CSEE 4840: Go Presentation

Group Members: Maximilian Comfere, Owen Cooper
Instructor: Prof. Stephen A. Edwards

Presentation Overview

Project Overview
Hardware Design
Software Design
Register Map

Project Overview

Game Modes

- PvP
- PvAI

Input

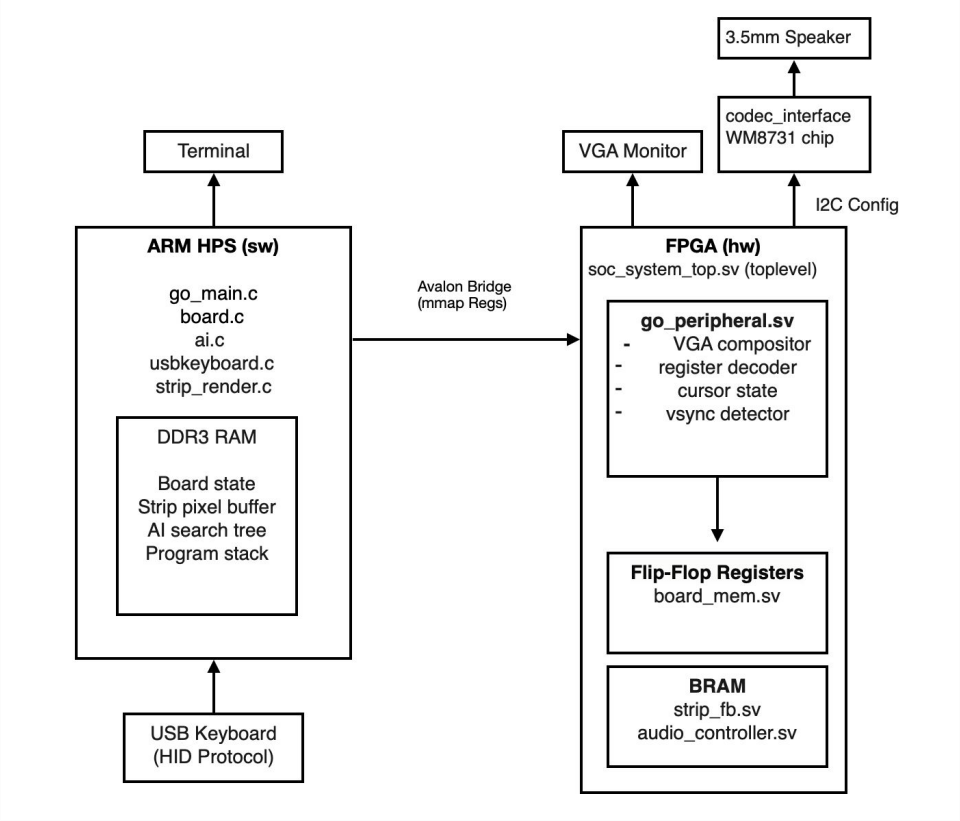
- USB Keyboard

Output

- VGA Monitor
- Sound Effects
- Terminal (running game log)

Hardware Acceleration

Block Diagram



System Architecture

- ARM runs the game logic in sw
- FPGA handles visual output and audio in hw
- They communicate through the Avalon Lightweight Bridge

Hardware Design

Files

- go_peripheral.sv
- strip_fb.sv
- audio_controller.sv
- board_mem.sv

Hardware Acceleration

go_peripheral.sv

- The main FPGA code
- Decodes ARM register writes and composites VGA pixels every frame

if (in_strip) → read strip_fb → palette color
else if (on_cursor) → green
else if (in_stone) → black or white circle
else if (on_grid) → black line
else → burlywood background

strip_fb.sv

- Double-buffered 640×60 framebuffer for the score strip
- writes to back buffer first
- Swap happens at vsync (pulse is the non-read window)

```
if (swap_armed && vsync_pulse)
    active <= ~active;  // flip buffers
```

cleanly during blanking

Double Buffer

- ARM writes to back buffer (DDR3)
- FPGA Reads from the front one
- Swap happens at vsync (pulse is the non-read window)

board_mem.sv

- 81-cell BRAM storing stone state (2 bits/cell)
- ARM writes, VGA reads

```
cells[write_addr] <= write_data;
// ARM places a stone
read_data = cells[read_addr];
// VGA reads it every pixel
```

audio_controller.sv

- Plays one of 4 sound effects from ROM
  - Place
  - Capture
  - Illegal
  - Game over
- Upsamples 8kHz PCM to 48kHz by holding each sample 6 times

if (sub_count == 5) → advance to next sample
else → hold current sample

Hardware Acceleration

- The board is rendered on the FPGA, not the ARM chip
- Combinational (far faster):
  - Compute which cell that pixel belongs to
  - Read that cell's value from board_mem
  - Calculate distance^2 from the cell center
  - Decide color based on distance^2 and cell value

Software Design

Files

- go_main.c
- board.c
- ai.c
- strip_render.c
- usbkeyboard.c

go_main.c

- Main game loop
- Reads keyboard, runs the UI state machine, and commands the FPGA

```
// One byte write = stone on screen instantly
go_regs[REG_SET_BLACK] = cell_idx(row, col);
go_regs[REG_AUDIO_CMD] = AUDIO_PLACE;
```

board.c

- Go rules engine
- Validates moves, removes captured groups, tracks ko, scores board

```
MoveResult board_place(BoardState *b, int row, int col);
// Returns: MOVE_OK / ILLEGAL_OCCUPIED / ILLEGAL_SUICIDE / ILLEGAL_KO
```

ai.c

- Three AI levels
- All C running on the ARM chip

Level	Strategy
L1	Random legal move
L2	Greedy, prefers captures, defends, center bias
L3	MCTS, 200 simulated games, picks best result

strip_render.c

- Renders for the score strip
- Draws text and boxes into a local buffer, then writes all of it to the FPGA

```
strip_clear(COLOR_STRIP_BG);
strip_text_centered(4, "TURN  BLACK", 2, WHITE, BG);
strip_present();  // 9,600 word writes → FPGA back buffer → swap on vsync
```

usbkeyboard.c

- Opens the USB keyboard via libusb
- Reads HID boot-protocol packets

```
libusb_interrupt_transfer(kbd, endpoint, &pkt, sizeof(pkt), &xferred, 10);
uint8_t key = pkt.keycode[0];  // first held key's scan code
```

Register Map

Offset	Name	Direction	Data	Effect
0x00	SET_BLACK	ARM → FPGA	cell index (0–80)	Places a black stone at that cell
0x01	SET_WHITE	ARM → FPGA	cell index (0–80)	Places a white stone at that cell
0x02	CLEAR_CELL	ARM → FPGA	cell index (0–80)	Removes the stone at that cell
0x03	RESET_BOARD	ARM → FPGA	any	Clears all 81 cells to empty
0x04	CURSOR	ARM → FPGA	{visible[7], cell_idx[6:0]}	Moves/hides the green cursor ring
0x05	STRIP_SWAP	ARM → FPGA	any	Arms a score strip buffer swap at next vsync
0x06	AUDIO_CMD	ARM → FPGA	1–4	Triggers a sound effect (place/capture/illegal/gameover)
0x07	AUDIO_STATUS	FPGA → ARM	bit 0 = busy	Tells software if audio is still playing
0x10000+	STRIP_FB	ARM → FPGA	32-bit words	Writes pixels to score strip back buffer (9,600 words)