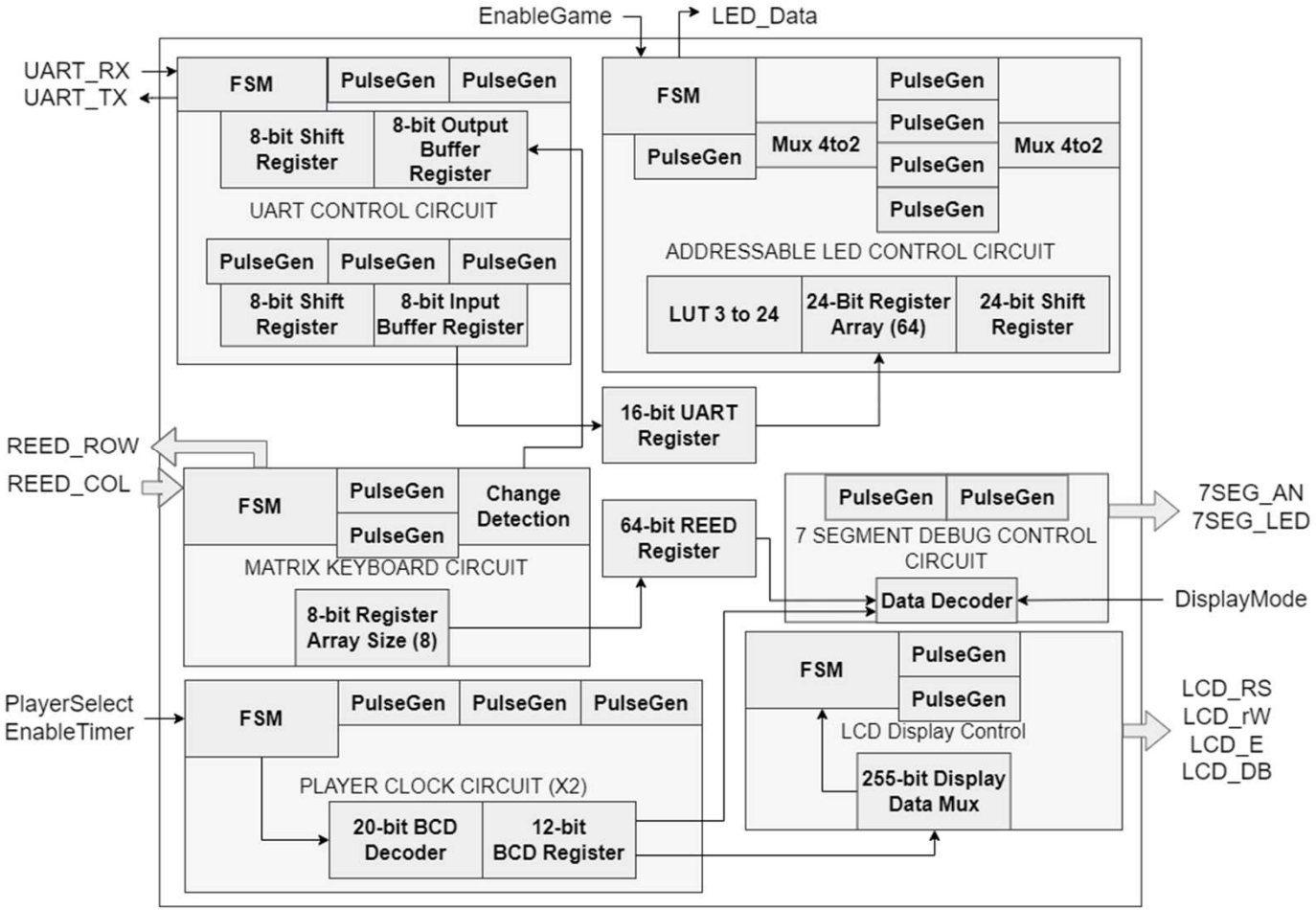
The Chess Mate

Produced by : Zachary Jump, Aidan Gallagher, Bradley Taylor, and Avie Sachdeva

Introduction

- Teaches players how to play chess while elevating the overall gaming experience
- Uses real-time piece tracking, intuitively mapping user-selected moves
- Will help guide and teach chess players of any level to learn and understand certain strategies
- Involves two boards to handle separate functions
  - Dragon12 using HCS12 microprocessor to handle the design
  - Nexys Artix-7 FPGA to handle the display

Block Diagram



Receiving

- The nexys will receive data from the Dragon HCS12 that mimics the structure of a chess board.
- From here the Nexys will be able to display certain things needed on the physical board.
- The Nexys will update the dragon with new data.

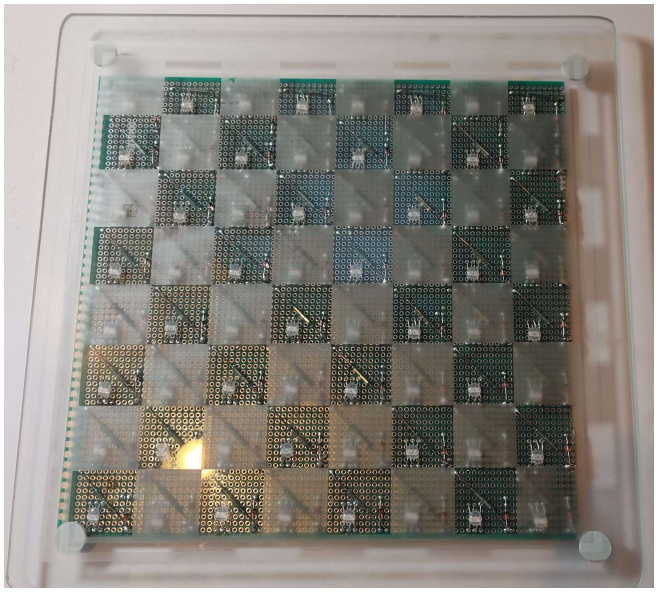
```
void main(void) {

 startup();
 Game_Over = 0;

 for(;;) {
 CaptureData();        //piece is picked up

 if ((temp_reg & piece_mask)== 3){
   RookMoves(p,q);
 }
 if ((temp_reg & piece_mask)== 4){
   QueenMoves(p,q);
 }
 if ((temp_reg & piece_mask)== 2){
   BishopMoves(p,q);
 }
 if ((temp_reg & piece_mask)== 1){
    KnightMoves(p,q);
 }
 if ((temp_reg & piece_mask)== 5){
     KingMoves(p,q);
 }
 if ((temp_reg & piece_mask)== 0){
   PawnMoves(p,q);
 }
 SendData();    //available moves are sent to nexys
 MSDelay(100);
 UpdateBoardCapture();    //piece is placed, start from the top
 Check(c,v);

if(Game_Over == 1){
  for(;;){    //play animation celebration forever or until system r

  for(k = 0; k <8; k++){

     for(l =0; l<8; l++){
      temp_chess_board[k][l] = temp_chess_board[k][l] & red_mask;
```

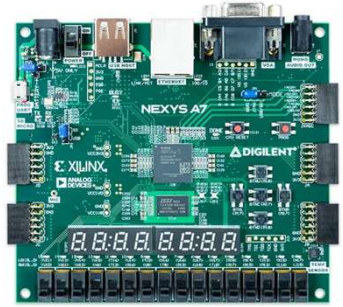
7	a7	b7	c7	d7	e7	f7	g7	h7
6	a6	b6	c6	d6	e6	f6	g6	h6
5	a5	b5	c5	d5	e5	f5	g5	h5
4	a4	b4	c4	d4	e4	f4	g4	h4
3	a3	b3	c3	d3	e3	f3	g3	h3
2	a2	b2	c2	d2	e2	f2	g2	h2

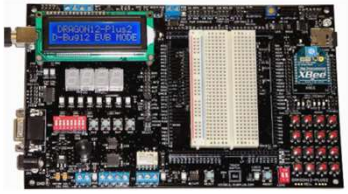
	K=0	K=1	K=2	K=3	K=4	K=5	K=6	K=7
l=0	Rooke	Knight	Bishop	King	Queen	Bishop	Knight	Rooke
l=1	Pawn	Pawn	Pawn	Pawn	Pawn	Pawn	Pawn	Pawn
l=2								
l=3								
l=4								
l=5								
l=6	Pawn	Pawn	Pawn	Pawn	Pawn	Pawn	Pawn	Pawn

Physical Hardware

- Dragon HCS12
- Nexys Artix-7
- Glass chess board



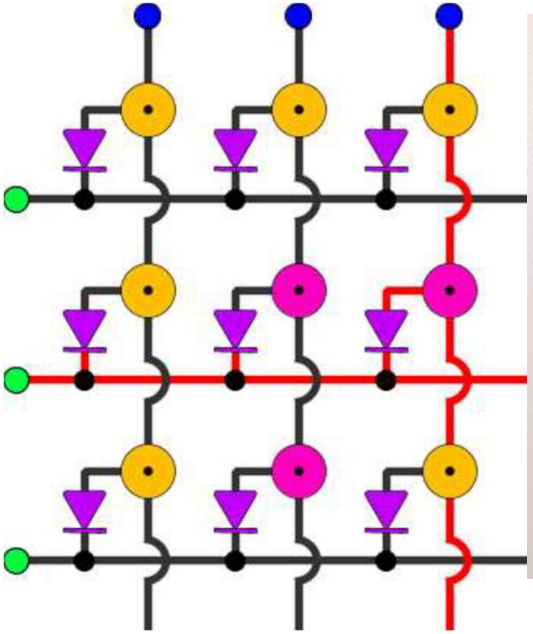


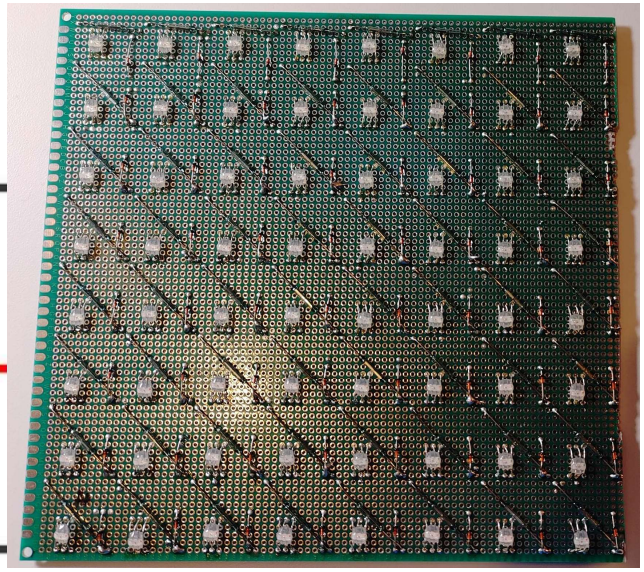


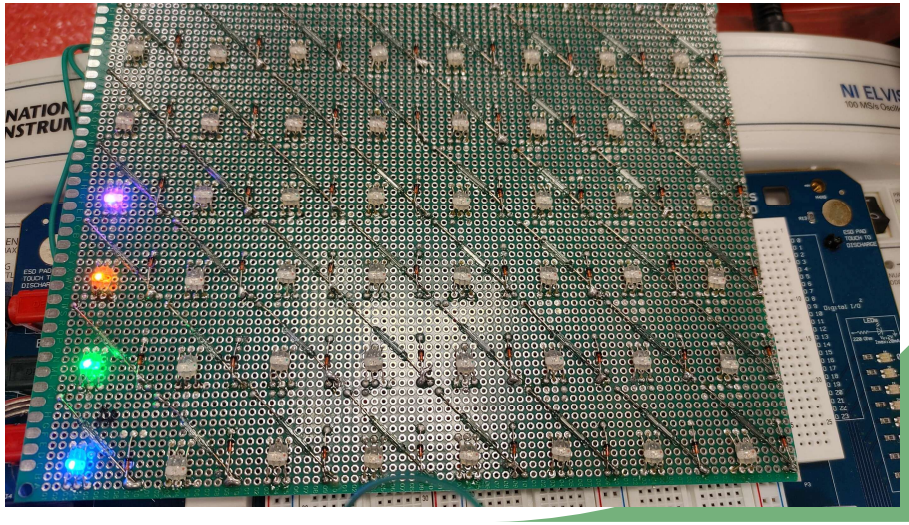
Electrical System

Circuitry Includes:

- ✓ Reed switches
  - Magnetic switches
- ✓ Diodes
  - To allow multiplexed matrix input data
- ✓ Addressable RGB LED strip
  - To address specific squares

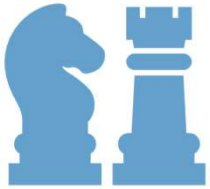




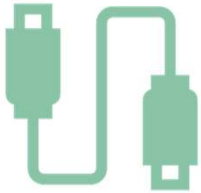


Functions involved

- Nexys-A7-100T-Final.xdc
- tbFinal.vhd
- UART_FSM.vhd
- UART_Top.vhd
- Clock_FSM.vhd
- Clock_Top.vhd
- Display7Seg64.vhd
- FinalTop.vhd
- LCD_FSM.vhd
- LCD_Top.vhd
- LED_FSM.vhd
- LED_Top.vhd
- matrixKey_FSM.vhd
- matrixKey_Top.vhd



Sending, receive, and updating data regarding chess pieces



These functions control all displays revolving around the LED's

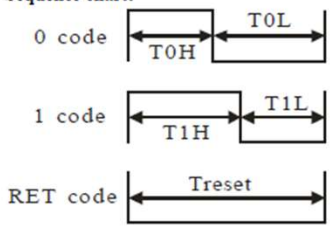
LED FSM

- Components:
  - my_genpulse_sclr
  - my_pashiftreg
  - counterModN
- Outputs:
  - y, idxZ, rstZ, nextBit
- Purpose:
  - Loops through 24 bits necessary for specific color needed on LED array.
  - FSM starts when signal is received from the Dragon Board.

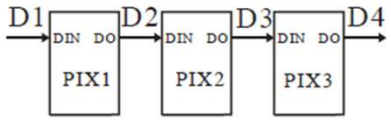
Data transfer time(TH+TL=1.25μs±600ns)

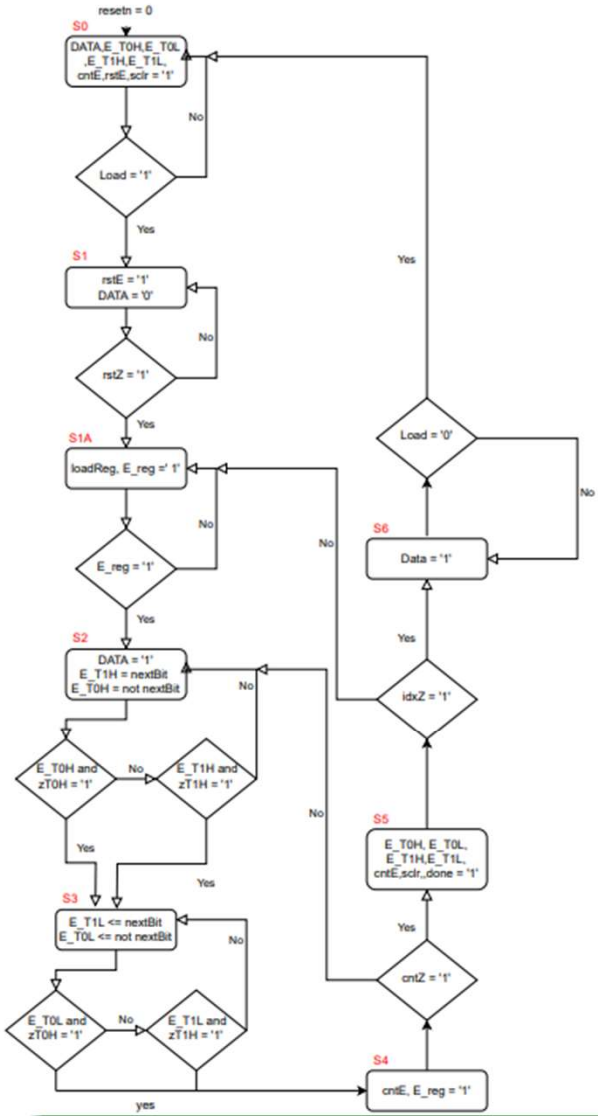
T0H	0 code ,high voltage time	0.35us	±150ns
T1H	1 code ,high voltage time	0.7us	±150ns
T0L	0 code , low voltage time	0.8us	±150ns
T1L	1 code ,low voltage time	0.6us	±150ns
RES	low voltage time	Above 50μs	

Sequence chart:



Cascade method:





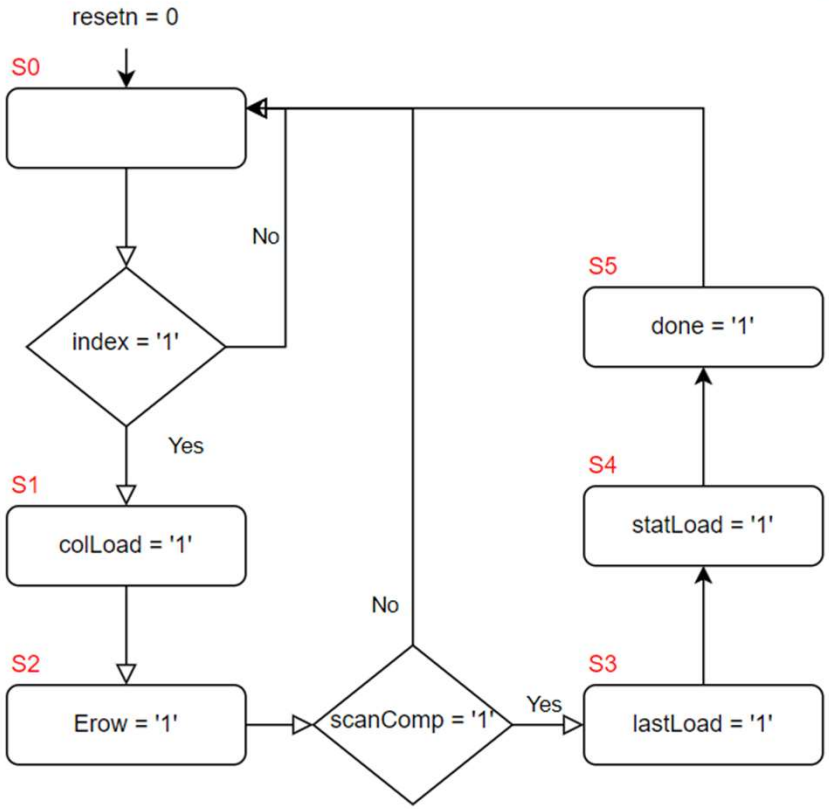

```
type state is (S0, S1, S1A, S2, S3, S4, S5, S6);
signal y: state := S0;
signal sclr: std_logic := '0';
signal E_T0H, E_T0L, E_T1H, E_T1L, zT0H, zT0L, zT1H, zT1L: std_logic := '0';
signal loadReg, E_reg, nextBit, rstZ, cntE, cntZ, rstE: std_logic := '0';
```

MATRIX FSM

- Outputs:
  - Y
- States & Signals

```
type state is (S0, S1, S2, S3, S4, S5);
signal y: state := S0;
```

- Purpose:
  - Two counters, one to delay while the other is a row counter.
  - Continuous loop. Turns on a column, applies power and reads input row by row.



Test Bench

- Setup to test the vhdl design that we have implemented
- STIM: a loop involving RXdata that we predefined as a string

```
ARCHITECTURE behavior OF tbfinal IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT FinalTop is
    Port ( clock : in STD_LOGIC;
           resetn : in STD_LOGIC;
           UART_RX : in STD_LOGIC;
           UART_TX : out STD_LOGIC;
           REED_COL : in STD_LOGIC_VECTOR (7 downto 0);
           REED_ROW : out STD_LOGIC_VECTOR (7 downto 0);
           LED_7seg : out STD_LOGIC_VECTOR (7 downto 0);
           AN_7seg : out STD_LOGIC_VECTOR (7 downto 0);
           LED_data : out STD_LOGIC;
           LCD_RS : out STD_LOGIC;
           LCD_rW : out STD_LOGIC;
           LCD_E : out STD_LOGIC;
           LCD_DB : out STD_LOGIC_VECTOR (7 downto 0));
        END COMPONENT;

   --Inputs
   signal UART_RX : std_logic := '1';
   signal REED_COL : std_logic_vector (7 downto 0) := x"01";
   signal resetn : std_logic := '0';
   signal clock : std_logic := '0';

        --Outputs
   signal UART_TX : std_logic;
   signal REED_ROW : std_logic_vector (7 downto 0);
   signal LED_7seg : std_logic_vector (7 downto 0);
   signal AN_7seg : std_logic_vector (7 downto 0);
   signal LED_data : std_logic;
   signal LCD_RS : std_logic;
   signal LCD_rW : std_logic;
   signal LCD_E : std_logic;
   signal LCD_DB : std_logic_vector (7 downto 0);

   -- Clock period definitions
   constant T : time := 10 ns;
```

```
   signal RXdata : std_logic_vector (1 to 44) := "0001110011"&"0000000001"&"1111"&"0001110011"&"0100000001";

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: FinalTop PORT MAP (
          clock => clock,
          resetn => resetn,
          UART_RX => UART_RX,
          UART_TX => UART_TX,
          REED_COL => REED_COL,
          REED_ROW => REED_ROW,
          LED_7seg => LED_7seg,
          AN_7seg => AN_7seg,
          LED_data => LED_data,
          LCD_RS => LCD_RS,
          LCD_rW => LCD_rW,
          LCD_E => LCD_E,
          LCD_DB=> LCD_DB
          );

   -- Clock process definitions
   clock_process :process
   begin
                clock <= '1'; wait for T/2;
                clock <= '0'; wait for T/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 10 ns.
      wait for 15 ns;
      resetn <= '1';

    -- insert stimulus here
        wait for 1 ms;
        for i in 1 to 44 loop
                UART_RX <= RXdata(i);
                wait for 104167 ns;
        end loop;
        wait for 1 ms;
        wait;
        end process;
END;
```

Possible issues

Code being written by multiple authors

Two boards sending and receiving data

Conclusion

- The project shows how we can implement VHDL skills into a physical project.
- Learned critical problem solving skills when debugging.
- The VHDL side of this project is complete however there are some improvements we would like to make in the future regarding the embedded C data and physical appearance.

Thank you for your attention
Any Questions?