CSEE 4840: Chess Project

Group members: Hooman Khaloo, Hongchi Liu, and Pengfei Yan

Instructor: Prof. Stephen A. Edwards

Contents

Project Overview

What we can do now:

- PvP Mode
- PvE Mode
- Set Colors

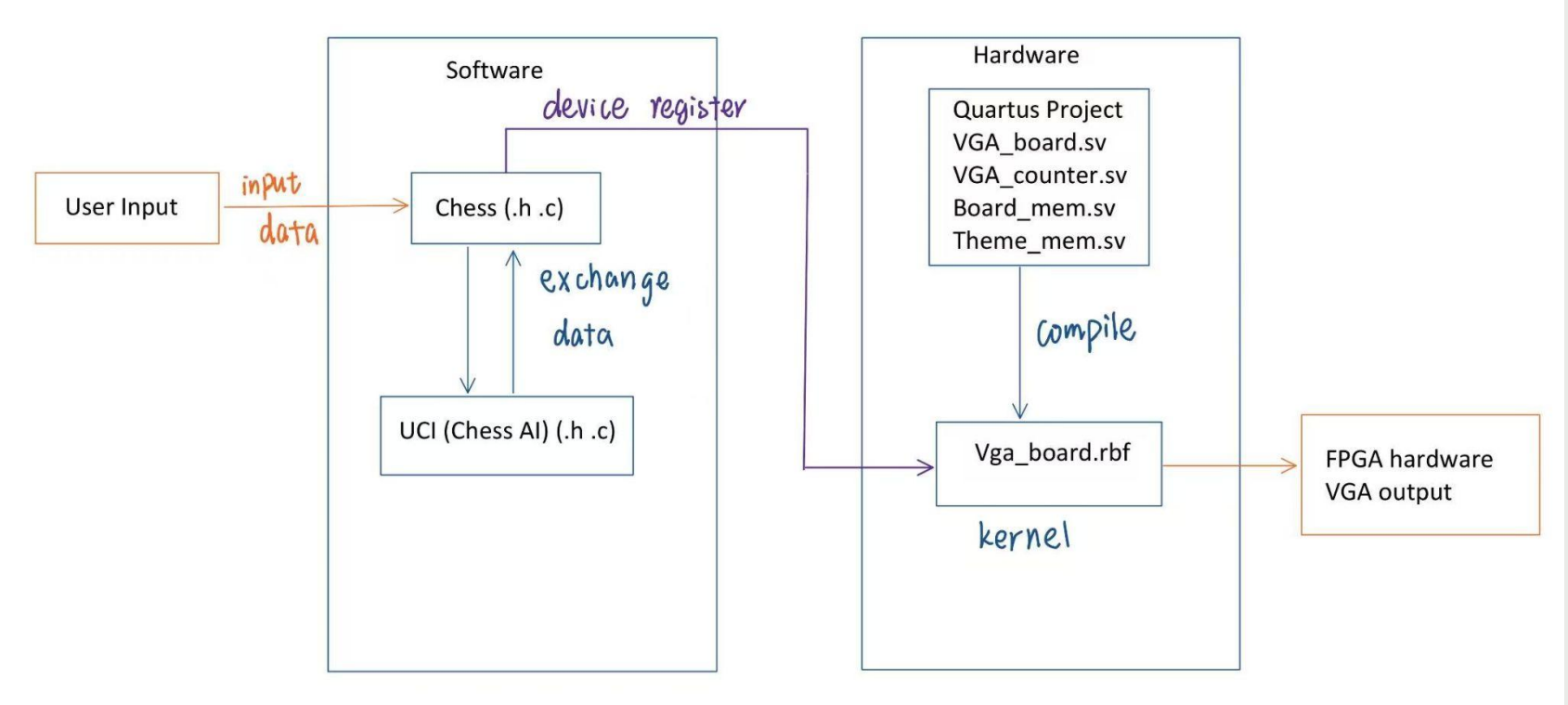
Input format:

- USB Keyboard via terminal

Output format:

- VGA display for chess board
- Terminal print for message

Block Diagram



Hardware

Vga_counters.sv
Vga_board.sv
theme_mem.sv
board_mem.sv

How to draw the board: Vga_counters.sv and Vga_board.sv

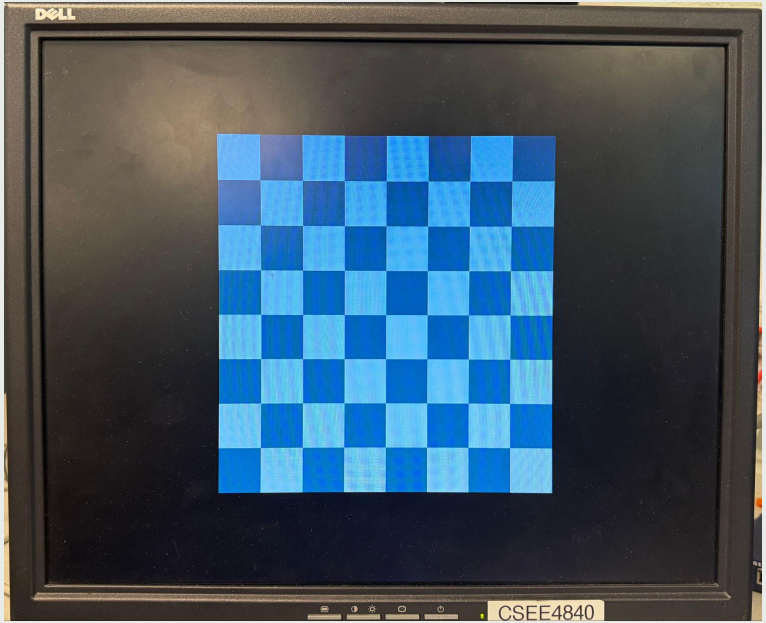
Resolution : 1024*768 60Hz
Clock: 50 MHz

logic [10:0] hcount;
logic [9:0] vcount;

x_box_no = (hcount - 256) / BOX_SIZE (64)
y_box_no = (vcount - 128) / BOX_SIZE (64)

x_parity = x_box_no % 2;
y_parity = y_box_no % 2;

Black or white box?
- x_parity XOR y_parity

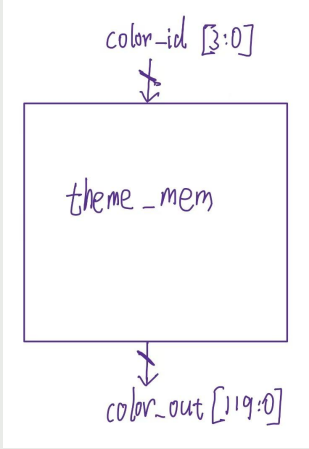


X XOR Y Table:

x/y	0	1
0	0	1
1	1	0

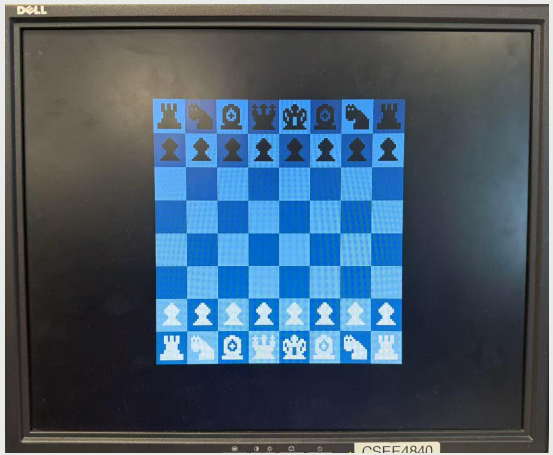
theme_mem.sv

We use 4 bits id to store 16 themes.
For each theme, we have 8 bit * 3 (RGB) * 5 (colors) -> 120 bits

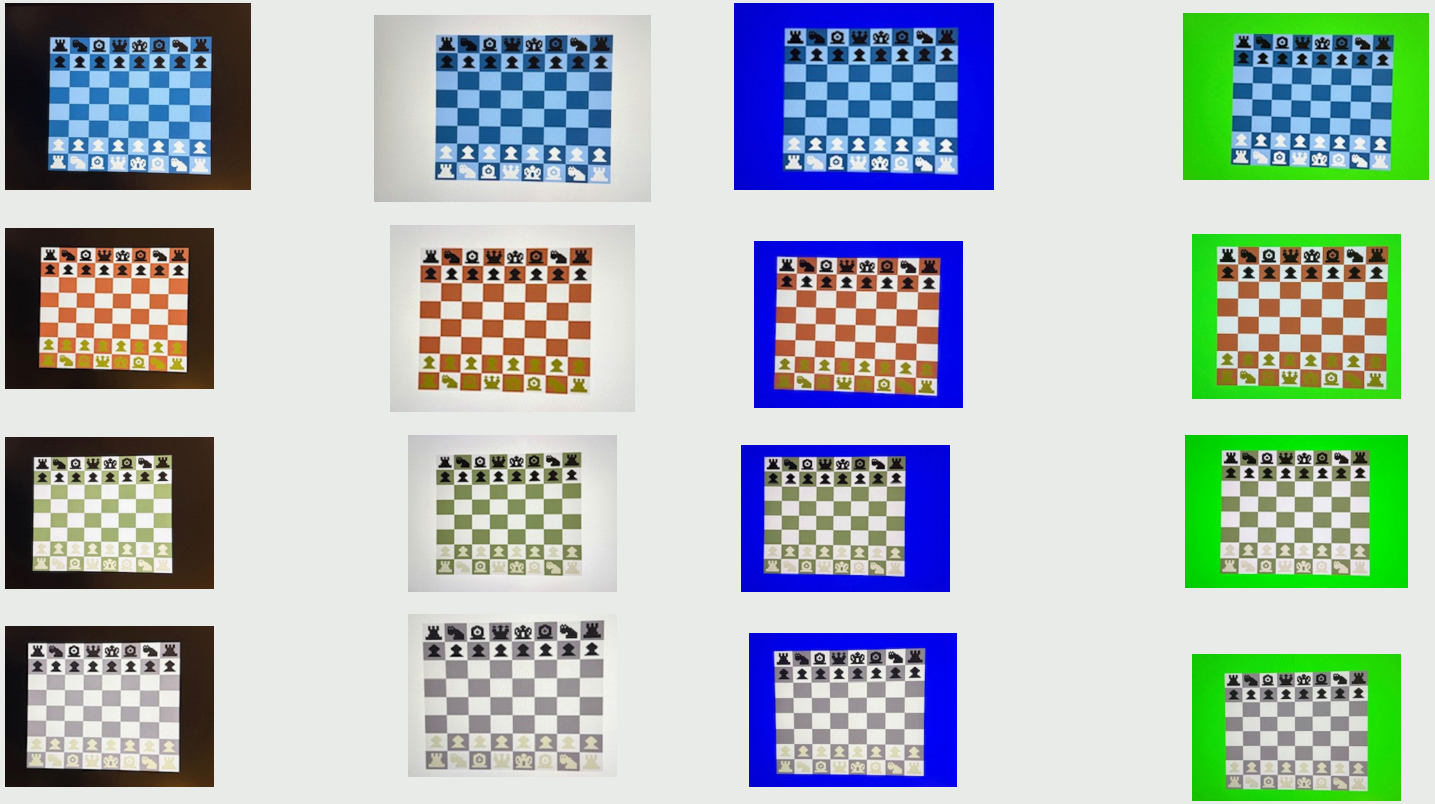


For example:

```
8'h00, 8'h00, 8'h00,    // Black piece: black
8'hFF, 8'hFF, 8'hFF,    // White piece: white
8'h29, 8'h64, 8'h95,    // Black box: dark blue
8'ha4, 8'hcc, 8'hfb,    // White box: light blue
8'h00, 8'h00, 8'h00     // Background: black
```

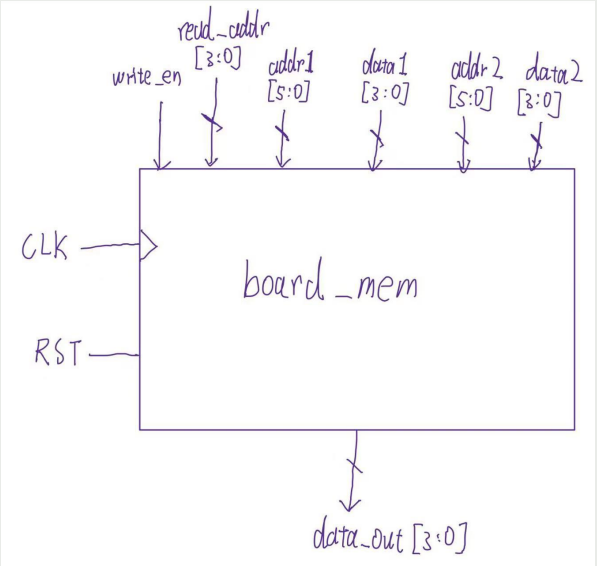


All 16 Color themes:



board_mem.sv

- Input/output: Figure 1
- read_addr and data_out -> **continuous** reading for vga counter
- 2 sets of writing ports
- 2 modes of writing:
  - addr1 = addr2 -> only write one piece
  - addr1 != addr2 -> write two pieces at one time
- Write data is 4 bits -> color and type (Figure 2)



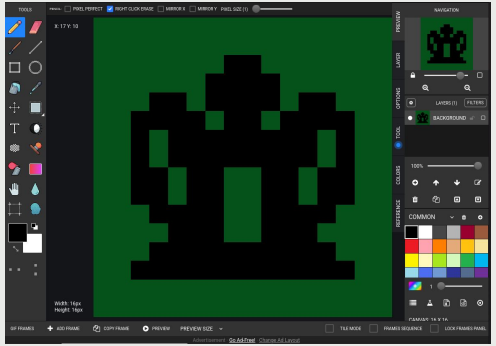

```
typedef enum {
    /*Type     # dec # bin   */
    EMPTY,  //    0    000
    PAWN,   //    1    001
    ROOK,   //    2    010
    KNIGHT, //    3    011
    BISHOP, //    4    100
    QUEEN,  //    5    101
    KING    //    6    110
} PieceType;
```

How to draw the pieces?

1. Use board mem to define color and type
2. Look at shape array (16*16 to save mem)
3. Calculate

```
local_x = (hcount - 256) % BOX_SIZE / 4
local_y = (vcount - 128) % BOX_SIZE / 4
```

4. Set black piece color or white piece color based on shape array and color



Pixel Drawing

```
king16[ 0] = 16'b0000000000000000;
king16[ 1] = 16'b0000000000000000;
king16[ 2] = 16'b0000000110000000;
king16[ 3] = 16'b0000001111000000;
king16[ 4] = 16'b0001101111011000;
king16[ 5] = 16'b0011110110111100;
king16[ 6] = 16'b0010111111110100;
king16[ 7] = 16'b0010111111110100;
king16[ 8] = 16'b0011011001101100;
king16[ 9] = 16'b0011011001101100;
king16[10] = 16'b0001111001111000;
king16[11] = 16'b0000111001110000;
king16[12] = 16'b0001111111111000;
king16[13] = 16'b0011111111111100;
king16[14] = 16'b0000000000000000;
king16[15] = 16'b0000000000000000;
```

Software

Chess.h
Chess.c

uci.h
uci.c

Data Structure

```
typedef enum {
    /*Type      # dec # bin   */
    EMPTY,  //    0    000
    PAWN,   //    1    001
    ROOK,   //    2    010
    KNIGHT, //    3    011
    BISHOP, //    4    100
    QUEEN,  //    5    101
    KING    //    6    110
} PieceType;
```

```
typedef enum {
    /*Type      # dec # bin   */
    NONE,   //    0    00
    WHITE,  //    1    01
    BLACK   //    2    10
} Color;
```

```
typedef struct {
    PieceType type;
    Color color;
} Piece;
```

/* Use a global 8*8 array of Pieces to present board */
Piece board[8][8];

- Easy to convert to what hw needs.

Important Functions

```
/* Initialize the board when game start */
void init_board();

/* Check whether a movement is valid */
bool check_move(playerColor, start_x, start_y, end_x, end_y);

/* Start a game in person vs person mode */
void pvp_mode();

/* Start a game in person vs AI mode */
void pve_mode();

/* Main is a while loop to let user choose game mode */
Int main();
```

Rules Check:

bool check_move(playerColor, start_x, start_y, end_x, end_y)

1. Use input coordinates to find startPiece and endPiece in global array valuable.
2. startPiece cannot be empty
3. Player's Color must match startPiece's color
4. endPiece's color cannot be Player's Color (Also cover the case of no move)
5. Use startPiece's type to keep check different rules

Rules Check:

Pawn: Move - check y and dy = 1 or 2. Diagonal capture dx = 1, dy = 1.

Rook: dx = 0 or dy = 0. No other piece on the path.

Knight: dx=1, dy=2 or dx=2, dy=1.

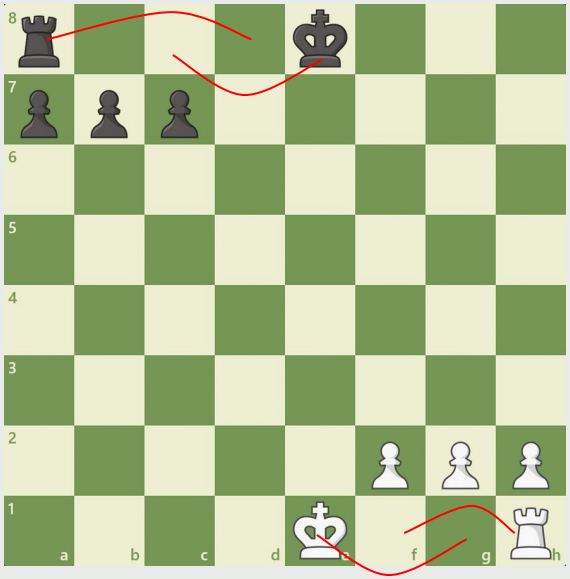
Bishop: dx = dy. No other piece on the path.

Queen: Rook or Bishop.

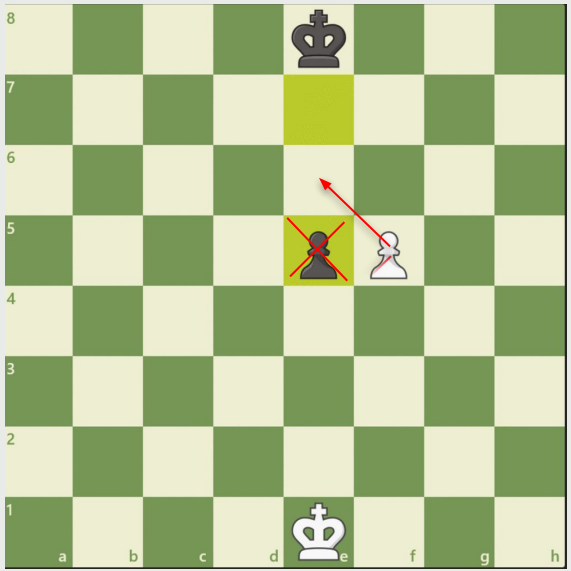
King: dx <= 1 and dy <=1.

Special Rules:

1. Pawn Promotion
2. Castling



3. En Passant Capture



```
/* Check castling */
bool whiteKingMoved = false;
bool blackKingMoved = false;
bool whiteLeftRookMoved = false;
bool whiteRightRookMoved = false;
bool blackLeftRookMoved = false;
bool blackRightRookMoved = false;

/* check en passant */
int lastDoublePushRow = -1;
int lastDoublePushCol = -1;
```

Source: https://www.chess.com/terms/special-chess-moves

How chess.c exchange data with uci.c

History buffer: char moves[2048]

One move: “e2e4\0” -> String with length 5.

Moves become {e2e4\0 f7f5\0 ...}

Register Map and Interface

Register Map:
Addr [3:0]
Writedata [7:0]

addr	writedata[7]	writedata[6]	writedata[5]	writedata[4]	writedata[3]	writedata[2]	writedata[1]	writedata[0]
0	-	-	-	-	color id			
1	Start X				End X		Start Y[2:1]	
2	Start Y[0]	End Y			Color and Type			
3	Done Movement							

Software and Function for Sending Data to Hardware

```
uint8_t reg1 = ((start_x & 0x7) << 5) | ((end_x & 0x7) << 2) | ((start_y >> 1) & 0x3);
uint8_t reg2 = ((start_y & 0x1) << 7) | ((end_y & 0x7) << 4) | (e_p & 0xF);


vga_board_regs[1] = reg1;
usleep(2000);  // Wait for next clock
vga_board_regs[2] = reg2;
usleep(2000);  // Wait for next clock
vga_board_regs[3] = 1;  // commit
usleep(2000);
```

Thanks for Your Attention.