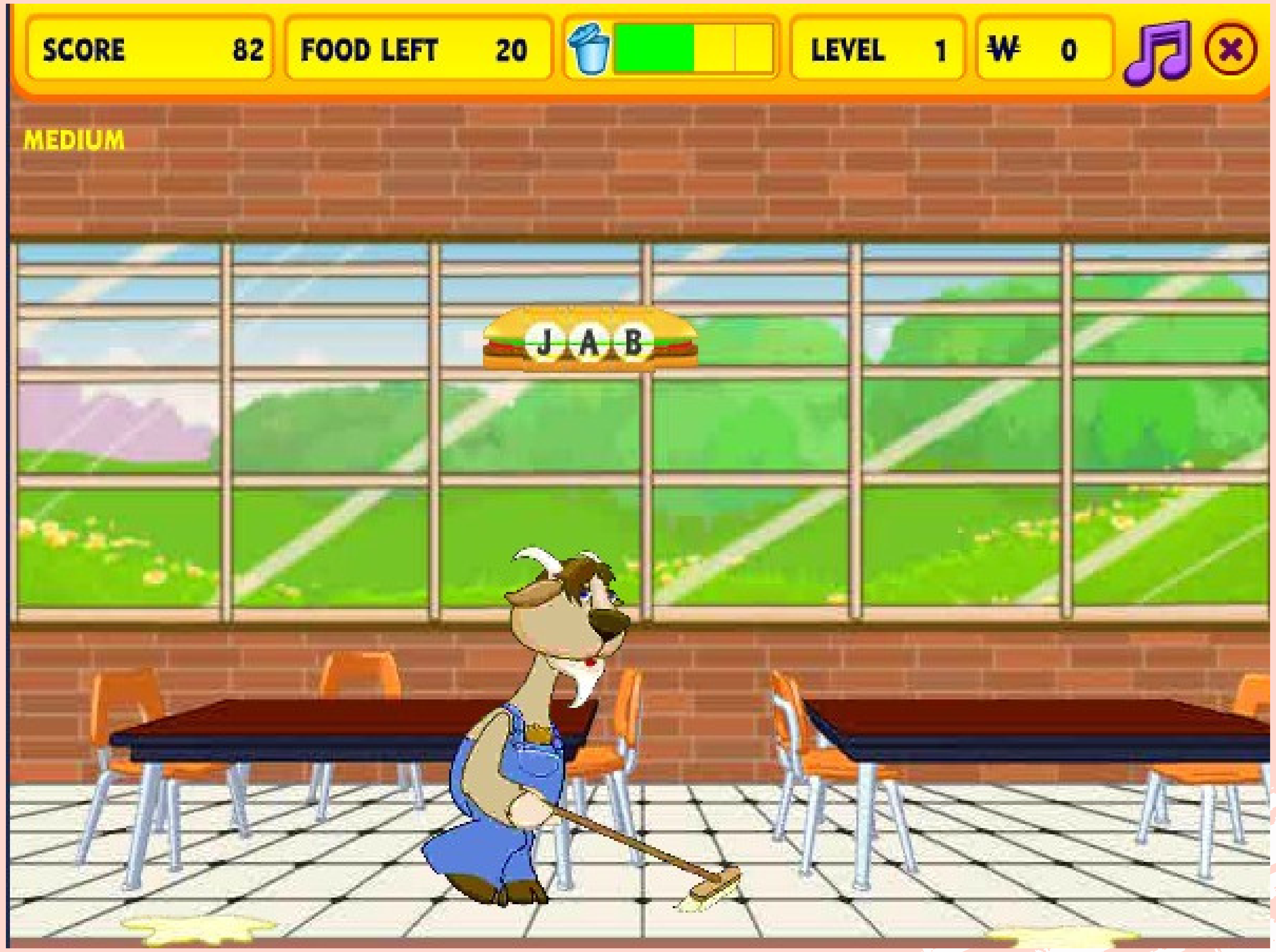
SNOWFALL

Team 9: The Big Macs

Luca Guidi, Jami Huang, Bryan Jaimes, Nicole Kwon

Goal/Motivation

- Our goal is to create a game that builds on our previous experience with the FPGA
  - inspired by Brick Breaker and Webkinz game "Lunch Letters"
- The player will control a paddle and catch falling blocks that are randomly generated
- For a real-life application, our design can be implemented as a fully fleshed-out game
  - Collision detection
  - Randomization of numbers



FUNCTIONALITY

Keyboard

user input

- Enter key to start game, space bar to stop paddle, and backspace to restart game after it ends
- Left and right arrow keys to move paddle

VGA Display

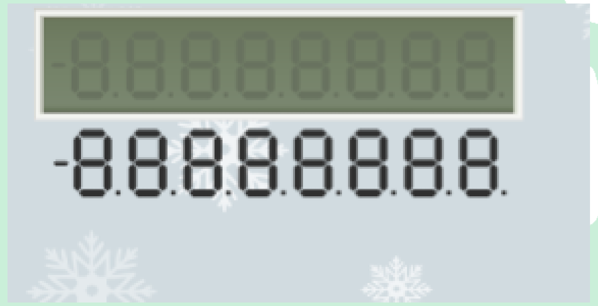
- Start screen, game (2 levels), end screen
- If paddle hits a bomb, the player loses a point
- The game is over when the timer is up

🟩 Paddle

⬜ Snow

🟥 Bomb

FPGA Board

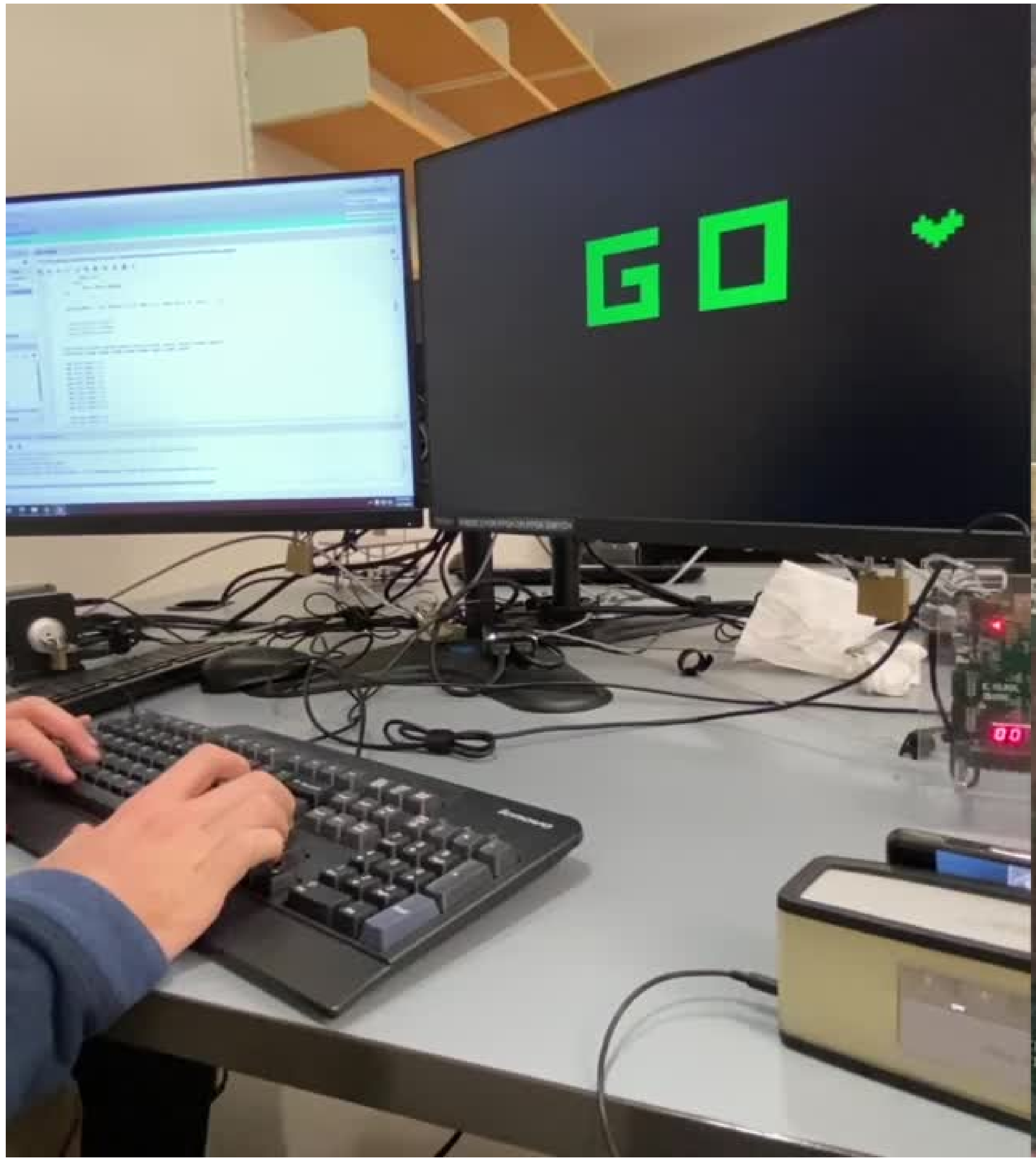


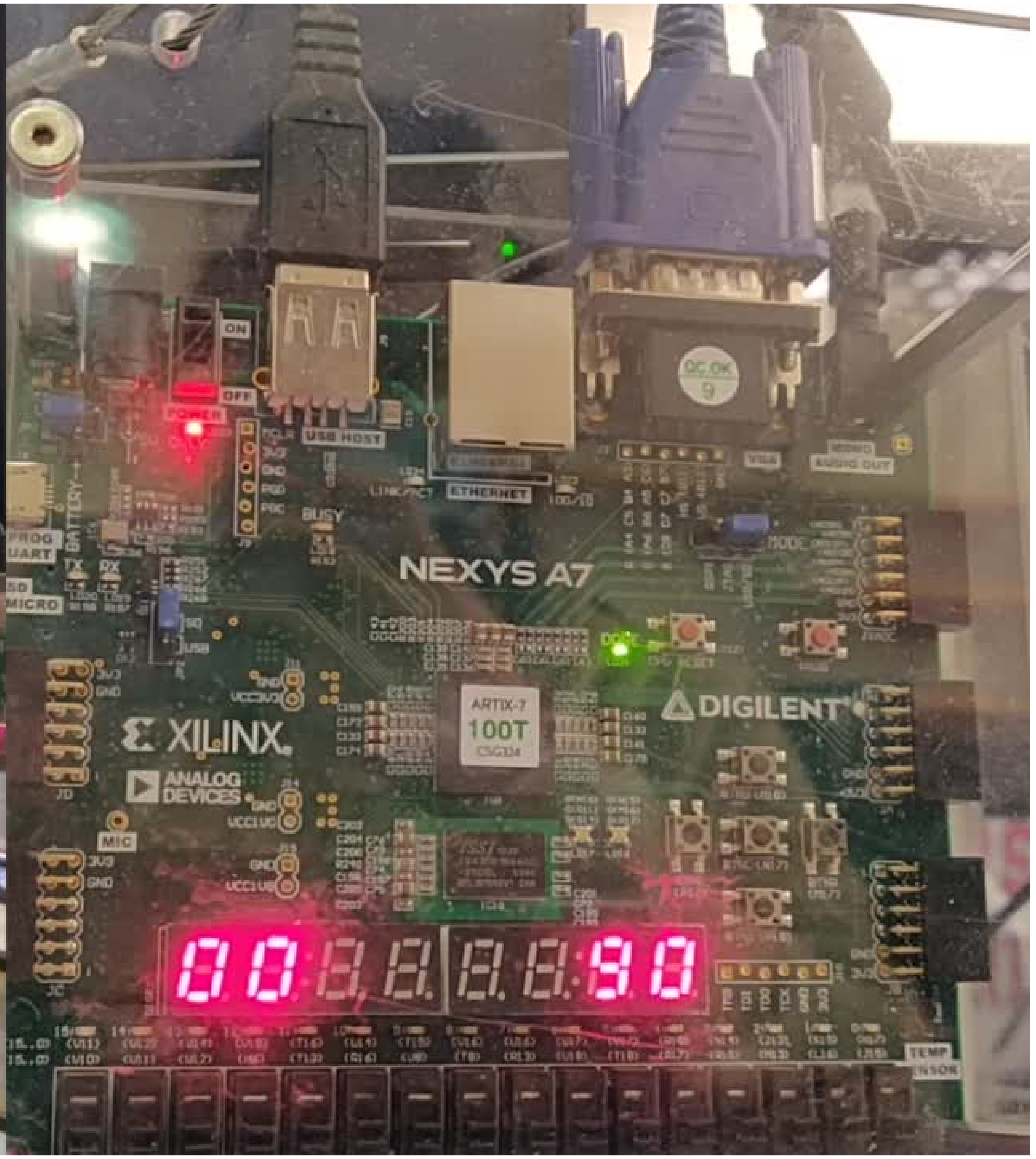
- Shows score with 7-segment display
- The score increments every time a "snow" block collides with the paddle

Speaker

Plays "Jingle Bells" as background music

DEMONSTRATION VIDEO

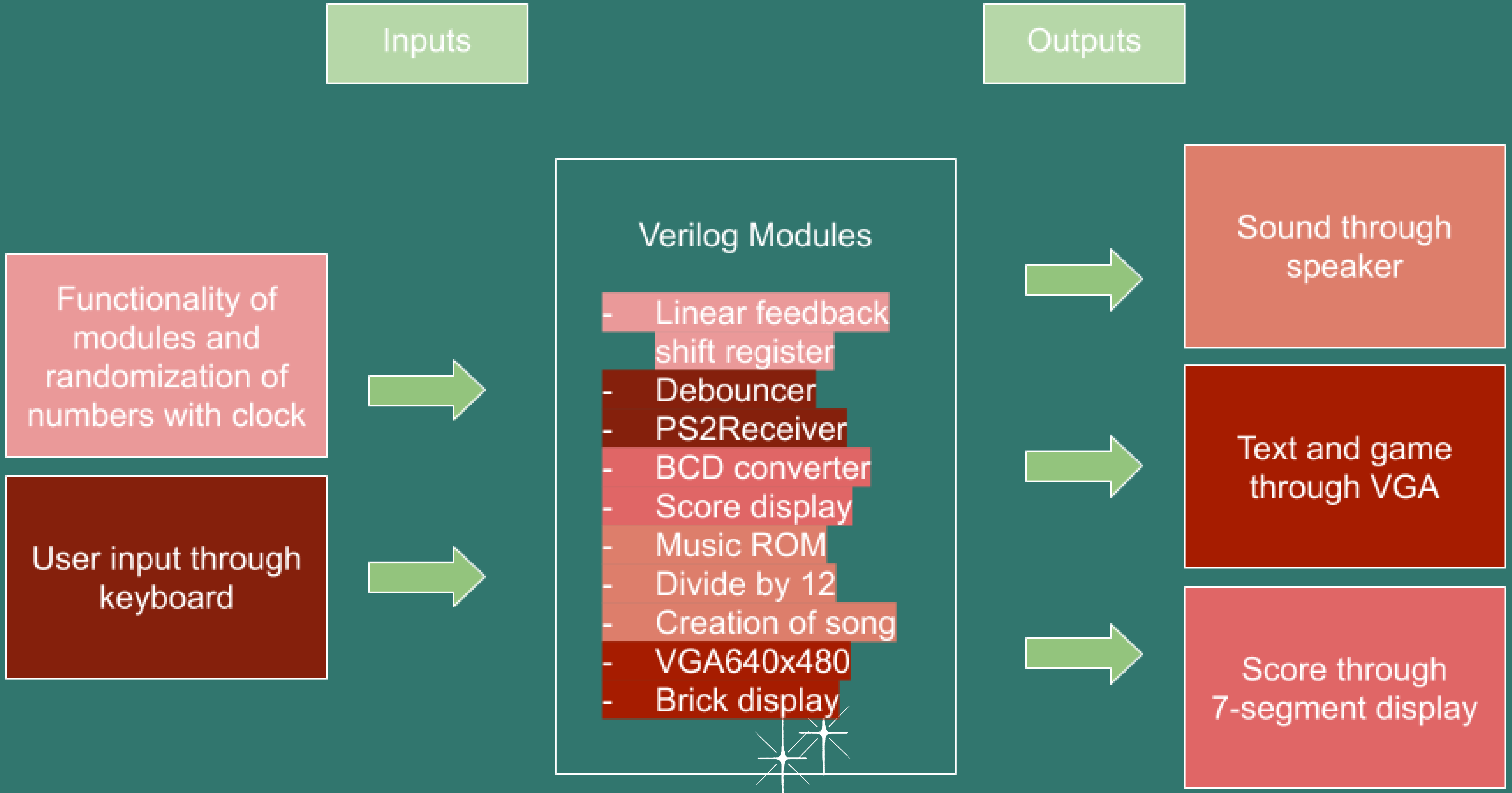




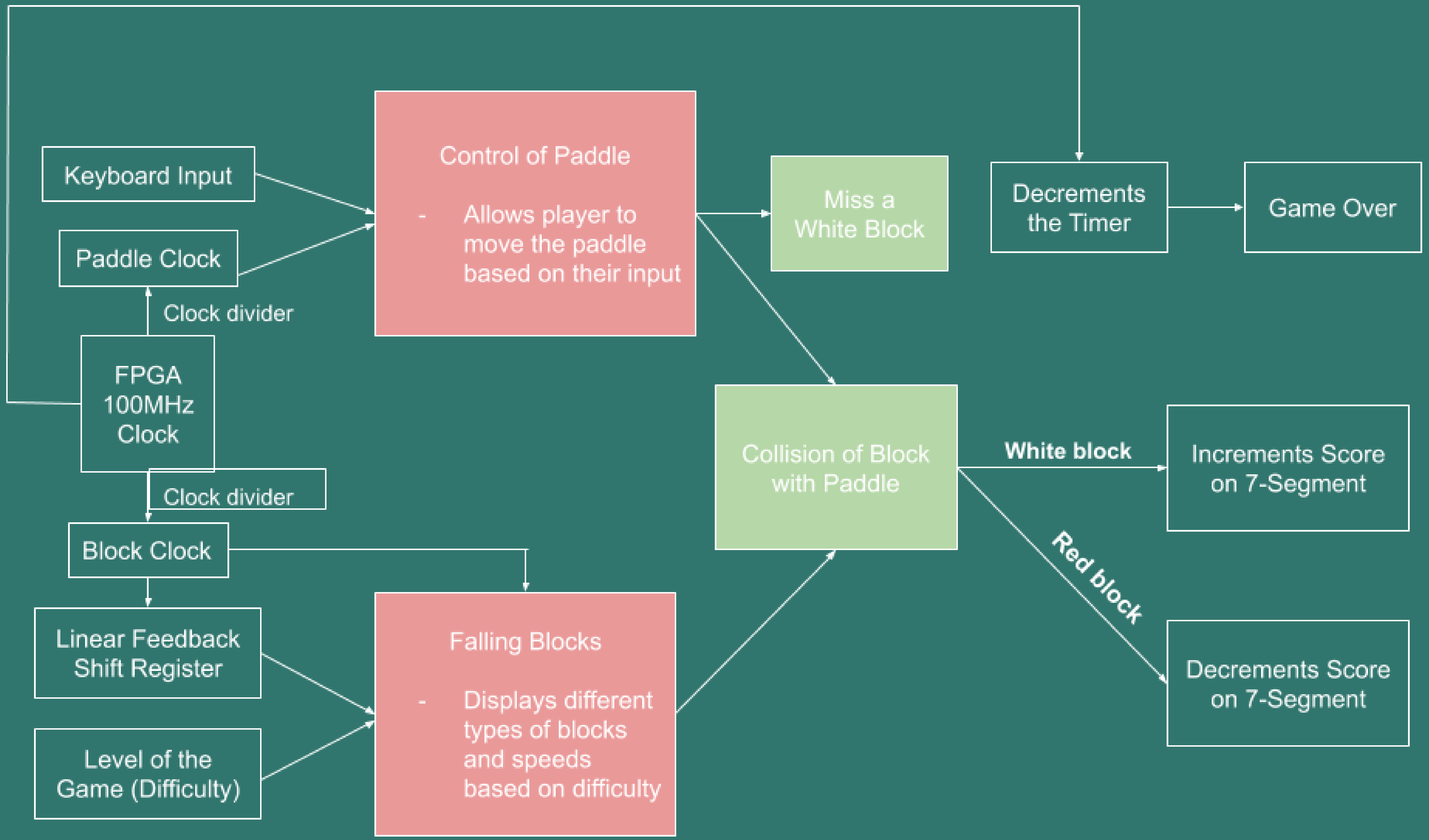
SPECIFICATIONS

- Requirements
  - Player needs to know rules prior to the game
  - The FPGA, VGA display, speaker, and keyboard must be set up for the game to work
- Constraints
  - Can only be played by one person at a time
  - Max score that a player can reach is 99

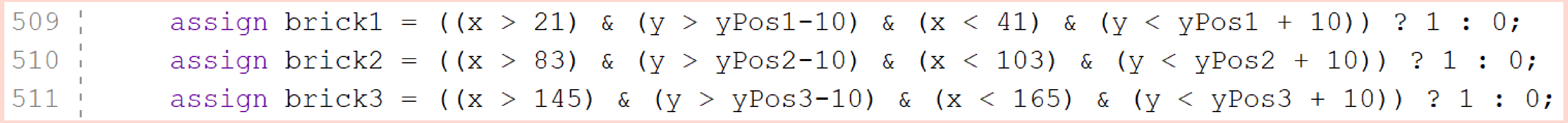
OVERALL BLOCK DIAGRAM


Inputs
Outputs
Functionality of modules and randomization of numbers with clock
User input through keyboard
Verilog Modules
- Linear feedback shift register
- Debouncer
- PS2Receiver
- BCD converter
- Score display
- Music ROM
- Divide by 12
- Creation of song
- VGA640x480
- Brick display
Sound through speaker
Text and game through VGA
Score through 7-segment display

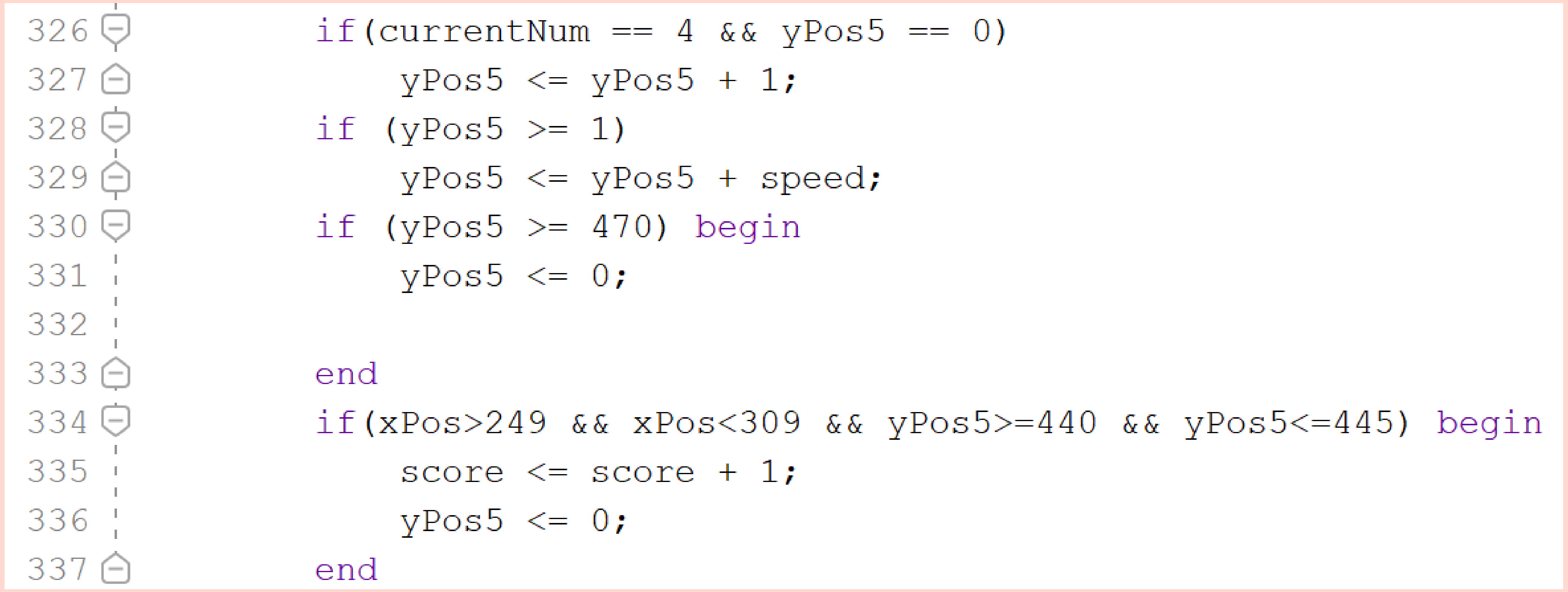
GAME BLOCK DIAGRAM


Keyboard Input
Paddle Clock
Clock divider
FPGA 100MHz Clock
Clock divider
Block Clock
Linear Feedback Shift Register
Level of the Game (Difficulty)
Control of Paddle
- Allows player to move the paddle based on their input
Falling Blocks
- Displays different types of blocks and speeds based on difficulty
Miss a White Block
Collision of Block with Paddle
Decrements the Timer
Game Over
White block
Increments Score on 7-Segment
Red block
Decrements Score on 7-Segment

CODE SNIPPET #1A: FALLING BLOCKS & COLLISION DETECTION



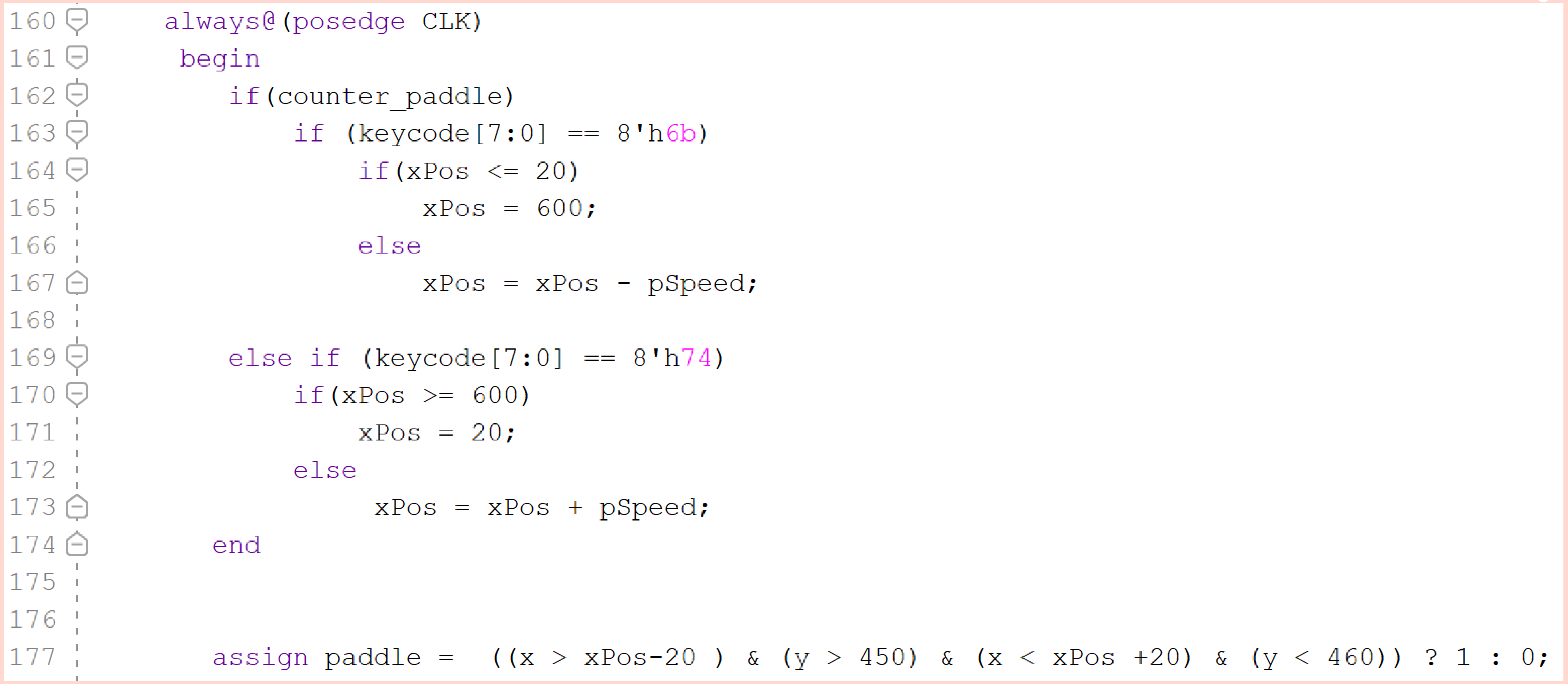
```verilog
    assign brick1 = ((x > 21) & (y > yPos1-10) & (x < 41) & (y < yPos1 + 10)) ? 1 : 0;
    assign brick2 = ((x > 83) & (y > yPos2-10) & (x < 103) & (y < yPos2 + 10)) ? 1 : 0;
    assign brick3 = ((x > 145) & (y > yPos3-10) & (x < 165) & (y < yPos3 + 10)) ? 1 : 0;
```



```verilog
        if(currentNum == 4 && yPos5 == 0)
            yPos5 <= yPos5 + 1;
        if (yPos5 >= 1)
            yPos5 <= yPos5 + speed;
        if (yPos5 >= 470) begin
            yPos5 <= 0;

        end
        if(xPos>249 && xPos<309 && yPos5>=440 && yPos5<=445) begin
            score <= score + 1;
            yPos5 <= 0;
        end
```

CODE SNIPPET #1B: PADDLE



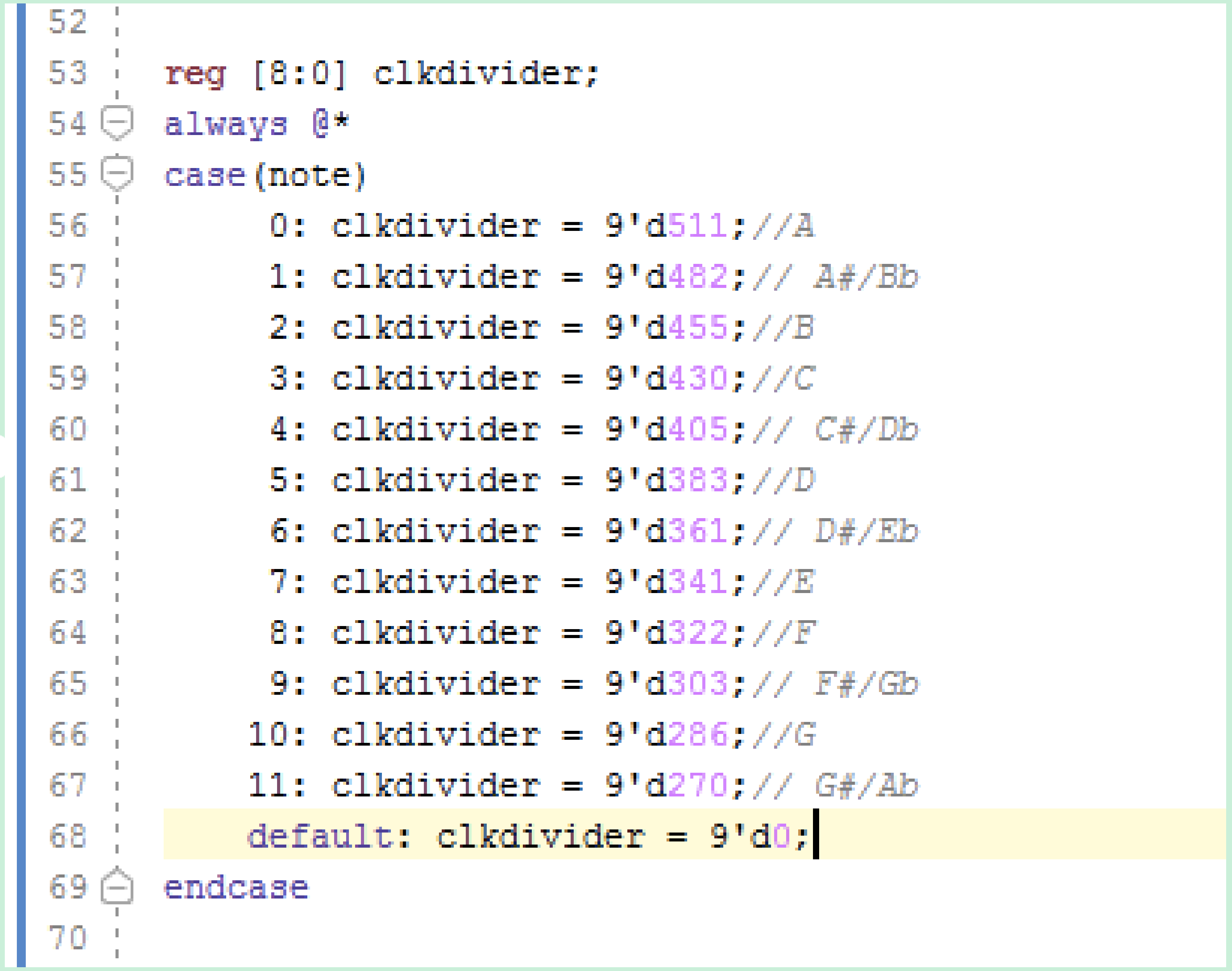
```
always@(posedge CLK)
 begin
   if(counter_paddle)
       if (keycode[7:0] == 8'h6b)
           if(xPos <= 20)
               xPos = 600;
           else
               xPos = xPos - pSpeed;

   else if (keycode[7:0] == 8'h74)
       if(xPos >= 600)
           xPos = 20;
       else
            xPos = xPos + pSpeed;
  end


  assign paddle =  ((x > xPos-20 ) & (y > 450) & (x < xPos +20) & (y < 460)) ? 1 : 0;
```

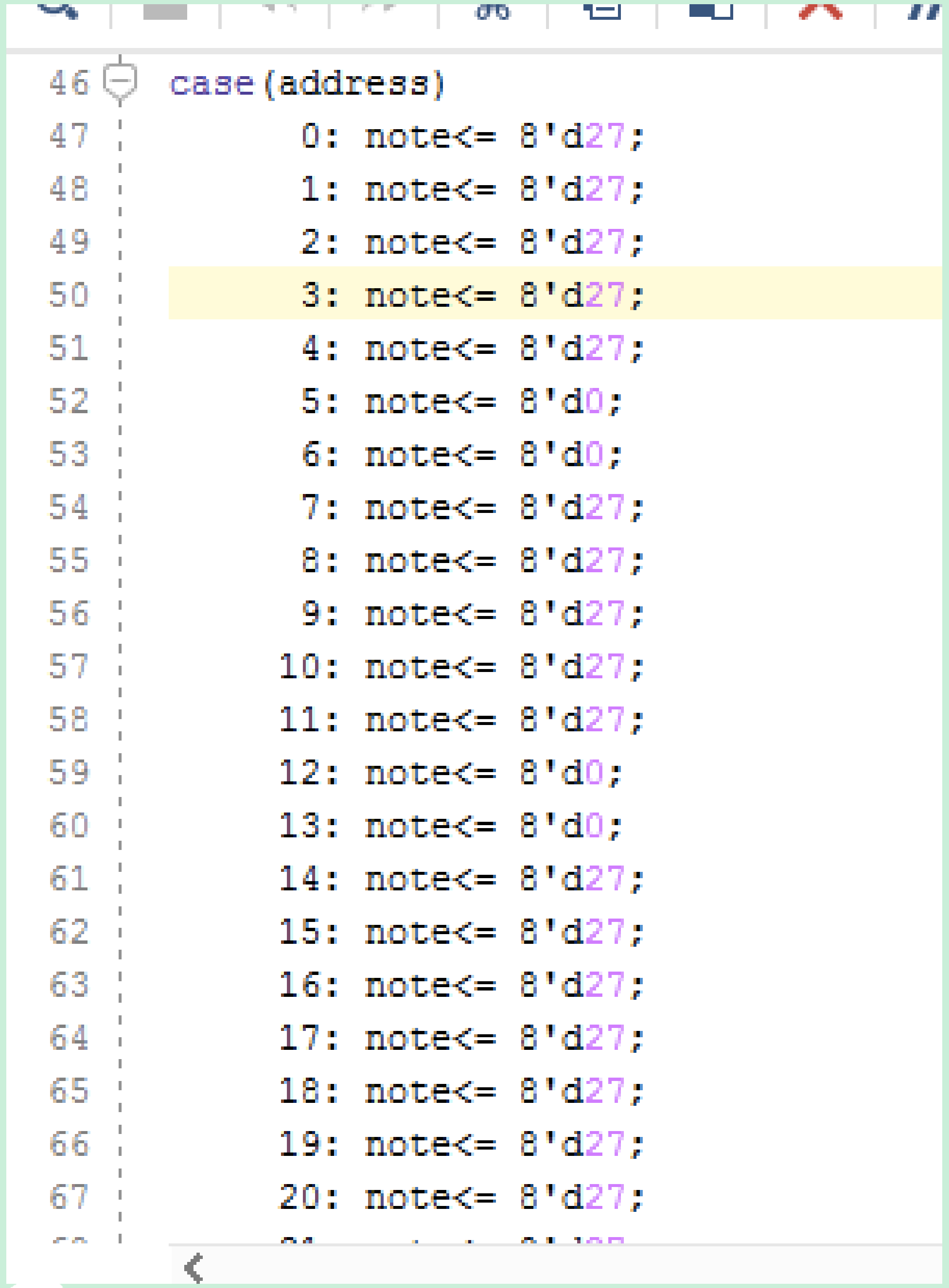
CODE SNIPPET #2: JINGLE BELLS SONG

Dividing into FPGA Clock 12 different notes



```
reg [8:0] clkdivider;
always @*
case(note)
     0: clkdivider = 9'd511;//A
     1: clkdivider = 9'd482;// A#/Bb
     2: clkdivider = 9'd455;//B
     3: clkdivider = 9'd430;//C
     4: clkdivider = 9'd405;// C#/Db
     5: clkdivider = 9'd383;//D
     6: clkdivider = 9'd361;// D#/Eb
     7: clkdivider = 9'd341;//E
     8: clkdivider = 9'd322;//F
     9: clkdivider = 9'd303;// F#/Gb
    10: clkdivider = 9'd286;//G
    11: clkdivider = 9'd270;// G#/Ab
    default: clkdivider = 9'd0;
endcase
```

Writing the song



```
case(address)
         0: note<= 8'd27;
         1: note<= 8'd27;
         2: note<= 8'd27;
         3: note<= 8'd27;
         4: note<= 8'd27;
         5: note<= 8'd0;
         6: note<= 8'd0;
         7: note<= 8'd27;
         8: note<= 8'd27;
         9: note<= 8'd27;
        10: note<= 8'd27;
        11: note<= 8'd27;
        12: note<= 8'd0;
        13: note<= 8'd0;
        14: note<= 8'd27;
        15: note<= 8'd27;
        16: note<= 8'd27;
        17: note<= 8'd27;
        18: note<= 8'd27;
        19: note<= 8'd27;
        20: note<= 8'd27;
```

SUCCESSES

- ✓ VGA Display
    - paddle, block, go & end screens
- ✓ Collision detection
- ✓ Randomization with LFSR
- ✓ Score Tracking via 7 segment display
- ✓ Background music

FAILURES

- ✗ Hard-coded text
- ✗ No instructions
- ✗ Depending on randomization, some falling blocks generate too close together
- ✗ Could not implement health mechanism

THANK YOU!

Any Questions?