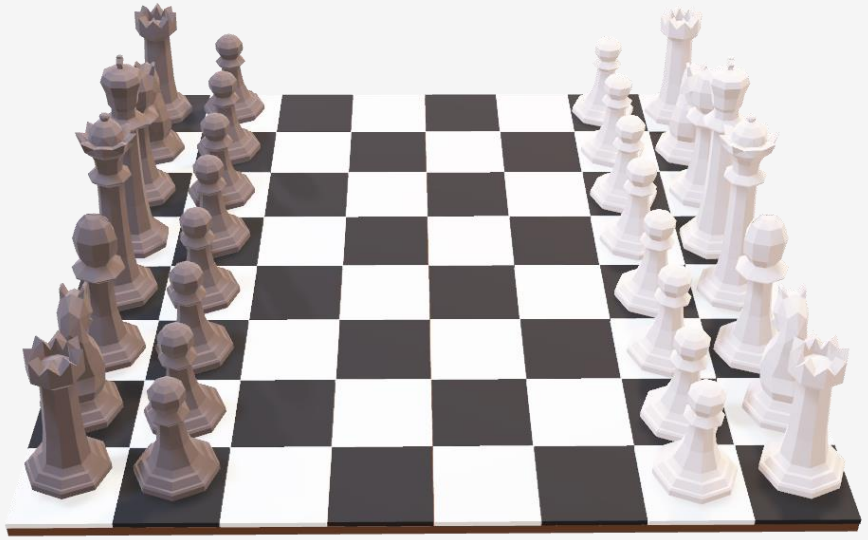
ChessAI

Play against a computer IRL

The Chess Board

What's so special about it?

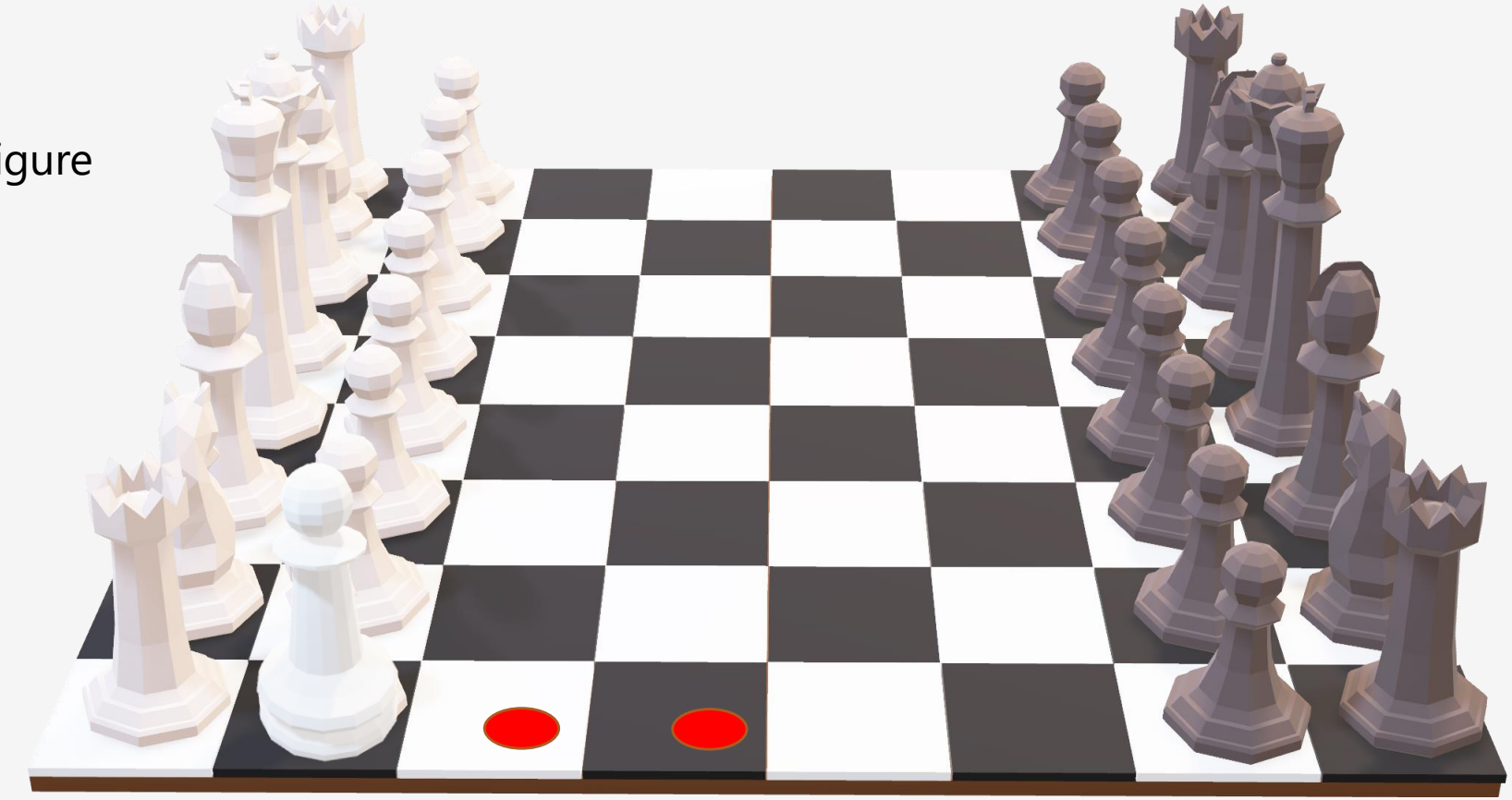
- Detects the positions of the figures
- Shows legal moves
- Shows opponent's moves



You can play a chess game against a computer without seeing or knowing how to use a computer.

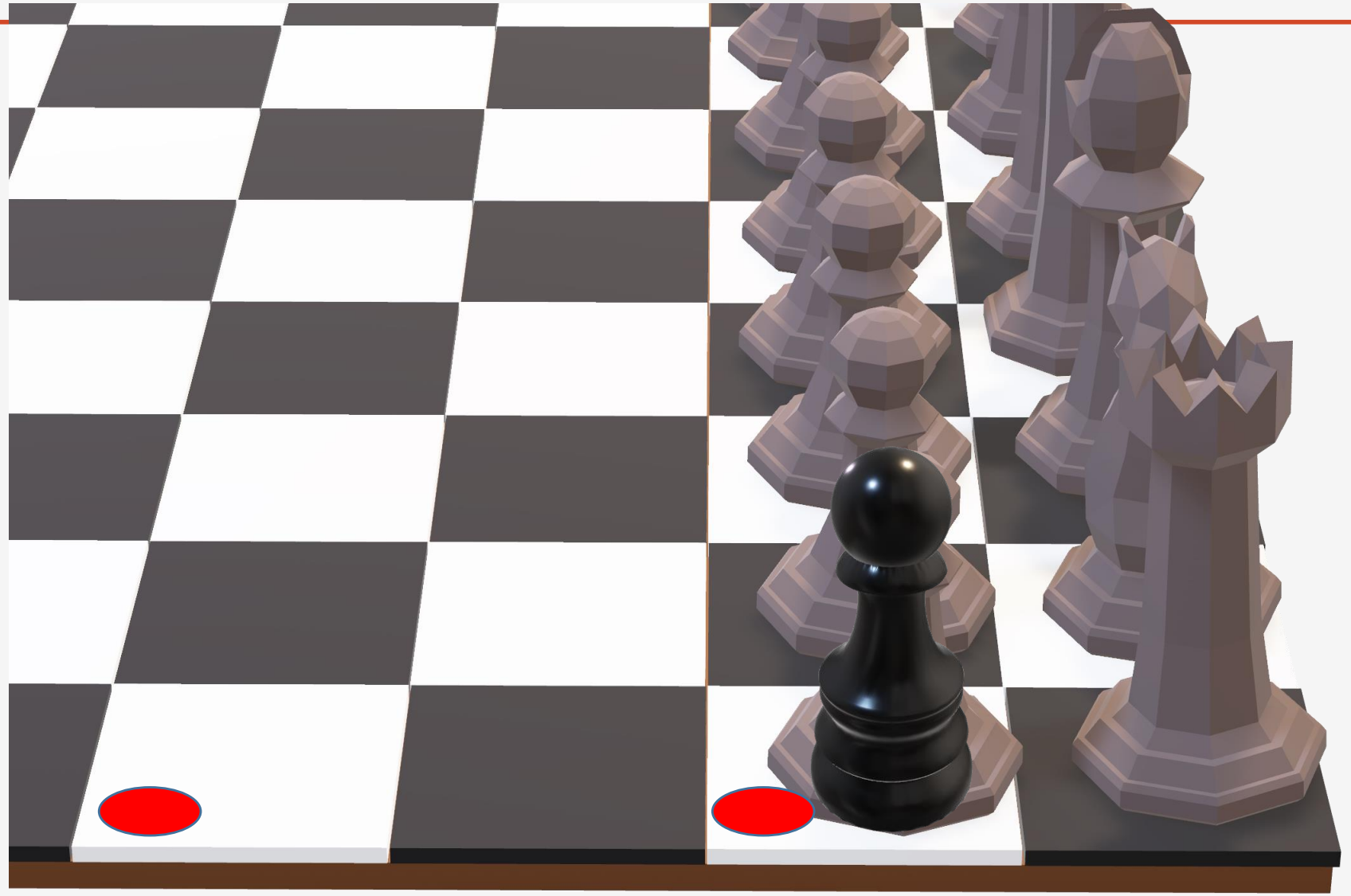
The Chess Board

Player lifts a figure

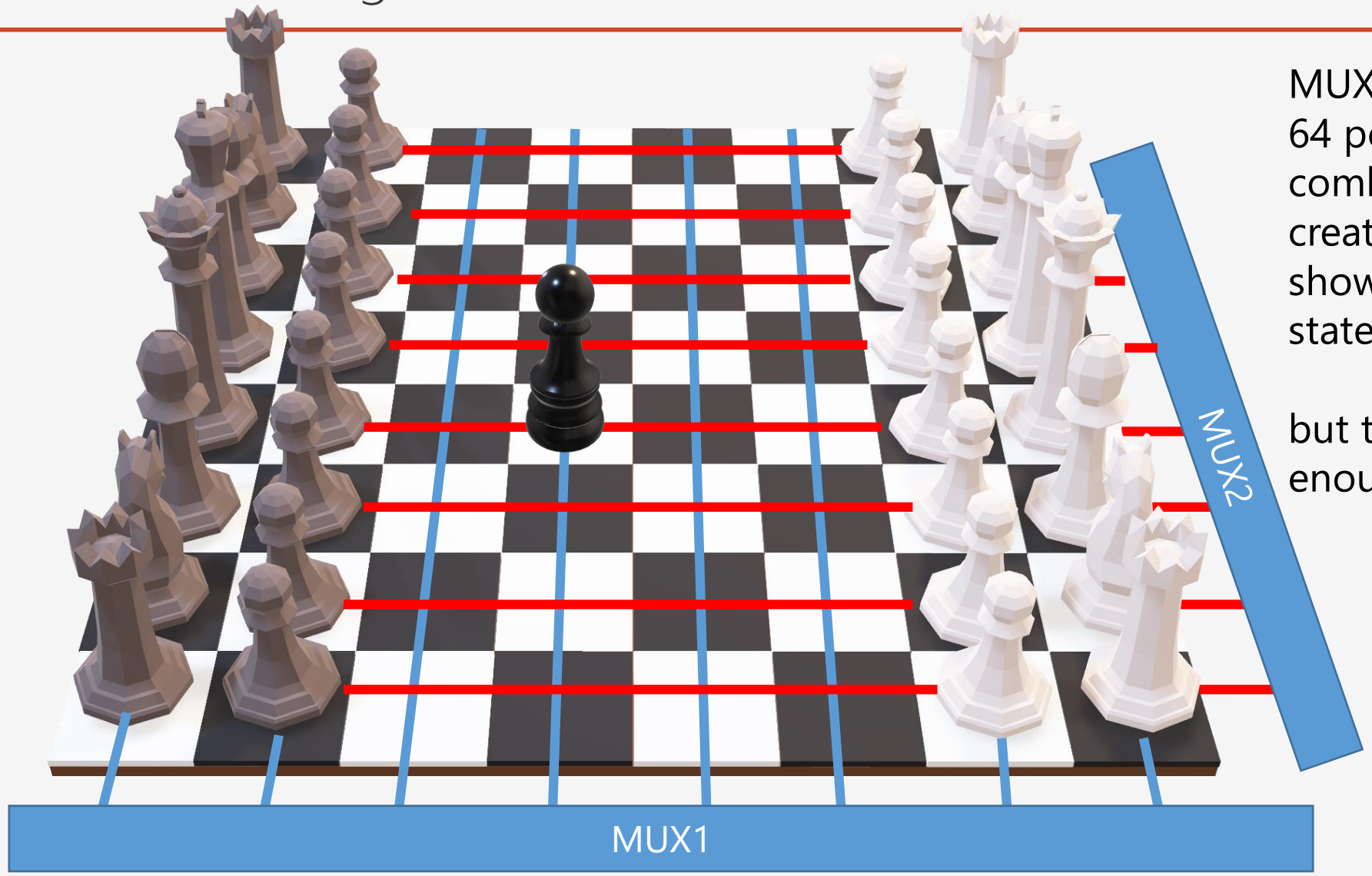


LEDs indicate legal positions

The Chess Board



The Chess Board – Figure detection



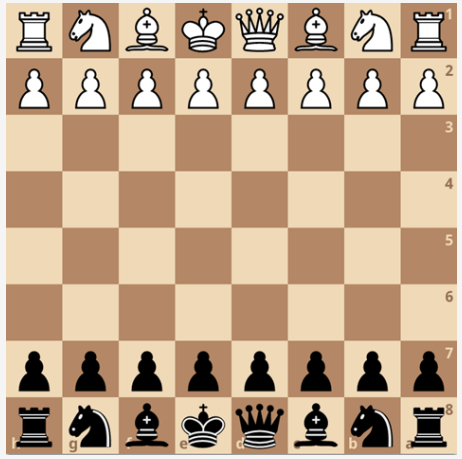
MUXes switch through 64 possible combinations and create a 64-bit string showing the board state

but that is not enough!

Chess Game FSM (Example binary)

Board_state [63:0] example progression

11111111
11111111
00000000
00000000
00000000
00000000
11111111
11111111



1**0**111111
11111111
00000000
00000000
00000000
00000000
11111111
11111111

10111111
11111111
00**1**00000
00000000
00000000
00000000
11111111
11111111

Chess Game FSM (Example decision tree)

Board States [63:0] example progression

1. There are exactly 32 pieces
2. All pieces occupy ranks 1,2,7,8

1. Orientation is white on top
2. Every piece type is now set (Queens on their respective colors)
3. from_to [11:0] should alternate between f3 and h3

1. Knight location updated
2. Send move to PC over UART

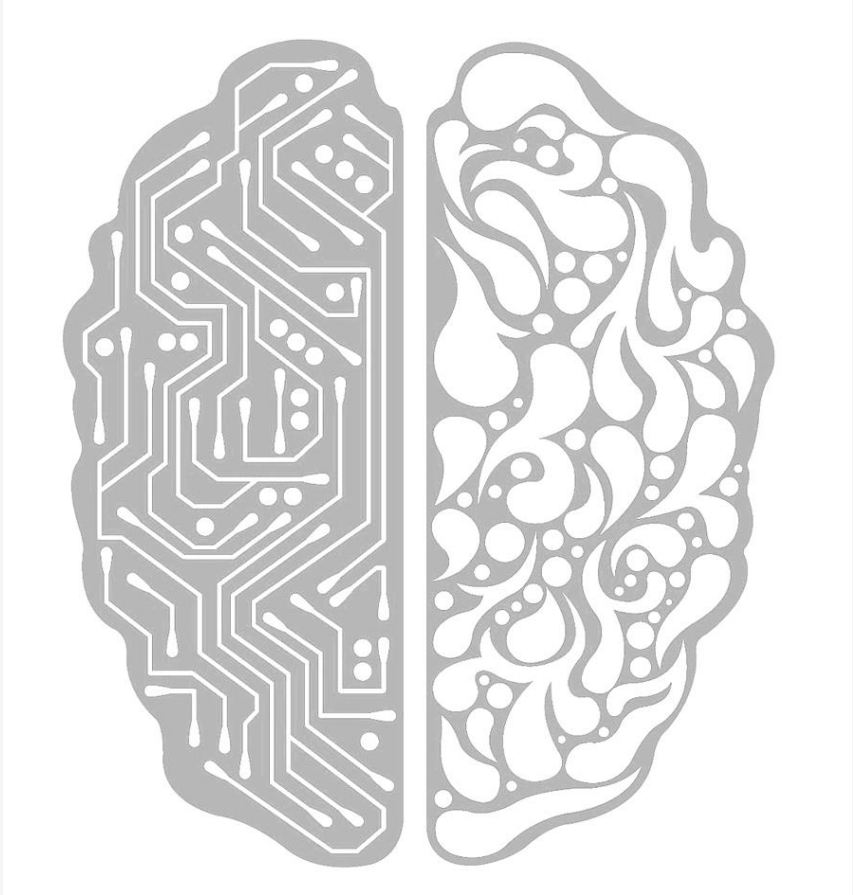
Chess Game FSM

- Inputs:
  - **Clk_in** - Standard 100mhz clock
  - **[63:0] board_state** - cleaned bitmask showing current state (piece / no piece) of each square
  - **Board_state_ready** - goes high when previous modules finish reading + cleaning
- Outputs:
  - **Show_pos** - controls when to turn on LEDs
  - **[11:0] from_to** - Abstractly controls which LEDs to turn on

Chess Game FSM (Challenges)

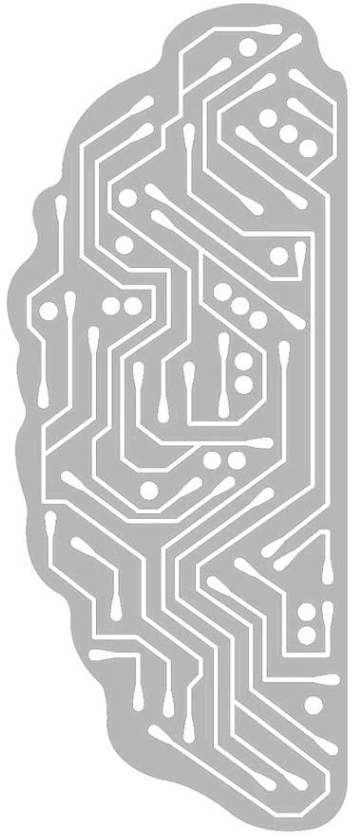
- Intricacies of move validity validation
- Evaluating possibility of capture when move order is broken
- Anything involving multiple pieces
  a. Capture
  b. Castling
  c. Piece promotion

Chess Board FSM ⟷ Python Chess AI



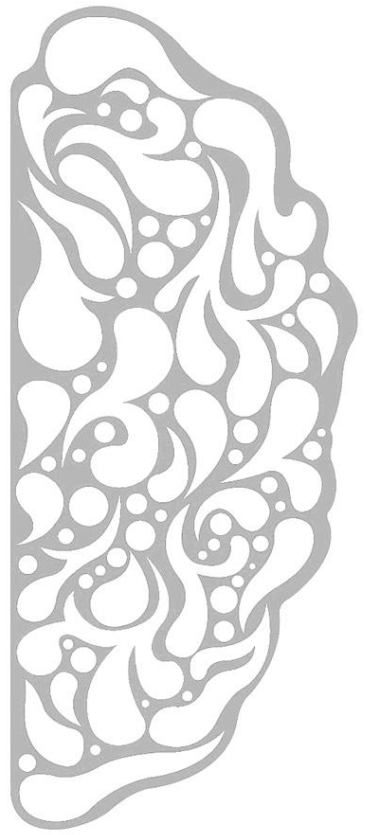
Communicating with python-chess

UART like Lab 2 (“g1f3”)

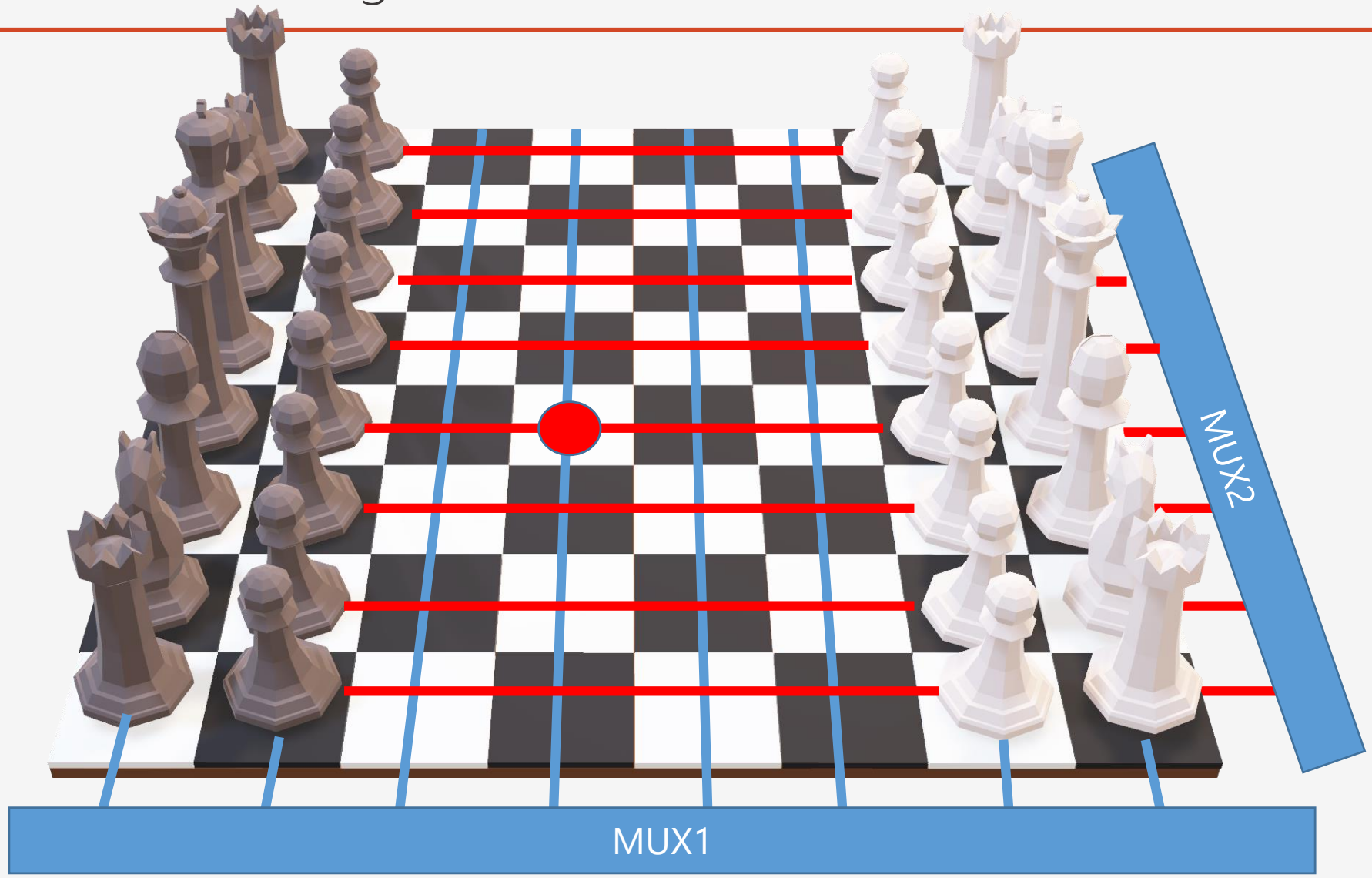


- A full piece movement has occurred
- Validity has been checked
- FSM has been updated
- Convert move to ascii (“g1f3”)

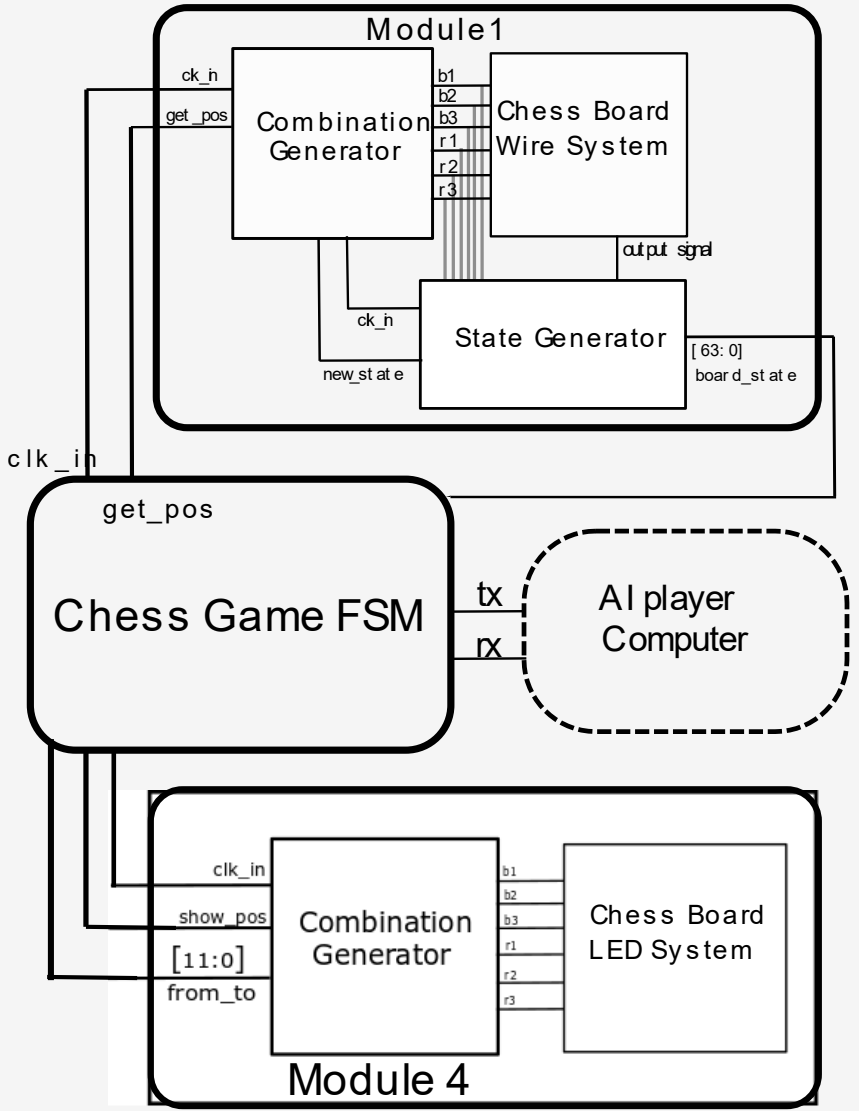
- Receive piece movement
- `board.push(chess.Move.from_uci("g1f3"))`
- `result = engine.play(board, chess.engine.Limit(time=0.1))`
- Send back piece movement



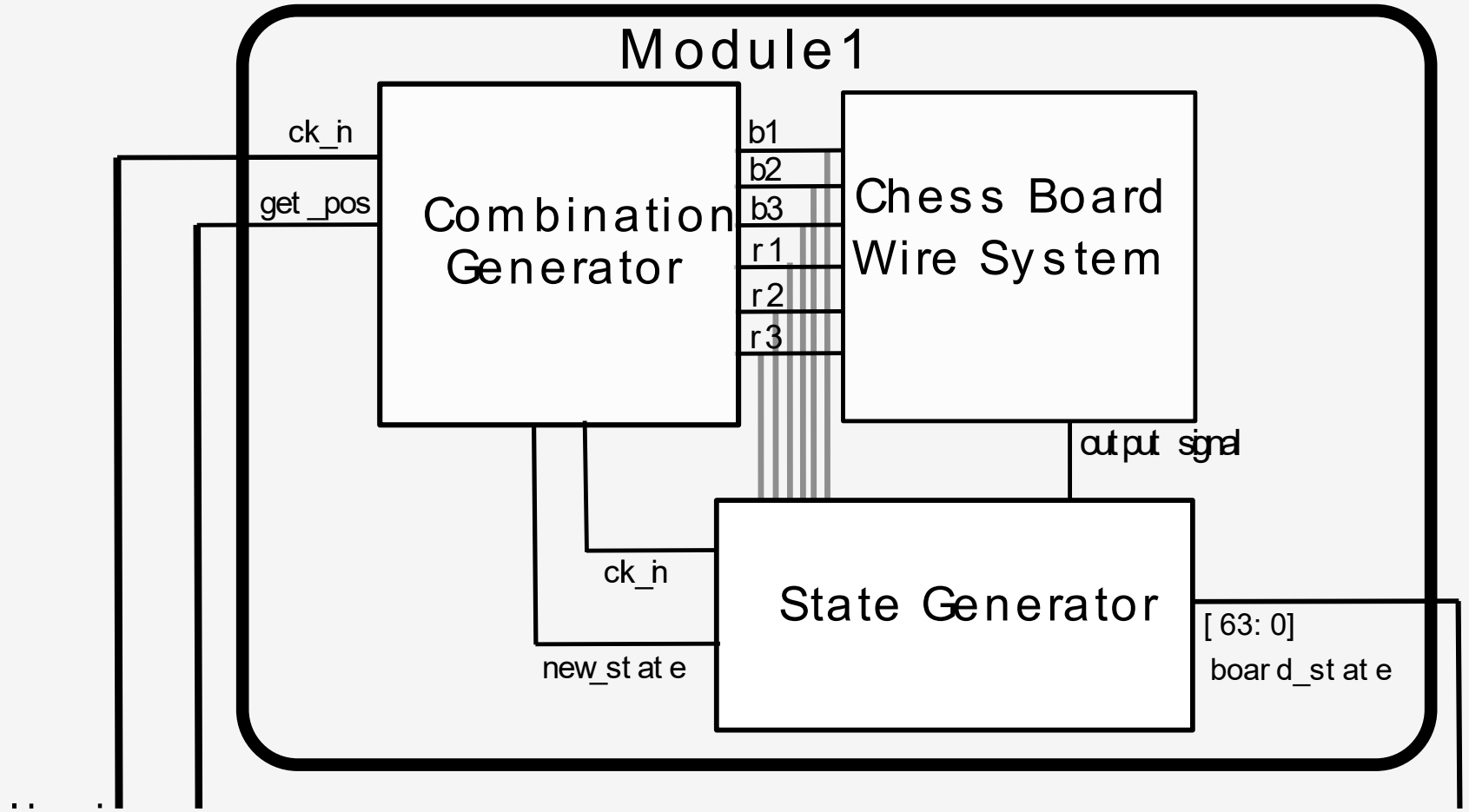
The Chess Board – Figure Movement Indication



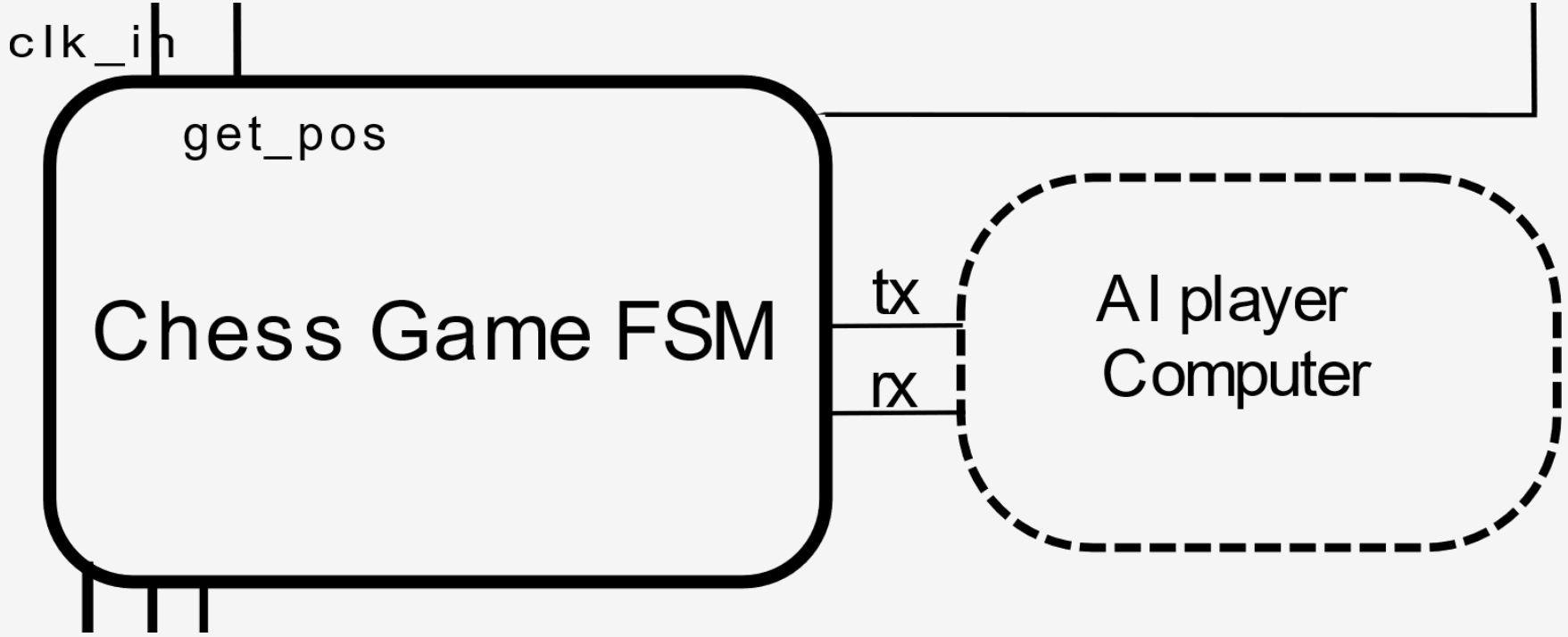
Block Diagram



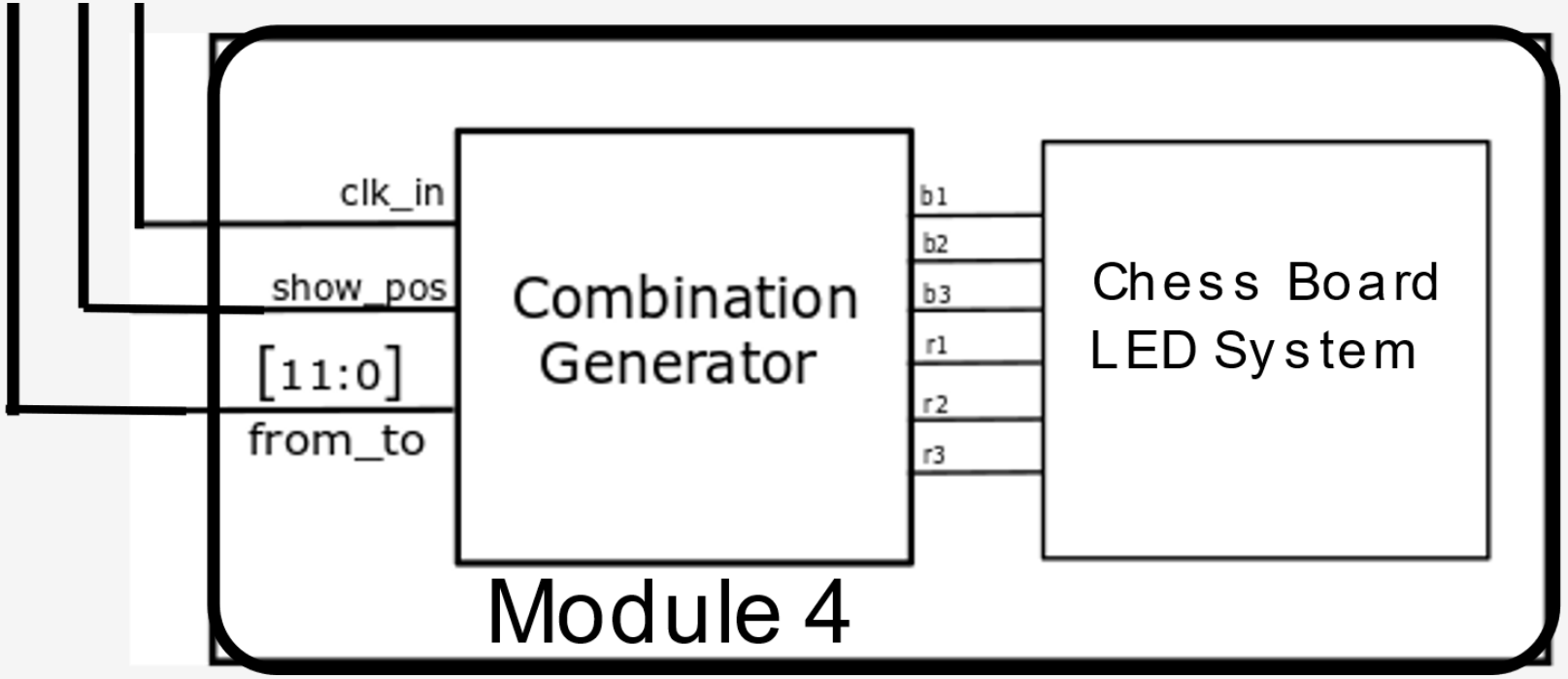
Block Diagram



Block Diagram



Block Diagram



Project Timeline and Workload

Week of	Bahrudin	Grayson
Nov 4	• Prepare the chess board hardware (LEDs, wire grid, mux connections)	• Design the initial FSM functionalities
Nov 11	• Finish the hardware. Work on Module 1 – detection of figures	• Implement an FSM that tracks figures given the occupied squares of the board.
Nov 18	• Work on Module 4 – Displaying of the movements of the opponent. • UART communication	• Add constraints to the movements to the FSM. • Convert the movements to appropriate encodings that will be sent to the computer.
Nov 25	Integration of the UART communication, Modules 1 and 4 and the game FSM	
Dec 2	Test the board, game logic, edge cases. Add on screen game state. (Add VGA output of the game progress)	