ECE532 – FINAL DESIGN REPORT
LIVE GO GAME RECORDER

Team 5
April 8, 2016

Jesse Barcelos
Fadime Bekmambetova
Chaehyun Park

Table of Contents

1. Overview

1.1. Background & Motivation

The board game Go, which originates from China, is estimated to be 2500 to 4000 years old and is the oldest game played in its original form [1]. Go game is a turn-based strategy game where two players, having either white or black stones, try to get more territories/points than the opponent on a 19x19 grid board. The players can place stones anywhere at the intersection of grid lines, making the Go game almost limitless with possibilities [2].

Its supreme complexity as a perfect information game has enticed many beginners to continue playing and keep the game alive for millennia. Its complexity has also attracted Google DeepMind to create AlphaGo AI and beat a human professional Go player, Lee Sedol.

It is estimated that there are 60 million Go players around the world [3] and with the surge of new players in Go from the public media coverage, a tool in aiding improvement of individual Go skills is much needed. The Go communities advise the beginners to always review the games [4]. However, beginners or intermediate-level players have a hard time reviewing on physical Go board simply because they do not remember the long sequences of up to 400 moves [2].

1.2. Goals

The main goal of the Live Go Game Recorder (LGGR) is to give Go beginners capabilities to accurately review their game. The LGGR solves the reviewing problem by converting a video stream of a live Go game into a virtual record of the Go game, specifically into an already widely used Smart Game File (.SGF) as shown in Fig. 1.

Figure 1. Go Board + Video Camera = SGF file **[5]**.

We wanted to automate the LGGR as much as possible where the user only needs to flip one switch and everything works with no calibration on Xilinx Nexys video board. Therefore, 6 specific technical functionalities have been devised to be implemented in the system as shown in the list below:

- Automatic board detection
- Automatic stone detection
- Hand detection
- Error checking for illegal moves
- FAT system on microSD card
- SGF file writer

1.3. Block Diagram

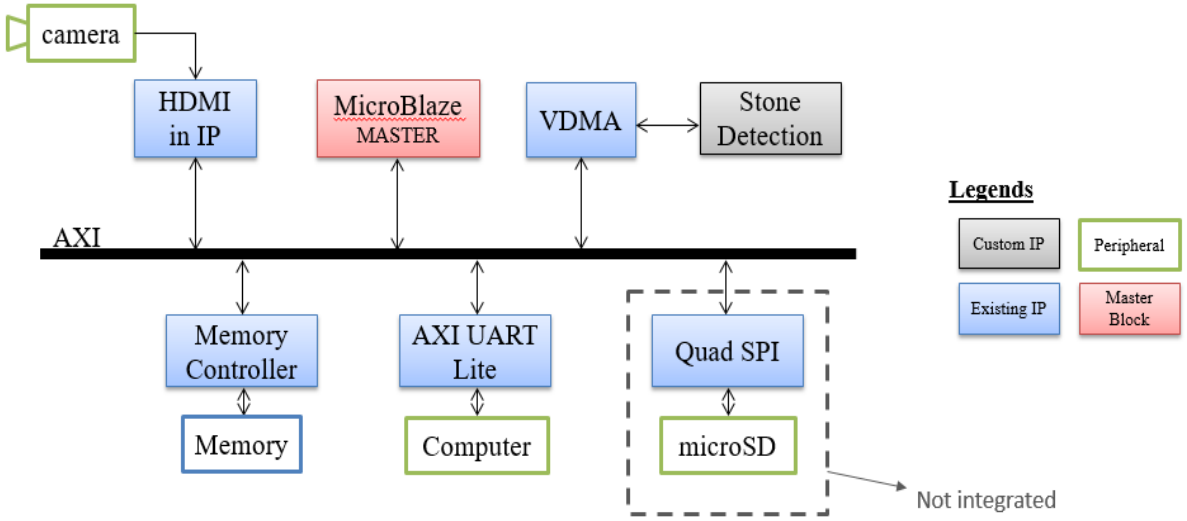

Figure 2 Block Diagram of LGGR

Frame of Go board is fed into the AXI the first VDMA (HDMI in IP in Figure 2) and written to the memory. Upon receiving a user command to run the stone detection IP through AXI UART, the second vdma (VDMA in the Figure 2) reads the frame from memory and passes the frame pixel-by-pixel to the stone detection IP. The stone detection IP eventually finishes processing to get the coordinates of the placed stone. The MicroBlaze then writes the coordinates including the color of the stone to the DDR memory. Eventually, when the user is finished recording the game, MicroBlaze reads the color of the stone and coordinates from memory and passes the results onto Quad SPI to write to microSD in .SGF format.

1.4. Brief Description of IP

TABLE I
IPS USED IN THE PROJECT

IP blocks present in the design

axi_vdma_1	axi_vdma	6.2	
axi_vdma_0	axi_vdma	6.2	
microblaze_0_local_memory_dlmb_bram_if_cntlr	lmb_bram_if_cntlr	4.0	
v_tc_0	v_tc	6.1	
microblaze_0_local_memory_ilmb_bram_if_cntlr	lmb_bram_if_cntlr	4.0	
v_tc_1	v_tc	6.1	
mdm_1	mdm	3.2	
microblaze_0_local_memory_lmb_bram	blk_mem_gen	8.3	
v_axi4s_vid_out_0	v_axi4s_vid_out	4.0	
microblaze_0_axi_periph	axi_interconnect	2.1	
rgb2dvi_0	rgb2dvi	1.2	
dvi2rgb_0	dvi2rgb	1.5	
microblaze_0	microblaze	9.5	
microblaze_0_local_memory_dlmb_v10	lmb_v10	3.0	
microblaze_0_local_memory_ilmb_v10	lmb_v10	3.0	
axi_mem_intercon	axi_interconnect	2.1	
v_vid_in_axi4s_0	v_vid_in_axi4s	4.0	
rst_mig_7series_0_100M	proc_sys_reset	5.0	
microblaze_0_xlconcat	xlconcat	2.1	
GoStoneDetectIP_0	GoStoneDetectIP	1.0	
axi_uartlite_0	axi_uartlite	2.0	
microblaze_0_axi_intc	axi_intc	4.1	
axi_timer_0	axi_timer	2.0	
axi_dynclk_0	axi_dynclk	1.0	
rst_mig_7series_0_pxl	proc_sys_reset	5.0	
axi_gpio_video	axi_gpio	2.0	Registers
mig_7series_0	mig_7series	2.4	
xlconstant_0	xlconstant	1.1	

2. Outcome

2.1. Results

We have met most of the originally proposed functionalities, tabulated in Table II.

TABLE II
STATUS OF PROPOSED FUNCTIONALITIES
WITH MODIFICATIONS

Functionalities	Status
Hand Detection	Omitted
~~Automatic Stone Detection~~ Semi-automatic Stone Detection	Completed
~~Automatic Board Detection~~ Manual Board Calibration	Completed
Error Checking	Completed
FAT System on microSD card	Completed on another board
SGF File Writer	Completed on another board

As shown in Table II, 5 out of 6 originally proposed functionalities with some modifications were completed. The modifications were unavoidable when the team decided to eliminate the hand detection functionality. Our project was originally designed to be automated assuming that the hand detection gracefully detects the hand above the board, allowing the intermediate frames with hands to be filtered out. However, the hand detection was not working so it had to be omitted out of the project.

The omission of the hand detection meant that intermediate frames with hands are not filtered out, thereby forcing the stone detection IP to possibly detect hands as stones. Therefore, a user input to feed in the frame without the hands was required in the project.

The automatic board detection was modified to manual board calibration. Originally, the plan was to precisely detect the position of the board in the frame so that the user does not need to align the board in a perfect manner. However, during the software prototyping phase, the auto board detection proved to be non-trivial. Due to the time constraints, the board was set to be positioned in a certain alignment.

To expedite the development progress, our work has been parallelized where FAT system on microSD card and SGF file writer have been implemented on the Nexys 4 DDR board. Due to integration issues, LGGR ended up with two systems: stone detection system, and SGF writer system. Moreover, to showcase the stone detection system, two additional softwares had to be created: a graphical overlay of the board for board calibration and a SGF file parsing writer.

Although LGGR project ended up in two projects, individual functionalities worked, especially the stone detection IP. If we could start over, we would have sacked the idea of automatic board detection earlier and allocated a lot more time for integration. Nonetheless, this project is considered to be a success.

2.2. Next Steps

The next step is to finish integrating the two systems into one LGGR system. Only after the integration is done, optimizations should be made to the stone detection IP and possibly eliminate HDMI output in the system.

The stone detection IP can be optimized by using BRAMs for storing the Moves data. Currently, slow memory access DDR is being used and the stone detection IP is able to achieve 139.5ms per 1280x720 resolution frame, approximately 8 frames per second. Although the performance was almost three times better than our reference design Go-getter, which had 350ms, there are rooms for optimization [6].

After optimizations, hand detection should be added to make the system completely autonomous after board calibration.

3. Project Schedule

Our actual weekly accomplishments differed greatly from our original milestones presented in the proposal. They have been compared side-by-side in Table III.

TABLE III
PROJECT SCHEDULE COMPARISONS

W#	Original Milestones	Final Accomplishments
1	Start Matlab Software Prototype	Started Matlab Software prototype
	Stream HDMI video through Xilinx board	Started automatic board detection prototype
	Configure switches for user inputs	
2	Finish Matlab board detection prototype	Configured switches for user inputs
	Finish Matlab stone detection prototype	Finished SGF file writer
3	Finish board detection on hardware	Streamed HDMI video through Xilinx board
	Start FAT system for microSD on MicroBlaze (MB)	
4	Finish stone detection on hardware	Eliminated automatic board detection
		Finished Matlab stone detection prototype
5	Finish FAT system for microSD on MB	Started implementation of FAT system for microSD card on MB
	Start Implementing SGF file writer on MB	
	Optimize memory controller	
6	Finish implementing SGF file writer on MB	Wrote overlay software for manual board detection
		Started integrating stone detection IP on the HDMI system
7	Catch-up	Finished FAT system for microSD on MB
8	Catch-up	Finished and tested FAT system & SGF file writer on Nexys 4 DDR board
		Wrote SGF file parsing writer for showcase
		Finished and tested stone detection IP on the HDMI system

The software prototype finished at week #4 when it was supposed to be finished by week #2. The progress was slower than expected and as mentioned earlier, the idea of automatic board detection was not eliminated until week #4. Under strict timelines, this has set back the progress of the project by 2 weeks.

In addition, there were several unresolved integration issues, such as something simple as declaring variables in the MB code, that have set back the development. Much of the time from week #6 to week #8 was spent in integrating the stone detection IP into the HDMI demo provided by Xilinx [7].

4. Description of the Blocks

No change to the HDMI demo provided by Xilinx was made except for adding two IPs: a second VDMA and a custom made stone detection IP [7].

4.1. VDMA

Xilinx VDMA version 6.2 has been used for feeding video frames into the custom IP, GoStonDetect IP by specifying it to have read-only channels with data width of 24 bits [8].

4.2 GoStoneDetect IP

GoStoneDetect IP is a 32-bit AXI Lite slave IP designed to perform two tasks, to obtain stone coordinates on the board and to check errors.

4.2.1. Stone Detection

GoStoneDetect IP is responsible for converting a video frame of Go board into the coordinates of the newly placed stone on the board. By comparing the previous board configuration with the current board configuration, it is able to calculate the newly placed stone and update the Move struct in the memory.

Move structs store the sequences of the moves played in the game in DDR memory. Each Move struct consists of three members: color, i, and j as shown below:

Move struct:

color	i	j	color	i	j	color	i	j	color	i	j	…	color	i	j

The color can either be black (2) or white (1) and i, j are the 2D coordinates on the board that has a range of 0 to 18 (inclusive). It has been assumed that Go games do not exceed 800 moves, so there is 800 Move structs allocated in the memory.

The algorithm in determining the coordinates from a frame is by checking the regions around the intersection of the lines. The regions have not included the black grid lines because of risks of detecting them as black stones when there is no stone on the lines. Rather, 4 rectangular regions have been checked near the intersection of lines as shown in Figure 3.

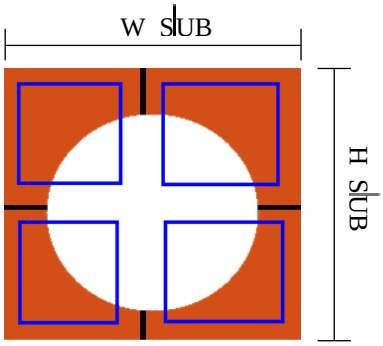

Figure 3 Regions being checked

The widths and heights of these blue rectangular regions in Figure 3 are configured within the MicroBlaze code:

```
#define incl_iA ((int) (W_SUB*0.3))
#define incl_iB ((int) (W_SUB*0.35))
#define incl_iC ((int) (W_SUB*0.65))
#define incl_iD ((int) (W_SUB*0.7))

#define incl_jA ((int) (H_SUB*0.3))
#define incl_jB ((int) (H_SUB*0.35))
#define incl_jC ((int) (H_SUB*0.65))
#define incl_jD ((int) (H_SUB*0.7))
```

After these rectangular regions have been configured, the code calculated for the maximum and minimum brightness values for each region and stored them in white_eval and black_eval array. Then minimum and maximum values of these entire arrays are determined to calculate mid_white or mid_black, where midpoint = (max + min)/2. This way, the project ensures that relative thresholds for checking color of the stones are used to accommodate different lighting settings.

The color of the stone is then determined by checking the following conditional statement:

```
If (white_eval – mid_white > max_white_dev * threshold){
        // Then it is white
} else if (black_eval – mid_black > max_black_dev * threshold){
        // Then it is black
}
```

However, because relative threshold levels are used, both black and white stone must be present on the board at the start of the recording. The current code assumes black goes first in the game, so the game can be recorded at the start of the third move.

4.2.2. Error Check

The error check is needed to determine whether the current frame fed into the IP makes sense. The error check function, *interpret(int print_result)*, can be found in *video_demo.c*.

For example, if the IP detects that two or more stones have been played since its previous board configuration, it should throw an error because it cannot determine the order of which

the stones have been played as shown in the code below:

```
for (j = 0; j<19; j++){
    for (i = 0; i< 19; i++){
        // if white appeared
        if (config_old[j][i] == NO_STONE && config_new[j][i] == WHITE_STONE){
            if (!action_seen || black_removed){
                white_placed = 1;
                coord_j = (char) (j&0xff);
                coord_i = (char) (i&0xff);
                action_seen = 1;
        }
        else{
            unresolved = 1;
        }
     …
```

As shown in the code above, once action has been seen, if another white stone is detected in another coordinate, then unresolved/error flag is set to 1.

Another check the IP does is the liberty check of removed stones. If stones disappear from the board, it should be from capturing a group of stones by removing all liberties around it as shown in the code below:

```
if (config_old[j][i] == WHITE_STONE && config_new[j][i] == NO_STONE){
    //check liberties
    if (i >= 1)
        if (config_new[j][i-1] == NO_STONE && config_old[j][i-1] == NO_STONE)
            unresolved = 1;
    if (i <= 17)
        if (config_new[j][i+1] == NO_STONE && config_old[j][i+1] == NO_STONE)
            unresolved = 1;
    if (j >= 1)
        if (config_new[j-1][i] == NO_STONE && config_old[j-1][i] == NO_STONE)
            unresolved = 1;
    if (j <= 17)
        if (config_new[j+1][i] == NO_STONE && config_old[j+1][i] == NO_STONE)
            unresolved = 1;
…
}
```

Currently, when the unresolved flag is set to 1, the code notifies the user of the error. The user cannot fix the error but is left with no choice but to keep recording the game. In the future, we hope to give the users a manual control to fix the error and continue with LGGR gracefully.

Through these two checks, the GoStoneDetect IP was able to achieve high accuracy in detecting stones on the board at around 139.5ms.

5. Description of the Design Tree

As we have been told, the Github repository will be uploaded by Monday, April 11, 2016. This document will also be updated with the information.

6. Tips and Tricks

- Start early
- Be ready to accommodate a lot of time for integration

References

[1] American Go Association, "A Brief History of Go," 2014. [Online]. Available: http://www.usgo.org/brief-history-go. [Accessed 2 April 2016].

[2] Sensei's Library, "Number of Possible Go Moves," 24 March 2016. [Online]. Available: http://senseis.xmp.net/?NumberOfPossibleGoGames. [Accessed 5 April 2016].

[3] British Go Association, "Frequently Asked Questions about Go," 17 March 2016. [Online]. Available: http://www.britgo.org/press/faq.html. [Accessed 5 April 2016].

[4] D. Ormerod, "Go Game Guru," 7 August 2011. [Online]. Available: https://gogameguru.com/how-to-get-better-at-go/. [Accessed 2 April 2016].

[5] A. Carta and M. Corsolini, Artists, *PhotoKifu Banner.* [Art]. 2016.

[6] BostonUniversityECE, "ECE Day '08 - Team 18: The Go-Getters (Part 1)," 14 May 2008. [Online]. Available: https://www.youtube.com/watch?v=2Jzi2h_dWCE. [Accessed 30 January 2016].

[7] Digilent Inc., "Nexys Video Resource Center," 18 February 2016. [Online]. Available: https://reference.digilentinc.com/nexys-video:start. [Accessed 20 January 2016].

[8] Xilinx, "AXI Video Direct Memory Access v6.2," 18 November 2015. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/axi_vdma/v6_2/pg020_axi_vdma.pdf. [Accessed 20 January 2016].

[9] Sensei's Library, "Number of Possible Go Games," 24 March 2016. [Online]. Available: http://senseis.xmp.net/?NumberOfPossibleGoGames. [Accessed 6 April 2016].