The Chess Mate

Your Companion for Masterful Moves

Aidan Gallagher, Zachary Jump, Bradley Taylor, Avie Sachdeva
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: aidangallagher@oakland.edu, zjump@oakland.edu, bradleytaylor@oakland.edu, aviesachdeva@oakland.edu

***Abstract*— The Chess Mate serves a purpose to teach players how to play chess or assist new players while also elevating the overall gaming experience. Players who interact with *The Chess Mate* will begin to learn and understand certain strategies for an improved experience in future games. At its core, the program operates on the principle of real-time piece tracking, which intuitively maps the user-selected moves. Beneath the sleek chess surface lies a carefully orchestrated electrical system that optimizes functionality without compromising aesthetics. Driving the computational and display processes are the Dragon and Artix-7 boards, which are set up through serial link to communicate and pass information to complete the task at hand. The completion of this project will transcend the conventional gaming experience and will help guide and teach chess players of any level.**

I. Introduction

As a team, we wanted to contribute to the evolution of chess by implementing our VHDL and embedded software skills into a more modern version of the game we call *The Chess Mate.* Similarly, the game design revolves heavily around two primary aspects, the embedded software within the Dragon12 board that controls the instruction side of the game while the logic design within our Nexys board that handles displays such as the clock found on the LCD along with the LED's mounted on the circuit board beneath the glass chess face. Both boards hold an important role in the functionality of the game, without one working properly the game will not work as intended. The general structure of our circuitry and the VHDL logic we implemented can be found in figure 1. As shown in figure, there are four circuits within the VHDL block diagram. This included the *UART control circuit* to send and receive data, the *Matrix Keyboard* and the *LED Control Circuit* resembling the rows and columns of a chess board with corresponding LED's, and the *7 Segment Debug Control Circuit* to validate and test our design. Using this design, we can assist the players in every way possible by guiding the chess piece location upon startup involving a color code mirroring every chess piece along with indicating their next move.

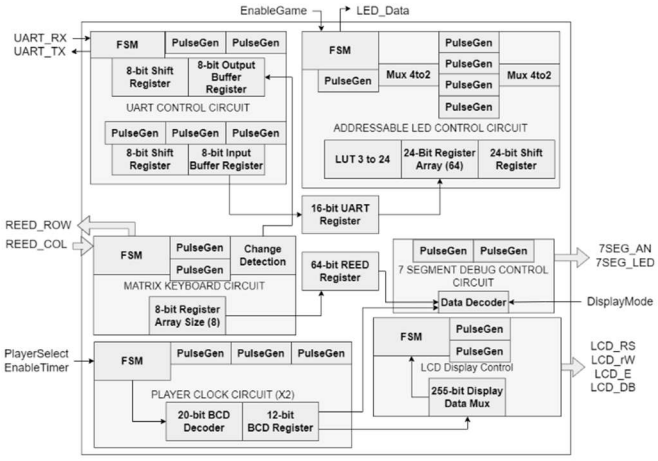


Figure 1: Block diagram

II. Methodology

A. Hardware

As mentioned previously, the project involves the Dragon12 HCS12 microcontroller and the Nexys Artix-7 FPGA board, however the project originally was drafted without the involvement of an FPGA board. It was later decided that for this project to correctly do what we intended, we must involve the FPGA in controlling all aspects regarding the external circuit interfacing as the Dragon 12 would not have sufficient I/O capabilities. Likewise, we found that there would be complications and major roadblocks that would make the project more complicated without the help of the FPGA board. With the addition of the FPGA, we used the FPGA to implement the piece detection reed switches, the addressable LED string, and LCD control circuit with the two devices being able to communicate via UART. Accounting for the chess game clock feature on the FPGA was also an easy decision to make since the ECE 2700 and ECE 4710 labs require much

practice with clock manipulating components and to minimize the different data types being needed to transmit between the boards.

Knowing the responsibilities of each board, we then started drafting what hardware we would need to include beneath the board to notify the Nexys of empty spaces, occupied spaces, and the representation of the spaces upon certain tasks. At first, we decided that we would implement photoresistors to detect spaces occupied since the resistor is light dependent meaning that a chess piece placed above it would limit the light signaling that the space is occupied. However, under further investigation, we found that it would be difficult to implement these resistors on a circuit design that is primarily designed to light certain spaces as light may bleed through spaces which in result will affect the light on other resistors. With this in mind, we decided to involve a reed switch since it is only activated by a magnetic field and not light. Within this switch you will find two metal conductors encased by a glass casing. When a chess piece attached to a small magnet is placed above this reed switch, the switch will then close signifying to the Nexys that a space is occupied. With this in mind, we now needed a way to separate each one of the 64 spaces to make them addressable. Similarly, we decided to involve diodes to allow multiplexed matrix input data connected to RGB LED's to address specific squares with specific colors needed as seen in figure 2 and figure 3.

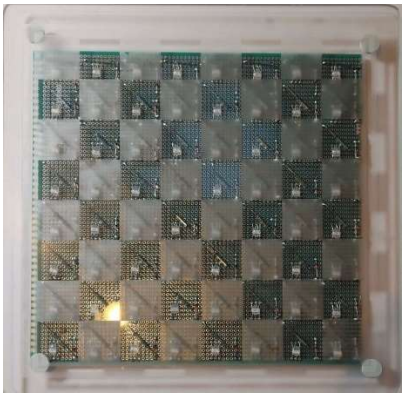


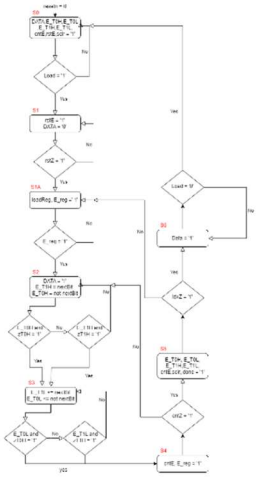
Figure 2: Glass Chess Face

Figure 3: Circuit Board

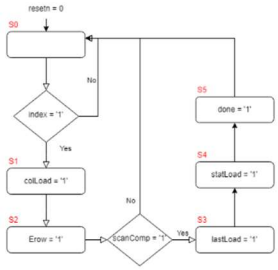
B. *Software*

The Chess Mate system takes advantage of two programs to design this project with: CodeWarrior IDE and Vivado. This implementation of the game of chess is credited almost exclusively to the Dragon12, its IDE, and Embedded C Programming Language. The software design in VHDL through Vivado treats the FPGA and the circuitry it houses as an interface/component of the Dragon12 and the Chess Mate C programmed to it. If we had used another communication method besides UART, the FPGA would be considered the "slave" in the system and the microprocessor would be considered the "master".

The FPGA required a Full Duplex RS-232 serial communication to simplify timing of lighting and piece move detection on the board. From earlier notes in Unit 3, we had a FSM and Datapath for UART transmit, but had no implementation so we started there, followed by the development and testing of a UART receiving FSM and Datapath. Since the TX/RX systems are so similar in there requirements, they were implemented in a single VHDL file with generic input parameters to allow reuse in future projects with runtime configurable Baud Rates. Since reusability was highly desired, the UART function only operates in 8N1 operation mode, so the addition of TX Request and RX Interrupt signals were added to trigger other circuits inside the FPGA. To keep the development of the UART Receive logic simple, we elected to simply delay until the middle of the first data bit and then sample during the middle of each bit instead of observing for noise and debouncing to detect a valid data frame.



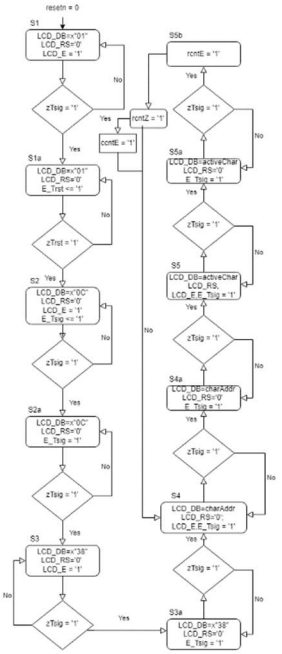
From the UART received data and the related interrupt, we can concatenate the last two bytes received to form the 16-bit status word from the dragon 12 and can use the MSB as a flag to signify a complete word was received and available for decoding and updating the Addressable LED circuit array. The LED array requires 24-bits for full color representation, so a 3 to 24 LUT is used to allow for simplified status word to be used in the dragon 12, with the low order byte from the dragon 12 now being used for the LED array index to be updated when the full word is received over UART.



The Reed switch array was designed and wired in the standard Multiplexed keyboard arrangement, with 1N4148 signal diodes at each location to allow for all locations to be detected simultaneously and accurately. A simple 250us pulse was used to signal reading of a column of reeds, changing which row was to be powered up, and to track that all positions were read. After all rows were enabled and read, the 64-bits are shifted into a status register after the last state of the status register is saved to a memory register, with any changes now being

easily detected by a simple XOR operation and a priority decoder to determine the highest index that changed. The value from the priority decoder was then prepended with the state of the changed index onto the MSB and sent over UART to the dragon, guaranteeing a unique byte for all changes to the reed states beneath the chess board.

To add to the usability of the board, the state of the reed switches was mapped to the 8 7-Segment displays which were treated as 64 individual LEDS for ease of viewing if any reed wasn't being detected. With the change of the Display Mode input to a High, the 7-Segment displays can also show the Chess Clock and last byte received over UART if the offboard LCD becomes unplugged during the game and cannot issue a reset to reconfigure the LCD to a working state.



The LCD control circuit was designed to take a variable data size input to be configurable for any HD44780 compatible LCD using 8-bit data transfer mode. For Chess Mate usage, the display data is hard coded for most of the LCD characters, with the top row being updated with the player clocks from the Clock Control Circuit via BCD to keep the ASCII decoding simple.

The clock control circuit generates an internal 1ms pulse from the 100Mhz clock, which is then used to track the time elapsed for each player during their enabled period. Subtracting the elapsed time in ms from 15 minutes, the circuit generates the decimal value of time remaining in ms. The time remaining is fed into a Binary to BCD iterative converter at each ms update, with the full second values being pulled from and stored in separate 12-bit registers for display usage. Future revisions to include the features used on other chess game modes such as awarding additional time for quick moves.

III. Experimental Setup

Testing and debugging this system is slightly tricky since the two systems require to be connected and communicating before an entire program can be stepped through. Once UART was verified to successfully transmit the bits expected to be received, debugging and testing the two programs was simple. CodeWarrior's debug feature allows us to step through every moment in our game while monitoring our serial communication results in real time. Vivado's simulation element was an avenue for debugging and tweaking the VHDL code while the C program was still being built. By the end of the project, the hardware and circuitry across the system became the tallest hurdle to overcome. Many more wiring, soldering, and power issues presented themselves than were originally expected.

IV. Results

From our development and debugging of the physical circuit, we discovered that the Nexys components were operating as we intended them to during game play, but problems with the LED and reed switch array prevented us from being able to complete a full game of chess as we were unable to detect row H, column 3, and column 4, and it was discovered after soldering all the devices down that the LED address lines were oriented backwards so the index locations were invalid resulting in unusable led states for position suggestions to the user.

Conclusions

Overall, the project itself tested three major aspects, our overall understanding of embedded C, our overall understanding of VHDL, and the importance of teamwork. It was exciting to see what a drafted design was once then shaped into a physical product that as a team we could present to others. However, this final product was not presented without roadblocks along the way. Moreover, this was something that we expected since the scope was widened to aspects, we were not familiar with involving the communication between the Dragon microcontroller and the Nexys FPGA. There are of course ways that the game could be improved within the near future. These improvements include but is not limited to the embedded design within the microprocessor, the physical display, and of course the cost of the project. As the project is something that can be mass produced to the public, we would like to have a more perfected microprocessor that correctly executes instructions to the FPGA with higher level instructions involved such as pawn upgrading. The display would also be changed to involve a box to encase the project with a goal to be plug and play making it easier for the players.