BİL 265 Proje

Hafiza Oyunu

Eda İnan - 221201076
Görkem Revi - 201101058
Korhan Arat Karaman - 221201037
Berk Eser İleli - 201201063
Emre Kaan Uzunöz - 201101053

1. Oyun Akışı

   FPGA programlandıktan sonra orta tuşa basıldığında 4x4 boyutunda tabladaki 16 kartın hepsi kısa bir süreliğine açılıp geri kapanır ve oyun başlar. Oyuncu bu sürede aklında tutabildiği kadar kart tutup kartları eşleştirmeye çalışır. Yanlış bir eşleştirme yaptığında kartlar geri kapanır ve buzzer öter. Doğru bildiği kart çifti sayısı 7-segment ekranda oyun boyunca gözükür. Bütün kartları eşleştirdiğinde oyunu kazanmış olur ve 7-segment ekranda WIN yazısı belirir.

2. Program Akışı

   2.1 Modüller

   -hafizaoyunu.v

   -debouncer.v

   -vga_controller.v

   2.2 Program Akışı

   Başlangıçta kartlar yan yana olacak şekilde tanımlanır. Oyun başlatıldığında kartlar karıştırılır ve ekranda gösterilir. Karıştırma için shuffling yapılır. Orta tuşa basıldığında çalışmakta olan bir sayacın o anki değerinin modu alınır, bu değer seed olarak kullanılır ve kartlar sondaki indisten başlanarak birbirleriyle yer değiştirir.

```
// Initialize Cards (8 Pairs: 0-7)
initial begin
    kare[0] = 3'd0; kare[1] = 3'd0; kare[2] = 3'd1; kare[3] = 3'd1;
    kare[4] = 3'd2; kare[5] = 3'd2; kare[6] = 3'd3; kare[7] = 3'd3;
    kare[8] = 3'd4; kare[9] = 3'd4; kare[10] = 3'd5; kare[11] = 3'd5;
    kare[12] = 3'd6; kare[13] = 3'd6; kare[14] = 3'd7; kare[15] = 3'd7;
end
```

Resim 1 – Kartların ilk tanımı

```
always @(posedge clk) begin
    if (gir) begin
        gir_pressed <= 1;
    end
    if (gir_pressed && !start) begin
        for (i = 15; i >= 0; i = i - 1) begin
            if (i == 15) begin
                rand0to15 = 6;
            end
            else begin
                rand0to15 = sayac % (15 - i);
            end
            tempKare = kare[i];
            kare[i] = kare[rand0to15];
            kare[rand0to15] = tempKare;
        end
        start = 1;
        gir_pressed <= 0;
        openCards <= 16'b1111111111111111;
    end
    else if (start && !is_flashed) begin
         if (flash_counter == 100_000_000) begin
            openCards <= 16'b0000000000000000;
            flash_counter <= 0;
            is_flashed <= 1;
        end else
            flash_counter <= flash_counter + 1;
    end
```

Resim 2 – Shuffling

gir orta tuşun sinyalini ifade eder, gir_pressed ise bu sinyal için hafıza görevi görür. start ise oyunun başlayıp başlamadığını ifade eder. Oyun başlatıldığında start ve gir_pressed değişkenine 1 atanır.
is_flashed başlangıçta kartların gösterip gösterilmediğinin hafıza değeridir ve başlangıç değeri 0'dır. Kartlar karıştırıldıktan sonra flash_counter sayacı sınır değerine ulaşana kadar kartlar açık kalmaya devam eder, sonrasında kapanır. is_flashed değişkenine 1 atanır.
flash_counter sayacının sınır değeri arttırılarak oyunun zorluğu, yani kartların başlangıçta ezberlenmesi için açık tutulduğu süre, azaltılabilir.

Bu aşama bittikten sonra oyuncu kartları eşleştirmeye başlar. numOfSelection kaç kart seçildiğini kontrol eder. İki kart seçildikten sonra kontrol edilir. Eğer kartlar eşleşiyorsa açık kalır, eşleşmiyorsa geri kapanır.

```
if (gir && start) begin
    if (!numOfSelection && !openCards[index]) begin
        openCards[index] <= 1;
        prevIndex <= index;
        numOfSelection <= 1;
    end else if (numOfSelection && !openCards[index]) begin
        openCards[index] <= 1;
        if (kare[index] != kare[prevIndex])
            clockOff <= 1;
        else
            correctGuess <= correctGuess + 1;
        numOfSelection <= 0;
    end
    else if (correctGuess == 8) begin
        reset <= 1;
    end
end
```

Resim 3 – Kart eşleştirme

openCards dizisi, kare dizisine ek olarak hangi indislerdeki kartların açık olduğunu kontrol eder. Örneğin birinci, yani ekranın sol üstündeki, kart açıksa openCards[0] == 1, değilse 0 olur.

correctGuess kaç doğru eşleştirme yapıldığını tutar. Oyun bittiğinde, yani bütün kartlar eşleştirildiğinde, orta tuşa basılırsa oyun baştan başlar.

```
else if (correctGuess == 8 && reset) begin
    start <= 0;
    reset <= 0;
    correctGuess <= 0;
    flash_counter <= 0;
    delay_counter <= 0;
    is_flashed <= 0;
    prevIndex <= 0;
    clockOff <= 0;
    numOfSelection <= 0;
    openCards <= 16'b0000000000000000;
end
```

Resim 4 – Reset bloğu

```
        // Delay for mismatched cards
        if (clockOff) begin
            if (delay_counter == 20_000_000) begin
                openCards[index] <= 0;
                openCards[prevIndex] <= 0;
                clockOff <= 0;
                delay_counter <= 0;
                buzzer_active <= 0;  // Deactivate the buzzer when mismatch handling ends
                buzzer <= 0;         // Ensure buzzer is off
            end else begin
                delay_counter <= delay_counter + 1;
                buzzer_active <= 1;
            end
        end

        // Manage the buzzer duration
        if (buzzer_active) begin
            if (buzzer_counter == 25_000) begin // Toggle every 25k cycles for a 2 kHz tone
                buzzer_counter <= 0;
                buzzer <= ~buzzer;
            end else begin
                buzzer_counter <= buzzer_counter + 1;
            end
        end else begin
            buzzer <= 0;  // Ensure buzzer is off when inactive
        end
    end
```

Resim 5 – Yanlış eşleştirmede kart gösterimi ve buzzerın çalışması

clockOff, yanlış eşleşme sinyali olarak kullanılır. clockOff 1 olduğunda kartlar delay_counter sayacının sınır değeri süresince açık kalır ve buzzer buzzer_counter sayacının sınır değeri süresince öter. Kartlar geri kapandıktan sonra buzzer susar.

Oyun gidişatı ve kartlar 640x480 çözünürlükte VGA ekranda basılır. Kartların değerleri renk olarak ayarlanır ve seçilen kartın etrafında beyaz bir çerçeve olarak imleç gösterilir. İmleç ekranın uçlarına ulaşınca kullanıcı indisi daha fazla hareket ettiremez.

```
// VGA Grid Display
always @(posedge clk) begin
    row = y / 120;
    col = x / 160;
    grid_index = row * 4 + col;

    red = 0; green = 0; blue = 0;

    if (active_video) begin
        if (openCards[grid_index] && row < 4 && col < 4) begin
            case (kare[grid_index])
                3'd0: red = 15;          // Red
                3'd1: green = 15;        // Green
                3'd2: blue = 15;         // Blue
                3'd3: begin red = 15; green = 15; end // Yellow
                3'd4: begin green = 15; blue = 15; end // Cyan
                3'd5: begin red = 15; blue = 15; end// Magenta
                3'd6: red = 3;           // Maroon
                3'd7: blue = 5;          // Navy Blue
            endcase
        end else begin
            red = 4; green = 4; blue = 4; // Gray for hidden cards
        end

        // Draw the cursor
        if (grid_index == index && is_flashed) begin
            if ((x % 160 < 5) || (x % 160 > 155) || (y % 120 < 5) || (y % 120 > 115)) begin
                red = 15; green = 15; blue = 15; // White frame
            end
        end
    end
end
```

Resim 6 – Kartların ve imlecin ekrana basılması

```
// Cursor Movement Logic
always @(posedge clk) begin
    if (UP && index >= 4) index <= index - 4;
    if (DOWN && index <= 11) index <= index + 4;
    if (RIGHT && index % 4 != 3) index <= index + 1;
    if (LEFT && index % 4 != 0) index <= index - 1;
    if (reset) index = 0;
end
```

Resim 7 – Girdi sınırlaması

FPGA'in 7-segment ekranında doğru eşleşme sayısı oyun boyunca gözükür. Bütün eşleşmeler yapıldığında, yani correctGuess == 8 olduğunda, 7-segmente WIN yazısı iki U harfinin birleşimi W olacak şekilde basılır.

```
reg [1:0] k = 0;

always @(posedge clk_yavas) begin
    if (correctGuess < 8) begin
        an = 4'b0111;
        case (correctGuess)
            4'h0: seg = ZERO;
            4'h1: seg = ONE;
            4'h2: seg = TWO;
            4'h3: seg = THREE;
            4'h4: seg = FOUR;
            4'h5: seg = FIVE;
            4'h6: seg = SIX;
            4'h7: seg = SEVEN;
        endcase
    end else begin
        case (k)
            2'b11: begin an = 4'b0111; seg = U; end
            2'b10: begin an = 4'b1011; seg = U; end
            2'b01: begin an = 4'b1101; seg = I; end
            2'b00: begin an = 4'b1110; seg = N; end
        endcase
        k = k + 1;
    end
end
```

Resim 8 – 7-segment ekranında görüntüleme

3. Referanslar

Oyunun genel işleyişi ve 7-segment ekranının programlanması, yani hafizaoyunu.v dosyasının bütünü, grup üyelerince yazılmıştır. Debouncing için kullandığımız debouncer.v modülü 265 dersi kapsamında yapılan 9. lab dersinden hazır alınmıştır[1]. VGA kontrolcüsü görevi gören vga_controller.v modülü ise Basys3 Reference Manual, internet ve ChatGPT yardımıyla oluşturulmuştur.