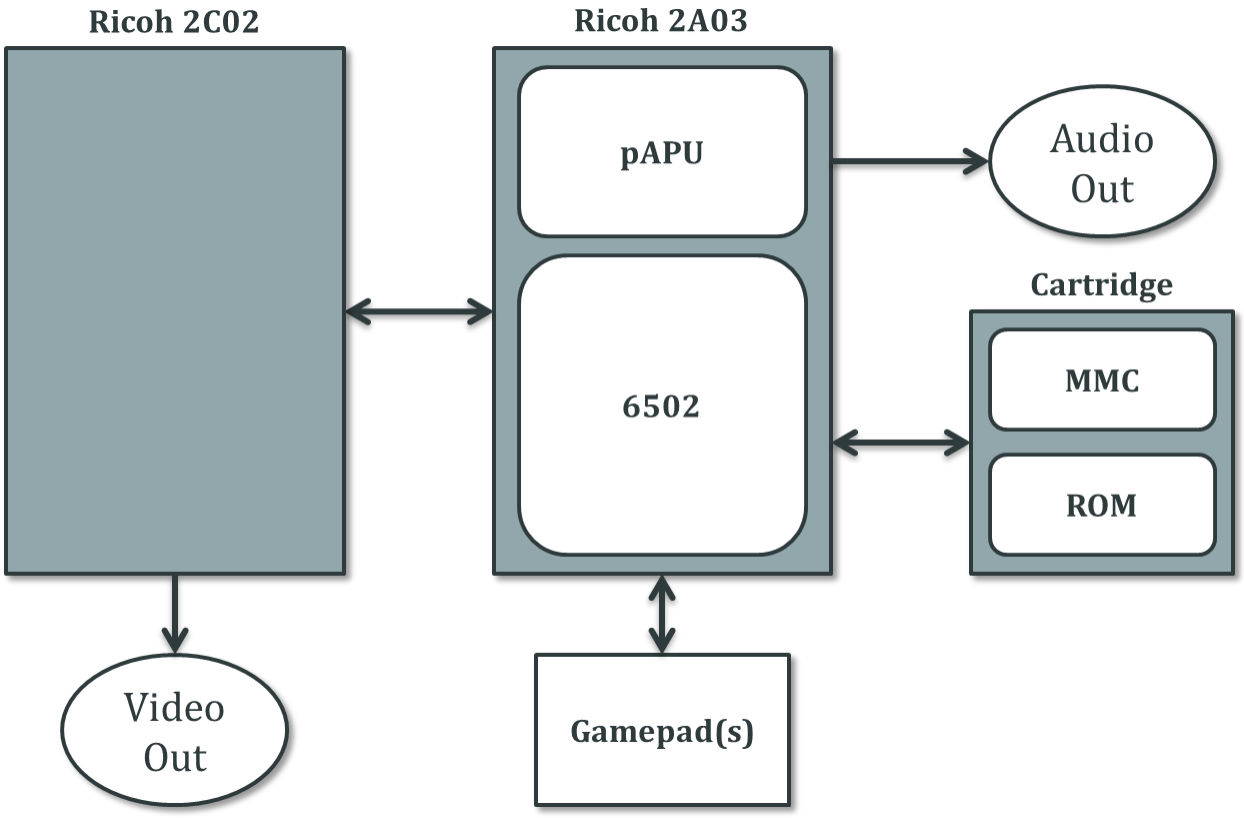


Figure 2 Nintendo Entertainment System block architecture.

The Nintendo Wii is a modern gaming system and immediate successor of the GameCube. Of particular interest is not the system itself, but the means in which humans interact with the system—the Wii controller, informally known as the wiimote. This controller is unique in that a conjunction of wireless technology and a host of on-board sensors are integrated to invoke a user experience that is unlike any other commercial implementation. Typically, manufacturers encode wireless signals in such a way that inhibits control and direct access to the raw data stream to anything other than the original system that it is intended to function with.

However, in the case of the wiimote, no such encoding is implemented, allowing a motivated hacker the opportunity to incorporate this unique, mass-produced, and relatively inexpensive piece of hardware into his/her design.

TECHNOLOGY SELECTION

The highlighted blocks shown in Figure 3 will compose the main core of the system. An Altera Cyclone IV FPGA was chosen to implement the bulk of the architecture. Despite being comparatively more expensive than an equivalent or better offering from Xilinx (in particular, the Spartan-6 class), the Cyclone IV was chosen due to availability of development tools. Since the FPGA is SRAM-based, and due to the required internal block RAM needed to synthesize the architecture, a 4Mbit configuration device will be integrated into the design; this enables both in-circuit serial programming of the configuration device for automatic power-on configuration and direct JTAG programming of the FPGA to speed up development. It is the author's intent to eventually migrate the completed architecture to reap the benefits of a Xilinx Spartan-6, however, that endeavor is beyond the scope of this project's objective. Another decision driving the selection of this particular FPGA is supply availability of an acceptable prototype package: a 144-pin QFP. Despite BGA packages being more prevalent and often cheaper, it was neither economically feasible nor time permitting to incorporate such a package into a prototype design due to the necessity of test equipment necessary to both mount the device and inspect for potential short circuits. Furthermore, the fine ball pitch of such devices would require trace width and via tolerances that bordered the minimum requirements of the chosen board manufacturing house.

Bluetooth module selection was directly driven by proprietary firmware capability; all other factors, including cost, were secondary at best and did not directly contribute to the success of integration. Several modules were tested for compatibility and it was ultimately determined that the Bluegiga WT12 was an ideal candidate for several reasons to be discussed. This device is a surface mount integrated RF module whose components are hidden beneath a ground shield with the exception of a built-in ceramic chip antenna.

Although multiple microcontrollers were tested, the ultimate factor which drove selection was package and pin layout. A TI MSP430 in SOP38 package was tested successfully and was able to perform the relatively simple tasks that it was assigned to do, however, it was anticipated that the pinout of the device would complicate routing due to the inflexibility of multiplexed pins and asymmetrical programming pin layout. A QFP Atmel AVR was ultimately selected to perform peripheral microcontroller duties for several reasons: 1) it allowed the writer to gain experience in developing a different device, and 2) price was significantly lower, and 3) Atmel

chips tend to map their device pins in such a way that is symmetrically appealing for board layout and prototyping.

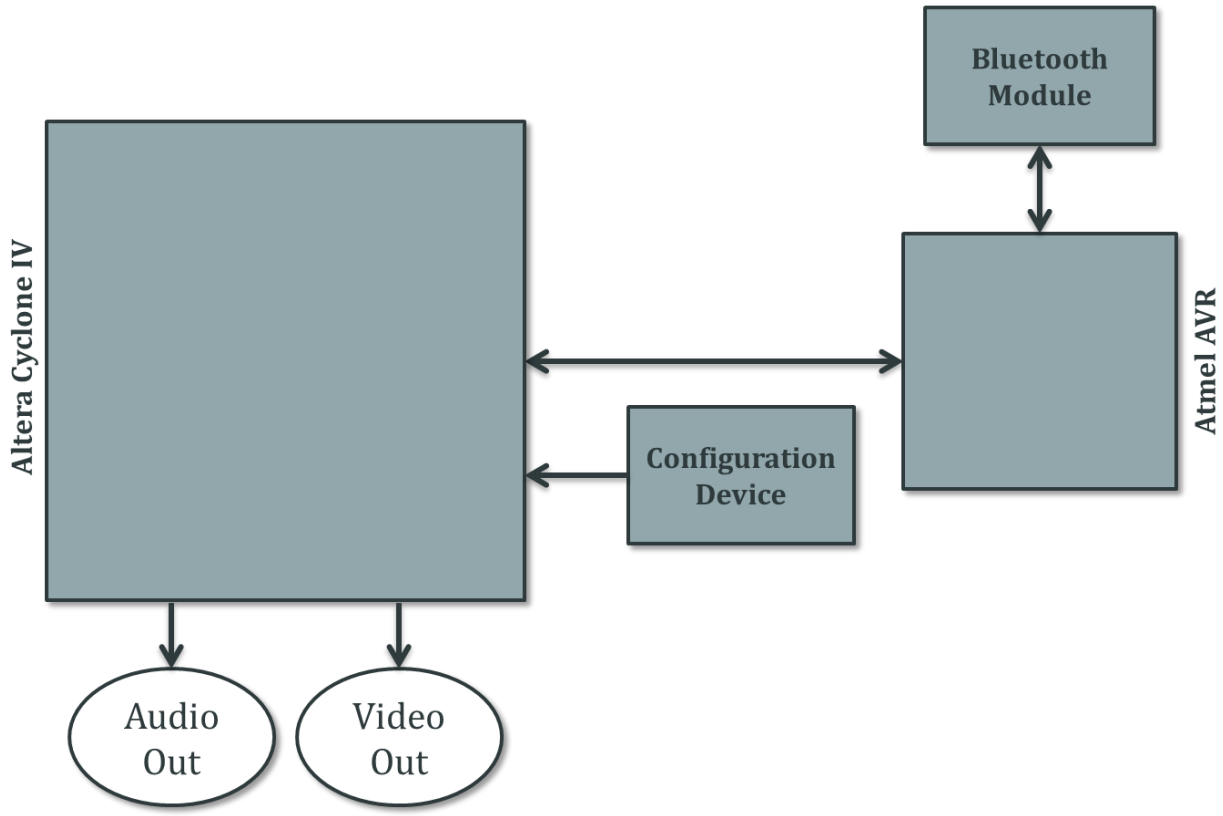


Figure 3 System core and audio/video block architecture.

For producing audio, a solution was attempted in which "jellybean" op amps and discrete filtering would be employed. However, it was later determined that a much cheaper solution can be had with lower component count while still achieving acceptable sound quality. A National LM4876 was chosen as the main device for audio amplification. This device was selected due to its purpose-designed topology intended for audio amplification, single-supply operation, stability at unity gain, and not requiring an output coupling capacitor to filtering out the DC component (which could in itself cost as much—if not more—than the amplifier itself). Since the original Nintendo did not have such stringent audio requirements and produced simple "synthetic" sounds, this was an ideal choice. To insure that clipping does not occur, a trimming potentiometer is also integrated for manual adjustment. Audio output is wired to a standard 3.5mm TRS jack for compatibility with off-the-shelve desktop stereo speakers that are typically used with computers.

The primary means of generating video was through standard VGA. Although a chip solution could have been pursued, this would have added unnecessary complexity and cost to the

prototype. Thus, it was easily settled that a R-2R DAC using 1% precision resistors would be implemented. Although multiple resistors would be necessary to implement this design, this nevertheless simplifies layout and reduces component count since only two carefully selected values are needed. This also gave the writer the flexibility of simultaneous impedance matching to ensure that voltage swing remains below the prescribed level.

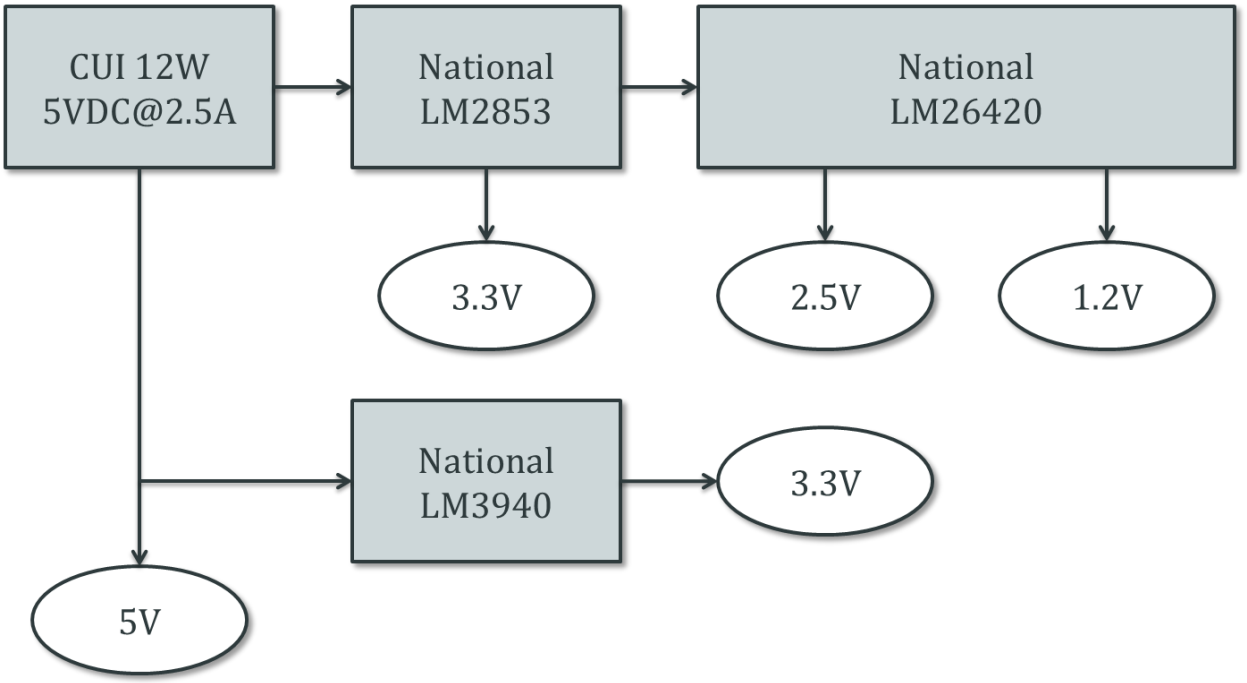


Figure 4 Power regulation block architecture.

The power section shown in Figure 4 will consist of the appropriate step-down and regulation circuitry needed to power all devices within the system. This segment of design is critical in that it must be capable of sourcing all voltage levels necessary for the main FPGA and all peripheral components. A 5V, 2.5A wall switching supply will provide main power to be stepped down and regulated. Although it not expected that the system will require more than 1.5A at any given time for this design, a 2.5A main source was chosen to allow for future expansion as necessary. The first stage regulator is a National LM2853 Buck regulator which will step down 5V to 3.3V for use by the FPGA I/O banks. This Buck regulator is capable of sourcing as much as 3A and will be sufficient to handle future peripheral expansion. The 3.3V output from the LM2853 will then be sent to a National LM26420 dual Buck regulator. The first channel will step down 3.3V to 1.2V for FPGA core switching. The second channel will step down 3.3V to 2.5V explicitly for the FPGA's analog PLL rail, which must be powered regardless of its use. Each channel is capable of providing 2A of current, with the higher load expected to come from the 1.2V rail. Both regulators switch at a nominal frequency of 550kHz and incorporate synchronous rectification, providing a reasonable compromise between reducing

board real estate and maximizing efficiency. Due to the inherent sensitivity of switching regulators, a high level of attention to detail must be placed on PCB to ensure clean power and to minimize the effects of electromagnetic interference, in particular, from the digital devices on the core section of the system. It will also be necessary to filter rails which will provide power for sensitive segments, in particular, the 2.5V analog PLL rail used by the FPGA. Furthermore, a National LM3940 LDO regulator is also integrated to better isolate heavy FPGA I/O switching from the microcontroller and Bluetooth module.

HARDWARE

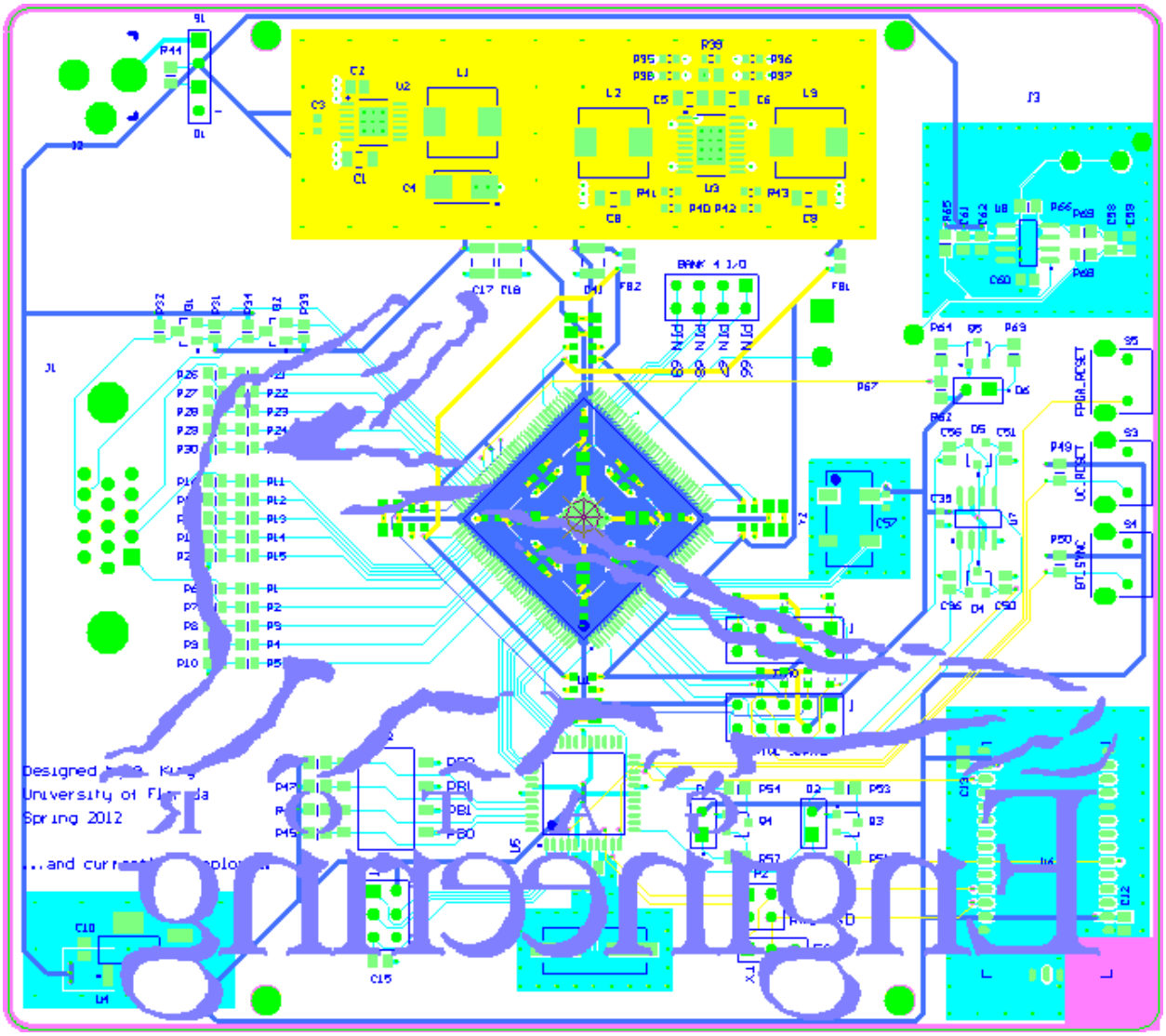


Figure 5 Consolidated PCB layer masks.

Figure 5 depicts the final PCB layer masks of the hardware design. The board cutout itself is intended to be a drop-in for a Serpac A31 enclosure whose front and rear panels are modified to accommodate protruding components for interfacing with external devices. Since the original layout was prototyped on three separate PCBs, it was anticipated that component density would be greater (although manageable). This had the potential to cause serious noise coupling issues, so care was taken to the best of the writer's knowledge to reduce this possibility by incorporating ground nodes around each sensitive device, which was then liberally stitched to an inner ground plane. Prior to having the multi-layer board manufactured, an attempt at basic 3D modeling was performed to gauge component placement and pre-emptively visualize the final product, as shown in Figure 6 below. 3D modeling was an important aspect of the design process as it enabled to writer to visual inspect component clearances prior to manufacturing, which reduced the possibility of have to spin multiple iterations for minor discrepancies that can be easily corrected. A few components were not included in this model, however, sufficient room was allocated for these components.

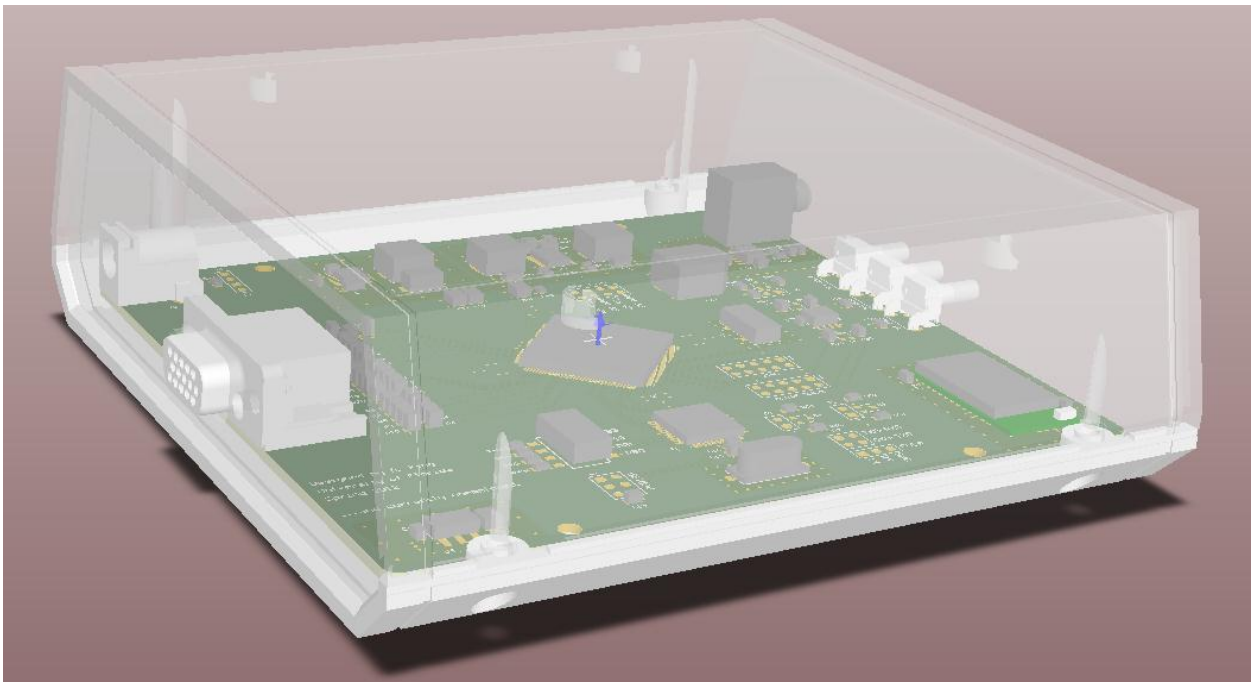


Figure 6 3D representation of PCB design with most components modeled.

Figure 7 depicts the final assembled product with the enclosure top removed to expose the PCB. Despite being largely successful on first attempt, a few correctable errors were encountered. The first was that the SOT-23 transistor used for driving LEDs was incorrectly mapped to its footprint. This was corrected by removing the surface mount component and temporarily replacing them with an equivalent TO-92 device. The second error was discovered in the audio amplification section. The ring of the 3.5mm TRS jack, which was supposed to be

wired to the virtual ground signal generated by the op amp, was accidently wired to proper signal ground in error. To correct this issue, the net was electrically disconnected from ground and hand wired from op amp to jack as can be seen in the top right corner of Figure 7.

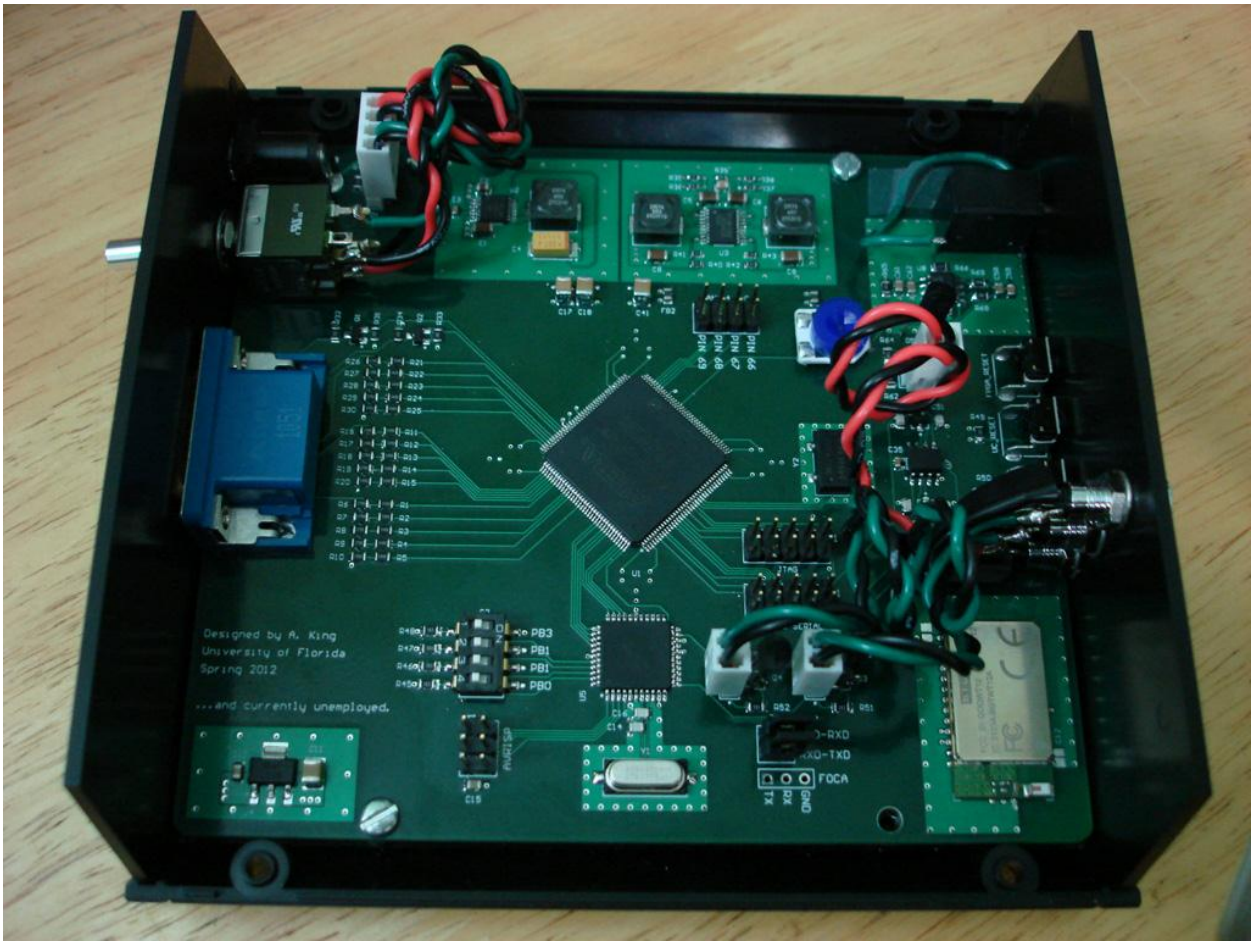


Figure 7 Assembled hardware with top of enclosure removed to expose PCB.

SOFTWARE

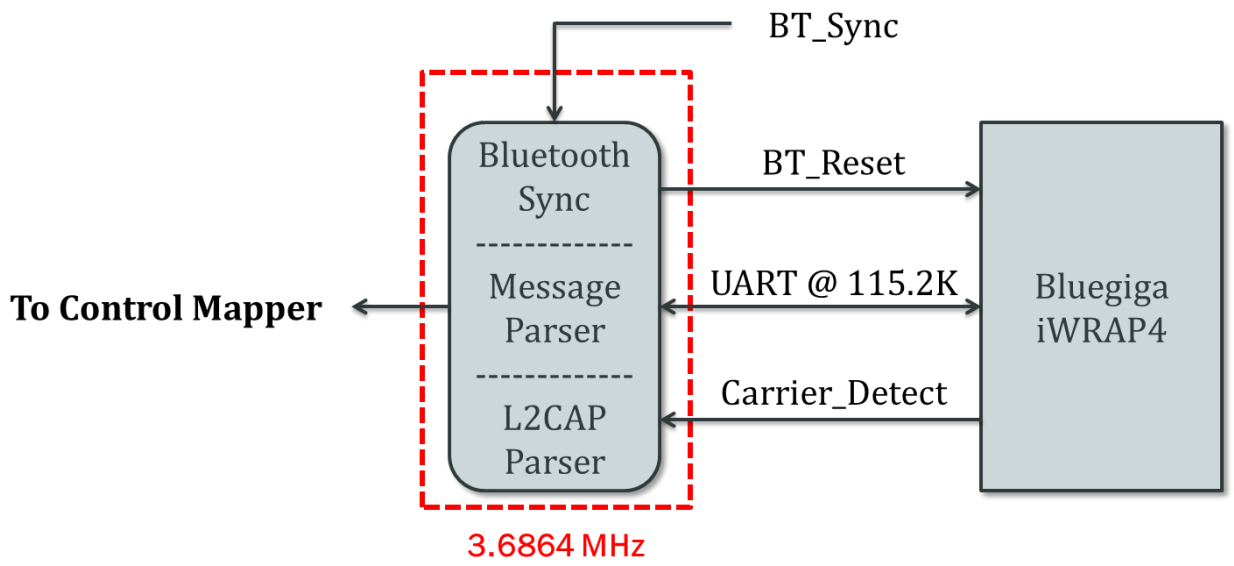


Figure 8 Block diagram of communication between microcontroller and Bluetooth module.

Figure 8 depicts the general wiring and software flow of communication between the microcontroller and Bluetooth module. Upon power up, the microcontroller initiates all registers necessary before communication can begin, to include establishing an appropriate UART baud rate and enabling externally driven interrupts. The Bluetooth module is then held in the reset state to insure that when synchronization is requested by the user, the device will be in a known state. The microcontroller waits for the Bluetooth sync input to be triggered, which executes the INT0 interrupt service routine. After debouncing and button release, the Bluetooth module is released from the reset state and its boot sequence initializes. When the boot sequence is complete—known by the microcontroller by parsing of the boot message through UART—the microcontroller attempts to connect to the MAC of a known wiimote within the area by establishing a L2CAP connection on PSM 13. Since the wiimote communicates via the HID protocol in an atypical manner, it was necessary to bypass the Bluetooth abstraction and connect directly to PSM 13, which happens to be the standard HID interrupt channel. It's worth noting that the ability to access the L2CAP layer is the primary driving factor for the success of WT12. Without this capability written into the module's proprietary firmware, the only other option for establishing a connection would be via HCI, which is not only significantly complex in terms of stack implementation but resource demanding; HCI is typically how wiimote signal access is performed by the hacker community since a proper computer has far more resources at its disposal than the average custom embedded platform. Continuing, if the Bluetooth module is unable to establish a connection with the requested MAC, the microcontroller's parsing routine

will detect this and exit the ISR while alerting the user of the event through the front panel-mounted LEDs. If, however, a successful link has been established, the microcontroller will them command the associated wiimote to hold the P1 LED (as opposed to all LEDs flashing as can be seen during the sync process) and enables the UART receive interrupt for active parsing of all data packages.

Whenever a button on the wiimote is either pressed or released, a formatted data package is sent to the Bluetooth module, which in turn is relayed to the microcontroller for parsing. The package 4 bytes of the sequence

```
0xA1 0x30 0xXX 0xYY
```

where the first two bytes contain the mode and channel respectively and remain the same for all packages, while the last two bytes contain encoded button information. The microcontroller receives this package sequence, parses it, and transmits the decoded sequence to the FPGA for further processing. Since the microcontroller and FPGA operate on independent clock domains, a mux recirculation synchronization scheme is employed; this was chosen over other methods since the clock domains of both devices are static and well known.

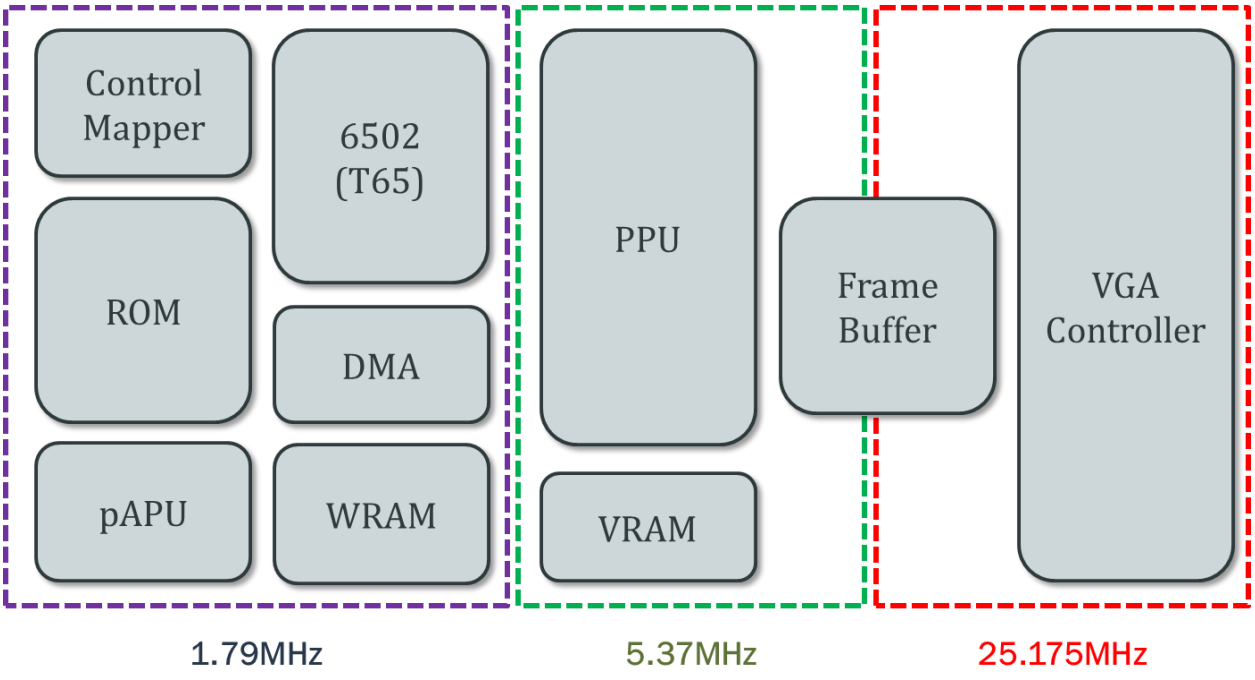


Figure 9 Simplified block representation of synthesized subsystems.

A modified version of the T65 open source project available on OpenCores.org [6] was used as a synthesizable basis for the required 6502 processor. Of the numerous synthesizable

6502 cores available across the internet, the T65 was chosen based on forum feedback as being the most reliable open-source VHDL implementation. Despite praise from the online community, the implementation is in fact ridden with a handful of relatively minor bugs and features that are unnecessary for system realization; in the case of binary coded arithmetic functions, undesirable altogether. It was therefore necessary to isolate and correct these bugs, and remove unwanted functions prior to system integration in order to achieve the closest emulation possible. In addition to the T65 core, a custom PPU and pAPU will need to be developed, along with at least one of the more widely used Memory Map Controllers. A script was written to convert a *.nes ROM file to *.mif file which can be loaded into internal block RAM instantiations prior to synthesis. This script must be slightly modified to account for various ROM sizes and mappings prior to execution.

It's worth noting that the FPGA is clocked by a single 25.175 MHz oscillator, which in turn is sent through an instantiated PLL to acquire the 1.79 MHz and 5.37 MHz clock signals as shown in Figure 9. This was done in an attempt to accurately emulate timing of the original system and posed a significant challenge in itself. The frame buffer is an instantiated dual-port RAM which is written to by the PPU in the 5.37 MHz clock domain and read out by the VGA controller as each frame is generated. Since the NES has an effective palette of 64 colors, 56 of which are unique, each address of the frame buffer was reduced to 6 bits wide, which was subsequently mapped to control the 15 bits required for RGB output. Furthermore, since the upper and lower eight scanlines are not actually displayed on screen, address space was further reduced to 56K, which utilizes less than 60% of total internal BRAM resources, freeing up the rest of the BRAM for other subsystems, including work RAM for the CPU, video RAM for the PPU, and storing the cartridge ROM image. For the sake of demo, however, a simple pong demo was written to demonstrate system functionality in general.

REFERENCES

[1] K. Horton. "FPGA Console of DOOOOOM!"
Internet: kevtris.org/Projects/console/sections/, [Aug. 25, 2011].

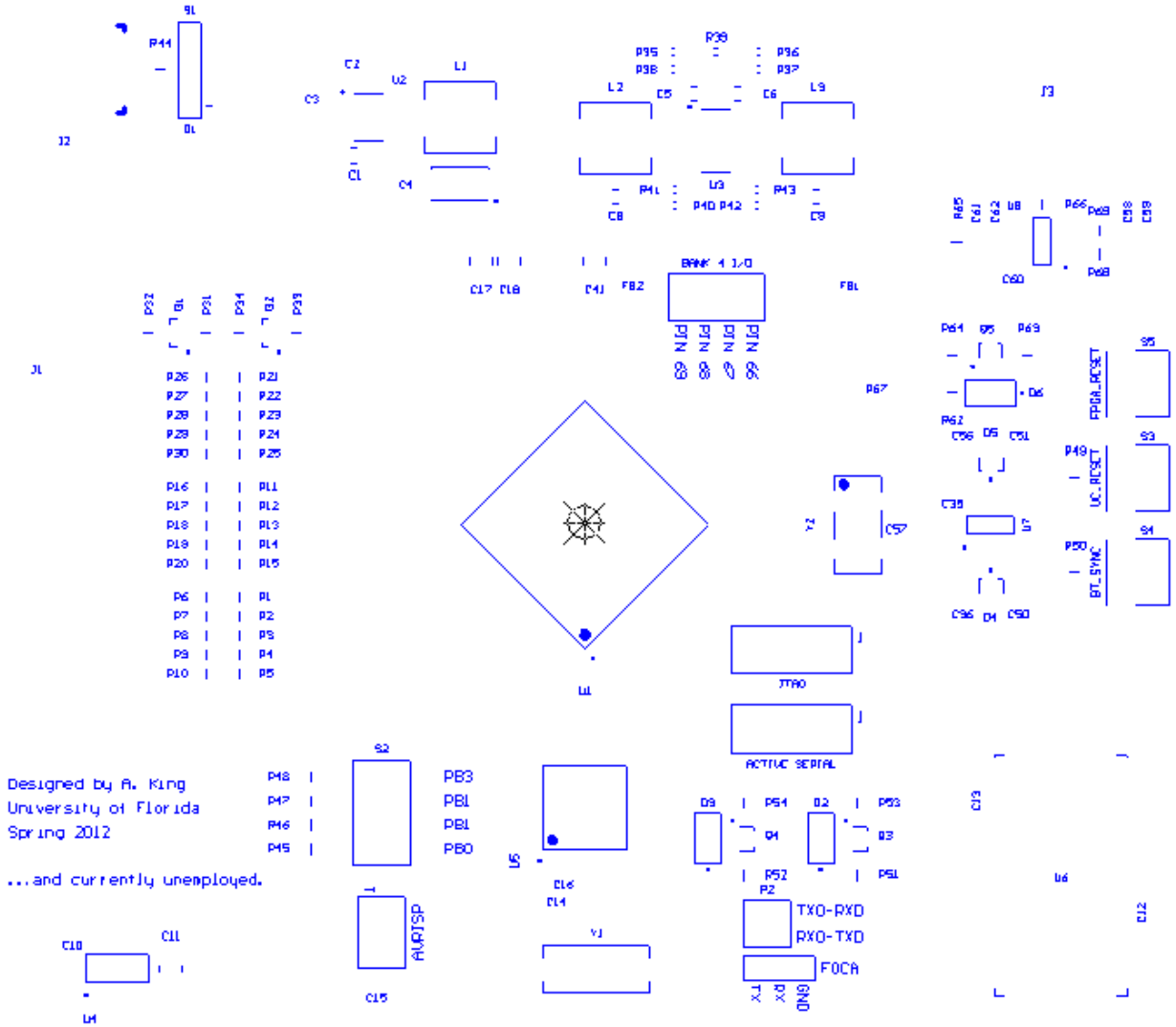
[2] D. Leach. "NES On-A-Chip."
Internet: cegt201.bradley.edu/projgrad/proj2006/fpganes/, [Aug. 25, 2011].

[3] D. Strother. "FPGA NES"
Internet: danstrother.com/fpga-nes/, [Aug. 25, 2011].

[4] J. Donaldson. "VeriNES Nintendo Emulator."
Internet: rm-rfroot.net/nes_fpga/, Aug. 14, 2011 [Aug. 25, 2011].

[5] P. Diskin. "Nintendo Entertainment System Documentation Version 1.0"
Internet: nesdev.parodius.com/NESDoc.pdf, Aug. 2004 [Aug. 25, 2011].

[6] OpenCores.org. "T65 CPU"
Internet: opencores.org/project,t65, Mar. 31, 2010 [Aug. 25, 2011].


Designed by A. King
University of Florida
Spring 2012
...and currently unemployed.
FPGA_RESET
UC_RESET
BT_SYNC
BANK 4 I/O
JTAG
ACTIVE SERIAL
AUART5P
TXD-RXD
RXD-TXD
FOCA
PB3
PB2
PB1
PB0