T.C. KOCAELİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ

FPGA TABANLI PMOD UYUMLU 4X4 MATRİS KEYPAD TASARIMI VE VERILOG SÜRÜCÜ UYGULAMASI

Hazırlayan: Mehmet Burak YILMAZ 250207098

Ders Sorumlusu: Doç. Dr. Anıl ÇELEBİ

KOCAELİ, 2026

İÇİNDEKİLER

1. GİRİŞ

1.1. Projenin Amacı ve Kapsamı

Gömülü sistemlerde kullanıcı etkileşimi, sistemin işlevselliği açısından kritik bir rol oynar. Bu projenin amacı; FPGA tabanlı sistemler için özgün bir 4x4 Matris Tuş Takımı (Keypad) donanımı tasarlamak, üretmek ve bu donanımı FPGA üzerinde Verilog HDL (Donanım Tanımlama Dili) kullanarak sürmektir.

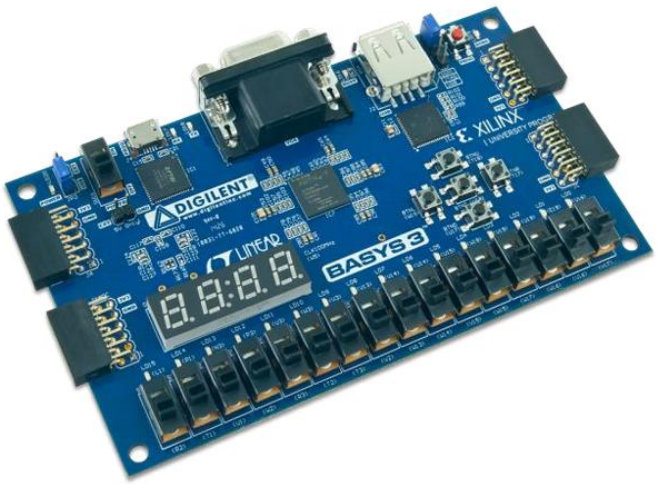
Proje kapsamında; hazır modül kullanımı yerine devre şeması ve PCB çizimi özgün olarak gerçekleştirilmiş, üretim aşamasında asit ve SMD lehimleme teknikleri uygulanmıştır. Yazılım tarafında ise, mekanik butonlardan kaynaklanan sinyal gürültülerini (bouncing) önlemek adına "Latching (Kilitlemeli)" tarama algoritması geliştirilmiştir.

Projenin tasarım ve uygulama aşamalarında; üretici firmanın teknik dokümanlarından, literatürdeki benzer FPGA uygulamalarından ve önceki yıllarda yapılan akademik çalışmalardan faydalanılmıştır [1-5].

2. GENEL BİLGİLER

2.1. FPGA ve Basys 3 Mimarisi

Projede kullanılan Basys 3 geliştirme kartı, Xilinx Artix-7 ailesine ait FPGA çipini barındırır. FPGA'lar, mikrodenetleyicilerin aksine işlemleri sıralı değil, paralel olarak işleme yeteneğine sahiptir. Bu özellik, giriş/çıkış işlemlerinin nanosaniyeler mertebesinde gecikmesiz kontrol edilmesine olanak tanır.



Şekil 2.1.1 *Basys 3 Kartı*

2.2. Matris Keypad Çalışma Prensibi

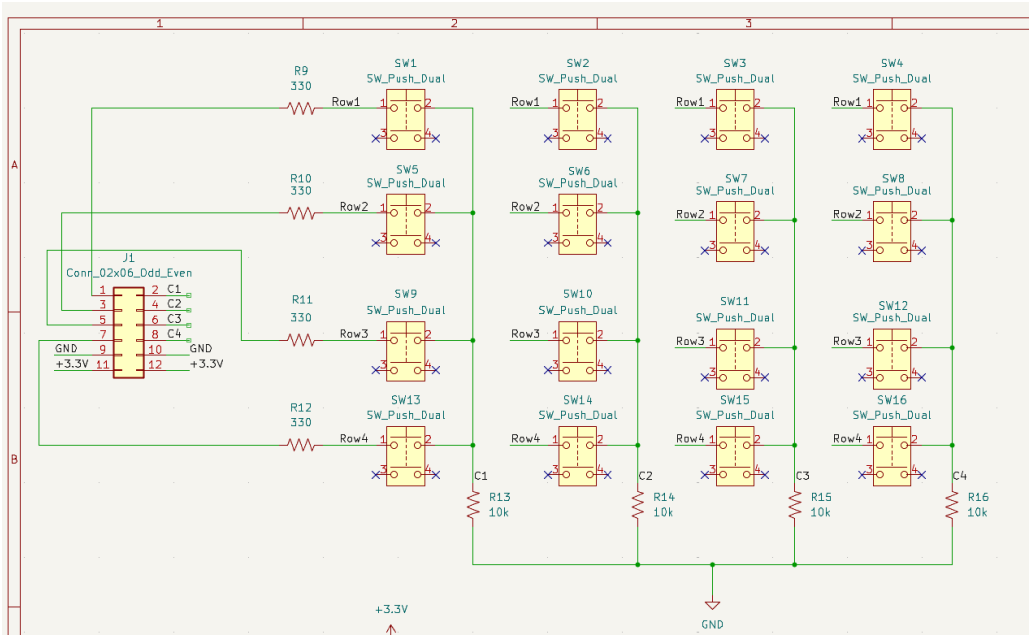
16 adet butonun her biri için ayrı bir giriş pini kullanmak FPGA kaynaklarını verimsiz kullanacağından, projede 4 satır ve 4 sütundan oluşan matris yapısı tercih edilmiştir. Bu sayede 16 buton sadece 8 pin (4 giriş, 4 çıkış) ile kontrol edilebilmektedir. FPGA, satırlara sırasıyla enerji verirken, sütunlardan dönen sinyali dinleyerek basılan tuşun konumunu (satır-sütun kesişimi) tespit eder.

3. DONANIM TASARIMI VE ÜRETİM

3.1. Devre Şeması ve Koruma Devreleri

Devre tasarımında, sinyal kararlılığını sağlamak ve FPGA'yı korumak için aşağıdaki önlemler alınmıştır:

- **Koruma Dirençleri (330Ω):** FPGA pinlerinden çekilecek akımı sınırlamak ve olası kısa devre durumlarında portun yanmasını önlemek için seri dirençler kullanılmıştır. Bu dirençler FPGA kartın PMOD çıkışlarında bulunan 220Ω dirençlere seri olarak ek güvenlik amacıyla eklenmiştir.
- **Pull-Down Dirençleri (10kΩ):** Butonlar basılı değilken giriş pinlerinin "yüzer" (floating) durumda kalmasını engellemek ve sinyali lojik-0 (Toprak) seviyesine çekmek için PCB üzerine fiziksel dirençler eklenmiştir. Bu sayede FPGA içindeki zayıf pull-down dirençlerine bağımlılık ortadan kaldırılmıştır.

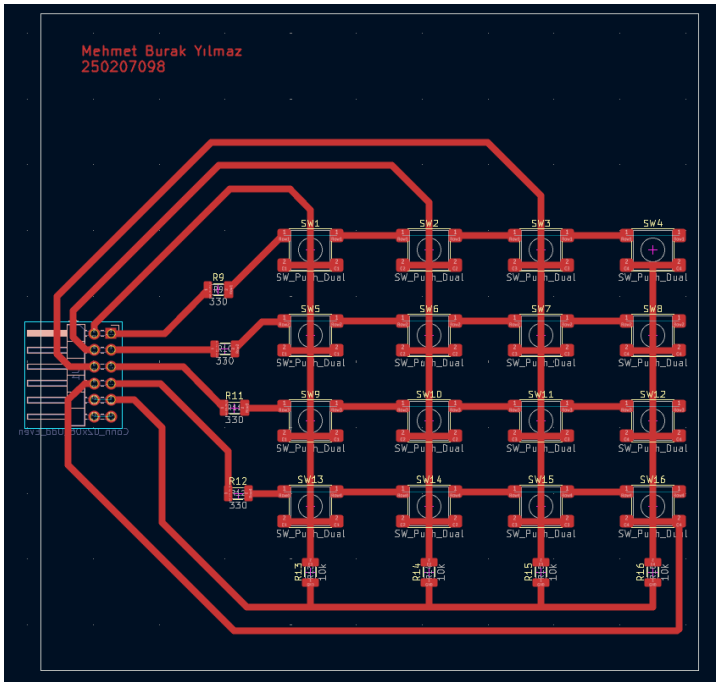


Şekil 3.1.1 *Keypad Şeması*

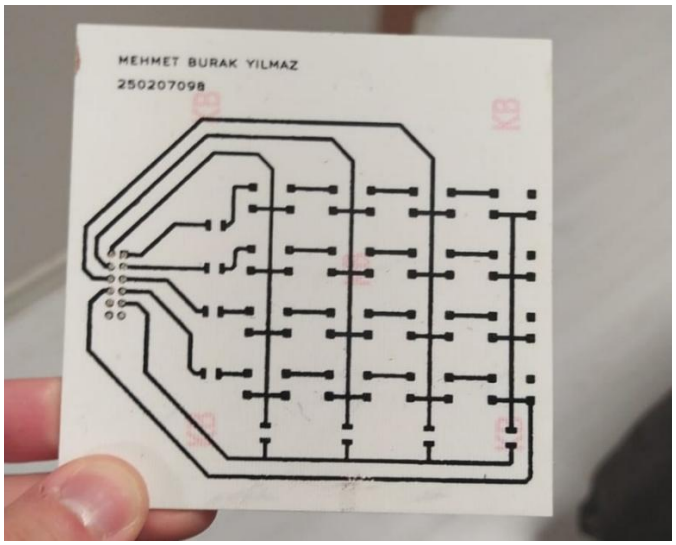
3.2. PCB Tasarımı ve Montaj Süreci

Devre şeması KiCad ortamında çizilmiş ve PMOD standardına uygun fiziksel yerleşim planı oluşturulmuştur. PCB üretiminde "Toner Transferi" yöntemi kullanılmış, bakır yollar asit indirme işlemiyle oluşturulmuştur. Montaj aşamasında SMD (Yüzey Montaj) butonlar ve dirençler lehimlenerek kart kullanıma hazır hale getirilmiştir.

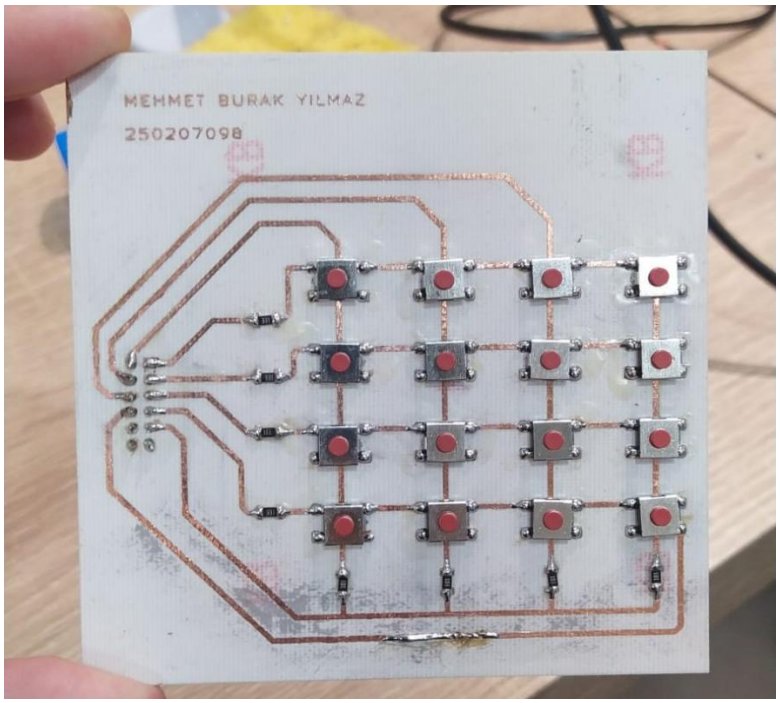
PCB tasarımında, özellikle direnç ve butonlar için standart kılıfların pedlerinin (ayaklarının) daha genişletilmiş hali kullanılmıştır. Bu değişiklik; el ile lehimleme işlemini kolaylaştırmak ve asit indirme işlemi sırasında bakır yolların ve pedlerin zarar görmesini (aşırı incelmesini) engellemek amacıyla yapılmıştır. Seçilen kılıfların detayları EK-B'de sunulmuştur.



Şekil 3.1.2 *Keypad PCB Tasarımı*



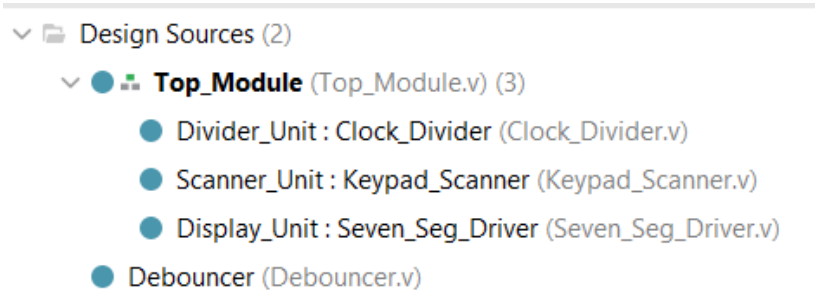
Şekil 3.2.1 *Asit İşlemi Sonrasında Keypad*



Şekil 3.2.2 Montajı Tamamlanmış Keypad

4. YAZILIM TASARIMI (VERILOG)

Donanımı kontrol eden yazılım, modüler bir yapıda tasarlanmıştır. Top_Module çatısı altında üç ana alt modül bulunmaktadır.



***Şekil 4.1** Top_Module*

4.1. Clock Divider (Saat Bölücü)

Basys 3 kartının 100 MHz'lik ana saati, mekanik buton taraması için çok yüksektir. Bu modül, sistem saatini bölerek tarama hızını yaklaşık 20 Hz (Limit: 2.500.000) seviyelerine düşürür. Bu hız, hem insan algısı için yeterince seri hem de elektriksel sinyallerin oturması için yeterince yavaştır. Clock Divider algoritmasının kodu EK-A' da verilmiştir.

4.2. Keypad Scanner (Hafızalı Tarama Algoritması)

Projenin en kritik yazılım geliştirmesi bu modülde yapılmıştır. Mekanik butonlarda oluşan ark (bouncing) ve temassızlık sorunları nedeniyle standart tarama algoritmalarında tuş algılama hataları gözlemlenmiştir. Bu sorunu çözmek için "Freeze & Latch" (Dondur ve Kilitle) algoritması uygulanmıştır:

1. **Freeze (Dondurma):** Tarama sırasında herhangi bir sütundan sinyal (col != 0) geldiği anda, satır sayacı (scan_timer) durdurulur. Böylece tarayıcının yanlış satıra geçmesi ve sinyali kaybetmesi engellenir.

```
// Basılıysa (else) olduğun yerde kal
if (col == 4'b0000) begin
    scan_timer <= scan_timer + 1;
end
```

Şekil 4.2.1 Satır Sayacını Durduran Verilog Kodu

2. **Latch (Kilitleme):** Tuş tespit edildiğinde okunan değer hafızaya alınır (reg). Kullanıcı parmağını çekse veya sinyalde mikrosaniyelik kopmalar olsa bile, çıkış (key_out) değişmez.

3. **Reset:** Ekranı temizlemek veya yeni değer girmek için Reset (btnC) sinyali kullanılır.

Bu yöntem sayesinde harici bir Debouncer (Gürültü Engelleyici) modülüne ihtiyaç duyulmadan, son derece kararlı bir okuma elde edilmiştir. Keypad Scanner verilog kodları EK-A'da verilmiştir.

4.3. Seven Segment Driver

Scanner modülünden gelen 4-bitlik veriyi (0-F) alır ve 7-Segment Display'in anlayacağı desene dönüştürür. Basys 3 ekranları "Ortak Anot" yapısında olduğu için çıkış sinyalleri aktif-düşük (Active Low - 0 ile yanar) mantığına göre kodlanmıştır. Seven Segment Driver verilog kodları EK-A'da verilmiştir.

```
always @(*) begin
    case(in_number)
        //                  gfedcba
        4'h0: seg = 7'b1000000; // 0
        4'h1: seg = 7'b1111001; // 1
        4'h2: seg = 7'b0100100; // 2
        4'h3: seg = 7'b0110000; // 3
        4'h4: seg = 7'b0011001; // 4
        4'h5: seg = 7'b0010010; // 5
        4'h6: seg = 7'b0000010; // 6
        4'h7: seg = 7'b1111000; // 7
        4'h8: seg = 7'b0000000; // 8
        4'h9: seg = 7'b0010000; // 9
        4'hA: seg = 7'b0001000; // A
        4'hB: seg = 7'b0000011; // b (Küçük b çünkü B, 8 ile karışır)
        4'hC: seg = 7'b1000110; // C
        4'hD: seg = 7'b0100001; // d (Küçük d çünkü D, 0 ile karışır)
        4'hE: seg = 7'b0000110; // E
        4'hF: seg = 7'b0001110; // F
        default: seg = 7'b1111111; // Hata veya boş (Hepsi sönük)
    endcase
end
```

Şekil 4.3.1 *Seven Segment Driver*

5. TEST VE SONUÇLAR

Üretilen kart Basys 3 üzerindeki JB PMOD portuna bağlanarak test edilmiştir.

- **Donanım Testi:** PCB yollarının iletkenliği ve Pull-Down dirençlerinin işlevselliği multimetre ile doğrulanmıştır. Tuşa basılmadığında gerilim 0V, basıldığında 3.3V olarak ölçülmüştür.
- **Entegrasyon Testi:** Geliştirilen yazılım yüklendiğinde, tuşlara basıldığı an ilgili sayının (0-9, A-F) 7-Segment ekranda belirdiği görülmüştür.
- **Kararlılık Testi:** Geliştirilen "Hafızalı Tarama" algoritması sayesinde, tuşa basılı tutarken veya hızlı bas-çek yaparken ekranda titreme (flickering) veya yanlış karakter görünmesi sorunu tamamen ortadan kalkmıştır.

Sonuç olarak; donanım ve yazılımın tam uyum içinde çalıştığı, PMOD uyumlu, kararlı bir giriş birimi başarıyla elde edilmiştir.

6. KAYNAKÇA

[1] Digilent Inc., "Basys 3 FPGA Board Reference Manual," Pullman, WA, USA, 2014.

Erişim: https://digilent.com/reference/programmable-logic/basys-3/reference-manual

[2] Digilent Inc., "Basys 3 Board Schematic (Circuit Diagram)," Rev C.1, 2014.

Erişim: https://digilent.com/reference/_media/basys3:basys3_sch.pdf

[3] U. Aldıoğlu, "PMOD 4x4 Keypad Design Project Report," Kocaeli Üniversitesi, Mühendislik Fakültesi, Elektronik ve Haberleşme Müh. Bölümü, Kocaeli, 2025.

[4] P. P. Chu, *FPGA Prototyping by Verilog Examples*. Hoboken, NJ, USA: John Wiley & Sons, 2008.

[5] Z. Özçelik, M. S. H. Gedik ve Z. Hayta, "PMOD Keypad Reference Document," Kocaeli Üniversitesi, Mühendislik Fakültesi, Elektronik ve Haberleşme Müh. Bölümü, Kocaeli, 2022.

Erişim: https://github.com/zeynepozcelk/KeyPad-VerilogHDL-Basys3

7. EKLER

EK-A

Clock_Divider.v

```
module Clock_Divider(
  input clk,
  input rst,
  output reg slow_clk
  );
  // LIMIT = 50.000.000 (Yaklaşık 1 saniye aralıkla yanıp söner)
  localparam LIMIT = 2500000;
  integer counter = 0;
  always @(posedge clk or posedge rst) begin
    if (rst) begin
      counter <= 0;
      slow_clk <= 0;
    end
    else begin
      if (counter == LIMIT - 1) begin
        slow_clk <= ~slow_clk; // 1 ise 0 yap, 0 ise 1 yap
        counter <= 0;
      end
      else begin
        counter <= counter + 1;
      end
    end
  end
endmodule
```

Keypad_Scanner.v

```
`timescale 1ns / 1ps
module Keypad_Scanner(
  input clk,          // Yavaş saat (Clock Divider'dan)
  input rst,          // RESET EKLENDİ (Ekranı temizlemek için şart!)
  input [3:0] col,      // Sütunlar
  output reg [3:0] row,   // Satırlar
  output reg [3:0] key_out // Çıkış artık hafızalı
  );
  reg [1:0] scan_timer;
  always @(posedge clk or posedge rst) begin
    if (rst) begin
```

```
        scan_timer <= 0;
        row <= 4'b0000;
      end
      else begin
        // Tuş basılı değilse (0000) gezmeye devam et
        if (col == 4'b0000) begin
          scan_timer <= scan_timer + 1;
        end
        case(scan_timer)
          2'b00: row <= 4'b0001;
          2'b01: row <= 4'b0010;
          2'b10: row <= 4'b0100;
          2'b11: row <= 4'b1000;
        endcase
      end
    end
  // --- 2. ADIM: TUŞ OKUMA VE HAFIZAYA ALMA ---
    always @(posedge clk or posedge rst) begin
      if (rst) begin
        key_out <= 0; // Reset gelince hafızayı sil
      end
      else begin
        // Eğer col == 0 ise, buraya girmez ve key_out ESKİ DEĞERİNİ KORUR.
        if (col != 4'b0000) begin
          case (row)
            4'b0001: begin
              if (col[0]) key_out <= 4'h1;
              if (col[1]) key_out <= 4'h2;
              if (col[2]) key_out <= 4'h3;
              if (col[3]) key_out <= 4'hA;
            end
            4'b0010: begin
              if (col[0]) key_out <= 4'h4;
              if (col[1]) key_out <= 4'h5;
              if (col[2]) key_out <= 4'h6;
              if (col[3]) key_out <= 4'hB;
            end
            4'b0100: begin
              if (col[0]) key_out <= 4'h7;
              if (col[1]) key_out <= 4'h8;
              if (col[2]) key_out <= 4'h9;
              if (col[3]) key_out <= 4'hC;
            end
            4'b1000: begin
              if (col[0]) key_out <= 4'hE;
              if (col[1]) key_out <= 4'h0;
              if (col[2]) key_out <= 4'hF;
```

```
            if (col[3]) key_out <= 4'hD;
          end
        endcase
      end
    end
  end
endmodule
```

Seven_Seg_Driver.v

```
`timescale 1ns / 1ps
  module Seven_Seg_Driver(
   input clk,
   input [3:0] in_number,   // Keypad'den gelen 4-bitlik sayı (0-F arası)
   output reg [6:0] seg,    // a, b, c, d, e, f, g ledleri
   output reg [3:0] an,     // Hangi basamağın yanacağını seçen uçlar
   output dp              // Nokta işareti (İsteğe bağlı)
   );
   assign dp = 1;
   always @(*) begin
     an = 4'b1110;
   end
   // --- SEGMENT KOD ÇÖZÜCÜ (Decoder) ---
   always @(*) begin
     case(in_number)
       //                    gfedcba
       4'h0: seg = 7'b1000000; // 0
       4'h1: seg = 7'b1111001; // 1
       4'h2: seg = 7'b0100100; // 2
       4'h3: seg = 7'b0110000; // 3
       4'h4: seg = 7'b0011001; // 4
       4'h5: seg = 7'b0010010; // 5
       4'h6: seg = 7'b0000010; // 6
       4'h7: seg = 7'b1111000; // 7
       4'h8: seg = 7'b0000000; // 8
       4'h9: seg = 7'b0010000; // 9
       4'hA: seg = 7'b0001000; // A
       4'hB: seg = 7'b0000011; // b (Küçük b çünkü B, 8 ile karışır)
       4'hC: seg = 7'b1000110; // C
       4'hD: seg = 7'b0100001; // d (Küçük d çünkü D, 0 ile karışır)
       4'hE: seg = 7'b0000110; // E
       4'hF: seg = 7'b0001110; // F
       default: seg = 7'b1111111; // Hata veya boş (Hepsi sönük)
     endcase
   end
endmodule
```

Top_Module.v

```
`timescale 1ns / 1ps
  module Top_Module(
  input clk,           // 100 MHz Sistem Saati
  input rst,           // Reset Butonu (btnC) - Ekranı temizlemek için şart!
  input [3:0] col,      // Keypad Sütunları (PMOD'dan gelir)
  output [3:0] row,     // Keypad Satırları (PMOD'a gider)
  output [6:0] seg,     // 7-Segment Segmentler
  output [3:0] an,      // 7-Segment Anotlar
  output dp            // Nokta
  );
  // --- İÇ KABLOLAR (Wires) ---
  wire w_slow_clk;      // Clock Divider'dan çıkan yavaş saat
  wire [3:0] w_key_data;  // Scanner'dan okunan HAFIZALI tuş verisi
  // 1. Modül: Saat Bölücü
  Clock_Divider Divider_Unit (
    .clk(clk),
    .rst(rst),
    .slow_clk(w_slow_clk)
  );
  // 2. Modül: Keypad Tarayıcı
  Keypad_Scanner Scanner_Unit (
    .clk(w_slow_clk),   // Yavaş saat
    .rst(rst),        // YENİ: Hafızayı silmek için reset lazım
    .col(col),        // Giriş
    .row(row),        // Çıkış
    .key_out(w_key_data)   );
  // 3. Modül: Ekran Sürücü
  Seven_Seg_Driver Display_Unit (
    .clk(clk),        // 100 MHz (Kombinasyonel olduğu için fark etmez)
    .in_number(w_key_data), // DİKKAT: Direkt w_key_data bağlandı
    .seg(seg),
    .an(an),
    .dp(dp)
  );
endmodule
```

EK-B

Komponent Tipi	Parça Adı	Kütüphane	Kılıf (Footprint)
Direnç	10kΩ, 330Ω	Resistor_SMD	R_1206_3216Metric_Pad1.30x1.75mm_HandSolder
Buton	Tactile Switch	Button_Switch_SMD	SW_SPST_PTS645Sx43SMTR92
Konnektör	Header 2x6	Connector_PinHeader_2.54mm:	PinHeader_2x06_P2.54mm_Horizontal