# Interactive 1 Player Checkers

Amelia Becker & Harrison Okun
November 13, 2015

# Overview

Our project will allow a human player to move physical checkers pieces on a board, and play against a computer's projected pieces. Checkers is a two player game—but you can't always find an opponent. There are programs that allow you to play against a computer, but they lacks the feel of really playing the game. For this project we can play against the computer by using a camera to track the movement of the player's pieces and a projector to display the computer's pieces. We will interface between the Nexys 4 and a computer to send the downsampled board configuration from the camera to the computer, which will run the AI to determine its next move. The signal will be transmitted back to the FPGA which will project the computer's move onto the board. We will set up the checkers board on the wall so that the projector and camera can be set up a fixed distance away on a table. Further additions may be other common features of virtual checkers programs such as an undo move feature, a turn select feature, or suggested move feature.

# Design

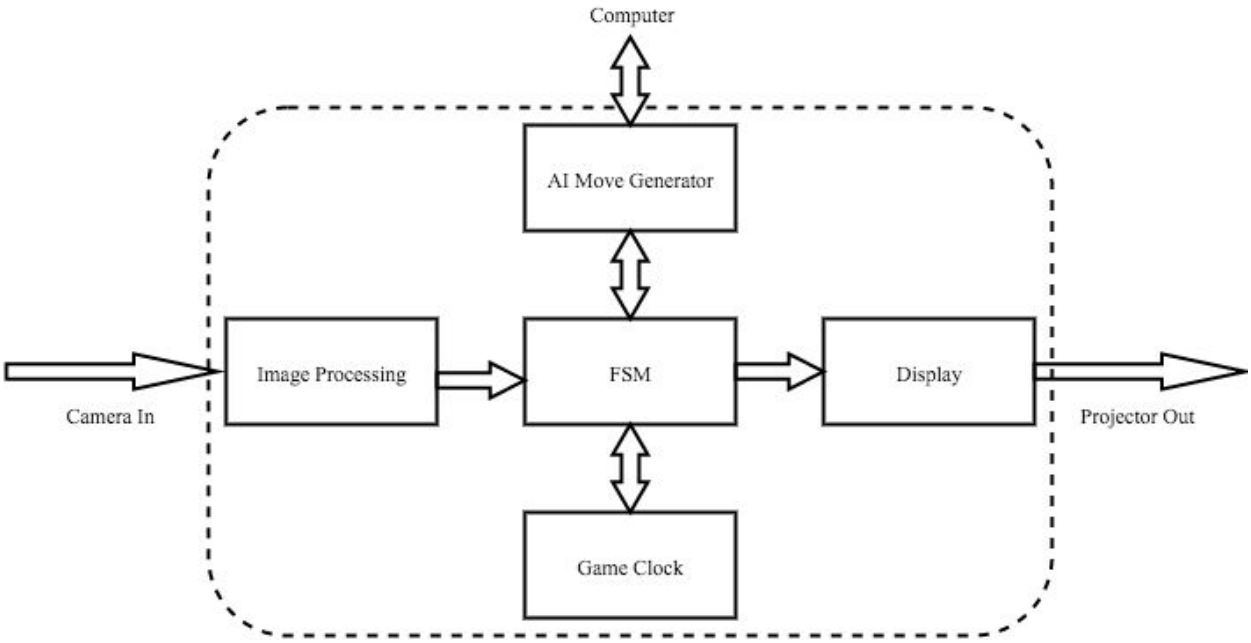


Figure 1: High Level Block Diagram

## Project Overview and Block Descriptions

There are five main components of our project as shown in Figure 1: Image Processing, AI Move Generator, FSM Logic, Display and Game Clock. The Image Processing block interfaces with the camera, telling it when to take images. It then processes these images to determine the board configuration. It transmits the board configuration to the FSM. The AI Move Generator interfaces with computer. It receives new board configurations from the FSM after the player has moved and sends this to the computer. It then receives the new board configuration from the computer and transmits this to the FSM.

The Game Clock shows how much time each player has. It receives signals from the FSM to switch which clock is ticking and tells the FSM when one player's clock expires. The Display module receives the board configuration from the FSM, translates it into VGA and sends this to the projector. The FSM acts as the control unit for the whole systems. It allows communication between the modules and keeps track of the state of the game.

## Design Decisions

In designing our project we had to make some decisions on how certain features would work. The main categories are how we will deal with information, how we will physically configure the board, and how we will represent the computer moves to the player. We will attach the game board to the wall, and put the projector and camera on a table. This makes it easier to set up. We decided to implement the AI on the computer because of our belief that doing so on the FPGA would be impractical. The algorithm would require quickly accessible memory, which the FPGA has a very limited memory (flash memory for example would be way too slow). We will represent the board configuration as a 96-bit binary number. There are 32 valid board locations in checkers. In each location, there can be six possible items: nothing, red piece, black piece, red king, black king, or remove piece image. Thus we can represent each location with 3 bits. We construct our board representations by starting in the upper left corner and going row by row down the board, appending the code of the item in each location. We use a button to signal that the human has moved. This allows us to know when to capture the new board. It would be very difficult to detect the movement of a piece. This also is pretty much necessary for the game clock. For the camera processing, we will reduce the resolution and only take the most significant bits of the different colors (for red and green) to filter the data before it gets to the memory. There are a few edge cases for the checkers game, like when a piece reaches the other side and is “kinged” or jumped ove by the opponent and removed. For kings on the computer we can display a different image. For our pieces we will change the color so it is clear what pieces have which capabilities. When a piece is taken we will display this with an “X” over the piece to be removed.

# Implementation

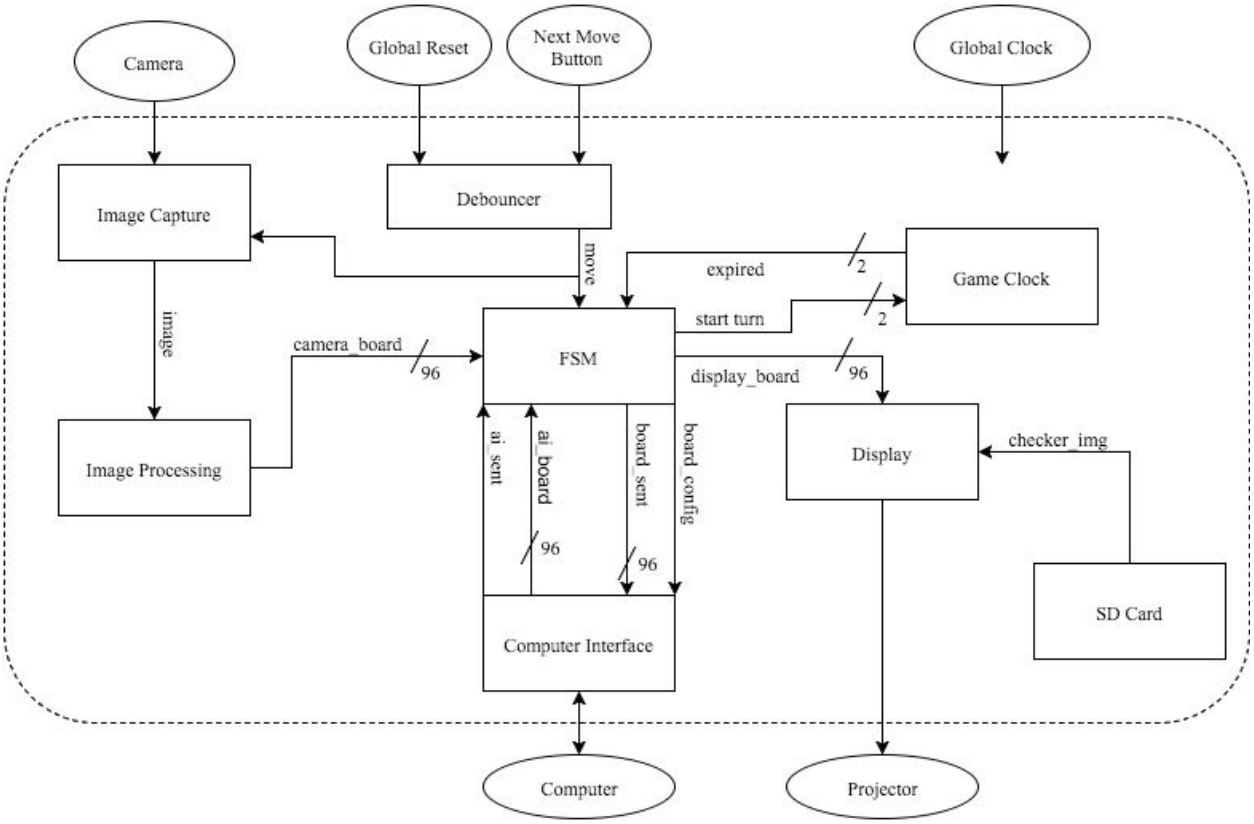


Figure 2: Detailed Block Diagram

## Modules

The above diagram gives a detailed look at the modules we will use in our design and how they will communicate. We now give a description of the purpose of each of these modules and a brief explanation of how we will test them. The modules are organized in the order they will be created.

### Debounce module

The Debouncer debounces and synchronizes external button inputs so they can be used with the clock synchronized modules. We will use the Debouncer module provided in lecture.

### Game Clock module

The Game Clock module will display the time remaining for both players on the hex display. It will use the module from lab 4 to interface with the hex display. It will send a 2-bit signal to the FSM when one player's clock expires. The first bit will indicate that a clock has expired and the second will indicate which player's clock expired. The game clock receives a 2-bit signal from the FSM telling it which clock to run. The first bit will indicate it should start or switch clocks and the second will indicate if it is the player’s turn or the computer’s turn. This will be a feature to facilitate gameplay. The testbench will test

the transitions and that the correct signals are sent when the time expires. The hex display will be helpful for debugging.

## Display module

The Display module will take in the current board configuration from the FSM. It will convert this into a VGA image of the computer's part of the board, which will then be sent to the projector. This allows the computer’s checker pieces to be displayed on the board. We will be able to test that the display is correctly interpreting the board configuration by connecting it to a monitor, so as not to have to debug with the projector every time.

## Computer Interface module

The Computer Interface module provides an interface between the computer and the FSM. It takes in a board configuration and an activation signal from the FSM and transmits the board state to the computer. It then receives the board configuration corresponding to the computer's move from the computer. It transmits this and an activation signal to the FSM. It allows the rest of the system to interact with the computer. We will test this with edge cases and verify the logic with other literature about the subject. The test will be used with the intermediate display on the monitor for sample games.

## Image Capture module

The Image Capture module tells the camera to capture an image whenever the move button is pressed and sends the images to the Image Processing module. This allows the system to capture the board state. Testing this module will also use the display making sure that the image is captured when and only when the move button is enabled, and that the information is sent in an easily manipulable format.

## Image Processing module

The Image Processing module takes in an image from the Image Capture module and converts it into a game board configuration which it then sends to the FSM. This will begin by downsampling the image and then filtering to take the most significant bits for the red’s and green’s. Then we will calculate the center of mass of the pieces and translate those located to the 92 bit string sent to the FSM module for the computer AI. It will process the raw image and turn it into a form usable by the rest of the system. The image processing module will be tested with the display to show what the image captured is, and to verify that it is the same as our 96 bit signal represents.

## FSM module

The FSM keeps track of the game state and provides a communication channel between modules. It will keep track of whose turn it is and what the current board state is. Whenever the player moves, it get the new board configuration from the Image Processing module and sends it to the Computer Interface module along with the activation signal. It also tells the Game Clock to switch to the Computer's clock. When the computer computes its move, the FSM receives the new board configuration and sends it to the Display module to be displayed. It also tells the Game Clock to switch to the player's clock. If the Game Clock ever expires or the computer indicates the game is over, the FSM records that the game is over and freezes the system. The FSM is essentially the control block for the whole system.

# Timeline: Testing and Division of Labor

Types of benchmarks:

- Module development: debounce, game clock, image capture, image processing, game FSM, display logic, computer AI logic
- Testing individual modules (see implementation sections): image capture, image processing, game FSM, display logic, computer AI logic
- Interface: mounting the projector and camera
- Optional developments: Undo move feature, Suggested moves (project arrow onto board)
- Buffer time
- CIM Deliverables: Project proposal draft (11/03), project design presentation (11/10 & 11/12), Project Proposal Revision (11/13), Project Checkoff checklist meeting (11/20), Project Status Update (11/30), Final Project Checkoff (12/7), Project Demo (12/8), Final Project Report (12/9)

| Amelia | Harrison |
|---|---|
| ● image capture<br>● image processing<br>● mounting setup | ● computer AI logic<br>● FSM logic<br>● display logic |

| Week | Harrison | Amelia | Both | Deliverables |
|---|---|---|---|---|
| 11/2 | | | Presentation materials, Global modules | Proposal draft (11/3 5PM), Presentation rehearsal (11/5 11AM) |
| 11/9 | Computer AI Logic module | Image capture module | | Presentation (11/10 2:30-5:00PM), Proposal Revision (11/13 5PM) |
| 11/16 | Display logic module, Test: AI module | Image processing module, Test: Image capture module | | Checkoff checklist meeting |
| 11/23 | Game FSM module, Test: Display | Projection and camera setup and calibration, | | |

| | | | | |
|---|---|---|---|---|
| | logic | Test: Image processing | | |
| 11/30 | | | Full testing, optional developments | Status update with mentor |
| 12/7 | | | | Final project checkoff (12/7 4-9PM),<br>Project demos (12/8 6-11PM),<br>Final project report (12/9 5PM) |

Figure 3: Project Timeline

# Resources

We will need additional resources for the player interface. We have a checker board and pieces, but we will need a camera and projector, and associated wires. We will also need the wires to connect the computer to our board. We will use the displays in the lab for testing. As we do not yet know how well the camera interface will be able to detect all of the existing checkers pieces, we may need neon tape to create bright, and easily trackable by the camera, pieces and board.

# Conclusion

Our project will allow a human to play checkers against a computer on a real board. This is accomplished by using a camera to capture an image of the board then processing this on the FPGA. We then send the board configuration to the computer which calculates the next move and sends it back to the FPGA. The FPGA converts this into VGA showing the computer's pieces and sends it to the projector which projects it onto the board. We will also have a game clock and have stretch goals of several other game features we could implement.