CSEE 4840 Project Proposal: 9×9 Go

Team Members:
Maximilian Comfere (mkc2182), Owen Cooper (odc2106)

CSEE 4840 — Spring 2026

Project Overview

This proposal outlines the design and implementation of a 9×9 Go game on the DE1-SoC FPGA board. Go is one of the oldest and most strategically deep board games in the world, yet its rules are elegantly simple: two players alternate placing black and white stones on a grid, competing to surround the most territory. The 9×9 board variant preserves the depth of full-size Go while being well-suited to the resource constraints of an embedded platform.

The system will feature a VGA-rendered graphical interface, USB controller or keyboard input, audio feedback for stone placement and game events, and multiple AI opponents of varying difficulty. Players can choose between Player vs. Player (local) and Player vs. Computer modes, selecting from several bot difficulty levels ranging from beginner-friendly to challenging.

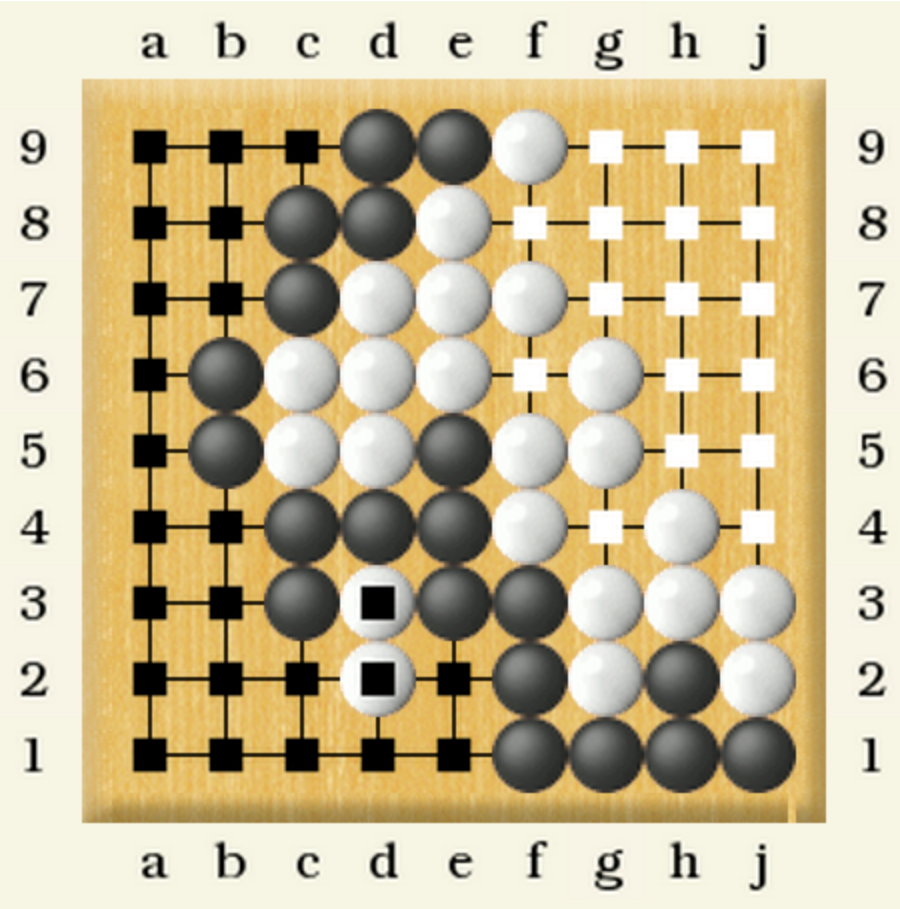

Figure 1: A 9×9 Go board with black and white stones.

Game Rules and Mechanics

Go's ruleset is compact, making it feasible to implement entirely in an embedded environment:

1. **Stone Placement:** Black and white alternate placing stones on the intersections of a 9×9 grid. Black moves first.
2. **Liberties and Capture:** A stone or connected group of stones is captured and removed from the board when all of its adjacent empty intersections (liberties) are occupied by the opponent.
3. **Ko Rule:** A move that would recreate the immediately previous board position is forbidden, preventing infinite loops.
4. **Passing and End of Game:** A player may pass their turn. The game ends when both players pass consecutively.
5. **Scoring:** We will implement Chinese-style area scoring (stones on the board + surrounded empty intersections). The final score is displayed at the end of the game.
6. **Suicide Rule:** A move that would result in the player's own group having zero liberties (without capturing an opponent group) is illegal.

Game Modes

1. **Player vs. Player:** Two players share a USB keyboard or use two USB controllers to take turns placing stones. A cursor navigates the board to select an intersection.
2. **Player vs. Computer:** The player competes against one of several AI bots:
   - **Level 1 — Random:** Places stones on random legal intersections. Suitable for beginners learning the rules.
   - **Level 2 — Greedy Capture:** Prioritizes capturing opponent stones and defending its own groups when in atari (one liberty remaining). Uses simple heuristics for territory.
   - **Level 3 — Monte Carlo:** Uses a lightweight Monte Carlo tree search (MCTS) with random playouts to evaluate moves. The number of simulations is tuned to run within a reasonable time on the ARM HPS. This provides a legitimately challenging opponent on a 9×9 board.

Key Features

1. **I/O:** The game runs on the DE1-SoC FPGA board. A USB keyboard or controller serves as input. VGA output displays the board, stones, UI elements (score, captured stones, current turn), and menus. Audio output provides sound effects for stone placement, captures, and game-over events via the 3.5 mm audio jack.
2. **Complete Rule Enforcement:** All Go rules (liberties, capture, ko, suicide, scoring) are enforced programmatically. Illegal moves are rejected with visual/audio feedback.
3. **Game State Display:** The GUI shows the current board, captured stone counts (prisoners) for each player, whose turn it is, and the currently selected intersection highlighted by a cursor.
4. **End-of-Game Scoring Screen:** When the game ends, the board displays territory ownership with colored shading and a final score breakdown.
5. **Difficulty Selection Menu:** A start screen allows the player to choose the game mode (PvP or PvC) and bot difficulty level before the game begins.

Technology Stack

Hardware:

- DE1-SoC FPGA board (Cyclone V)
- VGA monitor for display output
- USB keyboard or controller for input

- 3.5 mm speaker for audio output

Software:

- SystemVerilog for FPGA peripherals (VGA controller, audio controller, input handling)
- C running on the ARM HPS (Hard Processor System) for game logic, rule enforcement, and AI
- Linux userspace drivers for communication between HPS software and FPGA hardware

AI Engine:

- Game logic and AI implemented in C on the ARM processor
- Level 1–2 bots use rule-based heuristics (negligible computation)
- Level 3 bot uses Monte Carlo tree search; 9×9 board size keeps the search space manageable for the ARM Cortex-A9

Hardware-Software Interface

The system partitions work between the FPGA fabric and the ARM HPS:

Component	Runs On	Description
VGA Display	FPGA	Generates video signal, reads from a framebuffer/tile map in shared memory
Audio	FPGA	Plays sound samples triggered by the HPS via memory-mapped registers
Input Handling	FPGA + HPS	FPGA handles USB/keyboard low-level protocol; HPS reads decoded key events
Game Logic	HPS (ARM)	Full Go rule engine, board state, scoring, turn management
AI Engine	HPS (ARM)	Bot move computation (heuristic or MCTS)
Board Rendering	HPS → FPGA	HPS writes board state to shared memory; FPGA VGA module renders the grid, stones, and UI

Communication between HPS and FPGA uses memory-mapped I/O over the Avalon bus (lightweight HPS-to-FPGA bridge).

System Block Diagram