

**THE IMAGINATION UNIVERSITY PROGRAMME**

**RVfpga Deney 7**

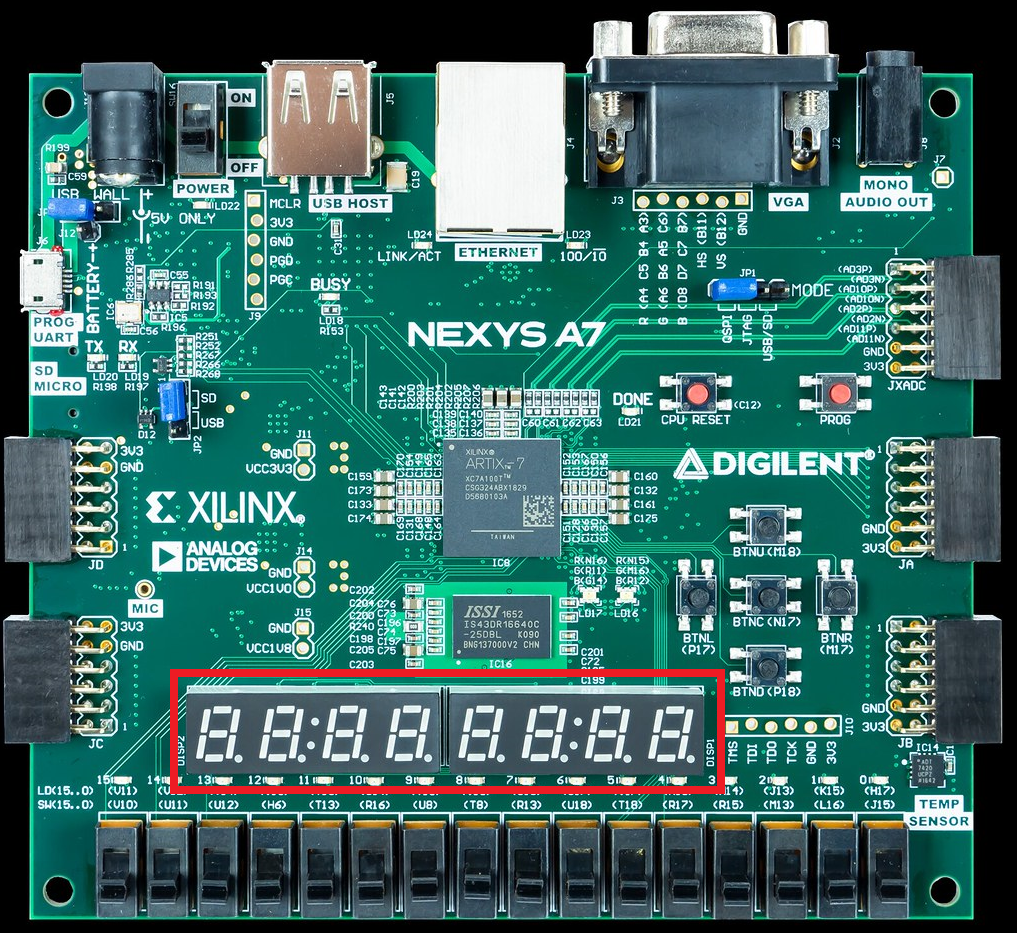
**7-Kesimli Ekranlar**

# GİRİŞ

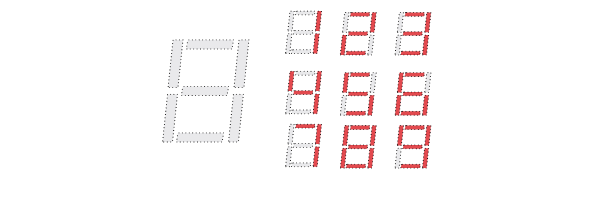
Bu deney RVfpga’in 7-kesimli ekranlarla çalışmak üzere nasıl genişletildiğini tanımlayıp 7-kesimli ekran denetleyicisini nasıl değiştirileceğini gösterir. Nexys A7 FPGA kartının sekiz 7-kesimli ekranı vardır. İlk nasıl çalıştıklarını tanımlayıp (Bölüm 2) ardından RVfpga’de içerilen 8-sayılı 7-kesimli ekranın denetleyicisinin yüksek-düzeyli standardını çözümleyip birkaç temel alıştırma sunuyoruz (Bölümler 3 ile 4). Son olarak bu denetleyicinin alçak-düzeyli gerçekleştirmesini çözümleyip, Verilator simülasyonu yapıp, denetleyici gerçekleştirmesini değiştirip üzerinde oynayacağın ek alıştırmalar sunuyoruz (Bölümler 5 ile 6).

# NEXYS A7 KARTINDAKİ 7-KESİMLİ EKRANLAR

# Nexys A7 kartı bir 8-sayı 7-kesimli ekran gibi davranmak üzere yapılandırılmış iki 4-sayı yaygın-anot 7-kesimli LED ekran (Figür 1) barındırır. Bu sekiz sayıların hepsi kesim başına bir LED ile “figür 8” örüntüsünde dizilmiş 7 kesimden oluşur (Figür 2). Bu kesimler açılabilir ya da kapatılabilir, yani 128 değişik örüntü belirli LED kesimli yakılıp söndürülerek gösterilebilir; daha özel olarak Figür 2’de gösterildiği gibi bu 128 örüntü arasından onluk tabandaki sayılar da seçilebilir.

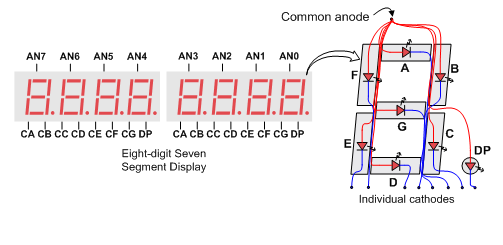


Figür 1. Nexys A7’daki 8-sayı 7-kesimli ekran



Figür 2. Onluk tabandaki sayılara denk gelen örüntüler

# Figür 3’ün sağında gösterildiği gibi bir sayının LED kesimleri A-G olarak etiketlidir. Yedi LEDin anotları bir “ortak anot” devre düğümüne bağlı olsa da LED katotlar ayrı kalır (Figür 3). Sekiz ortak anot sinyali, sayı başına bir sinyal (*AN0*-*AN7*), bir “dijital etkinleştirici” olarak davranır. Sekiz sayının hepsinde aynı kesimin katotları yedi sinyale bağlıdır, *CA*-*CG* (Figür 3’e göz at). (Önemli olarak sekizinci bir sinyal ondalık noktası için vardır, *DP*, ancak bu deneyde kullanmayacağız.) Örneğin sekiz sayının D kesiminin katodu CD adında bir devre düğümüne gruplanmıştır. Bu sinyal bağlantı şeması bir çoklanmış ekran oluşturur, katot sinyaller bütün sayılarda yaygındır ancak yalnızca anot sinyalinin sağlandığı kesimleri aydınlatır. Bu sinyallerin hepsi etkinken düşüğe sürülür; dolayısıyla, bir kesimi aydınlatmak için, örneğin, sayı 2’de D kesimi, AN2 anodu da CD katodu da düşüğe sürülmelidir.



Figür 3. 8-sayı 7-kesim Ekranların Nexys A7’deki bağlantısı

Bir tarayan ekran denetleyici devresi 8-sayı 7-kesimli ekranda 8-rakamlı bir sayı göstermek için kullanılabilir. Bu devre insan gözünün görebileceğinden daha hızlı bir oranda tekrarlayan sürekli yenilemeyle katotları rakamların örüntüsüyle değerlendirir; bu sırada devre anotları da bir kezde bir anot olacak biçimde değerlendirir. Böylelikle bütün sayılar sürenin sekizde birinde aydınlatılır, ancak insan gözü sayının aydınlatılmadan önce karartılmasını algılayamadığından sayı sürekli aydınlatılmış gibi görünür.

8 sayıların hepsinin parlak, sürekli aydınlatılmış gibi görünmesi için sekiz sayının hepsi 1-16ms arasında değerlendirilip yenileme döngüsünün 1/8’inde bütün sayılar aydınlatılmış olmalı (örneğin 16ms’lik bir yenileme dönümünde bütün sayılar 2ms için aydınlatılır). Yukarıda açıklandığı gibi denetleyici bir sayının katotlarını doğru örüntüyle düşüğe sürüp karşılık gelen anot sinyalinin de düşüğe sürülmesi gerekir. Ancak Nexys A7 yaygın anot noktasına yeterli akım sağlamak için NPN transistörleri kullandığından anot etkinleştirmeleri tersine döner. Böylelikle AN0..7 ile CA..G/DP sinyallerinin ikisi de etkinken düşüğe sürülür.

Süreci görselleştirmek için *71* sayısını en sağdaki iki sayıda göstermek istediğini varsay. Denetleyici devre *AN0*, *CB*, ile *CC’yi* ilk 2ms için düşüğe sürer, böylece en sağdaki sayıda *1* gösterilir. Ardından sonraki 2ms için devre *AN1*, *CA*, *CB*, ile *CC’yi* düşüğe sürer, böylelikle sonraki yüksek önemli sayıda 7’yi gösterir. Bu süreç sürekli tekrarlanırsa insan gözü en sağdaki iki sayıda *71* sayısını görür.

# 8-SAYI 7-KESİMLİ EKRAN DENETLEYİCİSİNİN YÜKSEK DÜZEYLİ SPESİFİKASYONU

Bu bölümde ilk olarak RVfpga’de kullanılan 8-sayı 7-kesimli ekran denetleyicisinin yüksek düzeyli spesifikasyonunu tanımlayıp, çözümleyip, ardından onu kullanmak için alıştırmalar sağlıyoruz.

1. **Yüksek düzeyli spesifikasyon**

Bu kursta kullanılan 8-sayı 7-kesimli ekran denetleyicisi RVfpga için özel tasarlanmıştır. İki yazmaç içerir, *Enables\_Reg* ile *Digits\_Reg*, bunlar sırasıyla 0x80001038 ile 0x8000103C adreslerine eşlenmiştir (önemli olarak bu adresler Sistem Denetleyicisi için ayrılmış adres aralığında kullanılmayan adreslerdir, şuradan bakabilirsin <https://github.com/chipsalliance/Cores-SweRVolf>).

**GÖREV:** *Enables\_Reg* ile *Digits\_Reg* yazmaçlarının bildirildiği, değer atandığı yerleri bul. 8-sayı 7-kesimli ekran şu dosyada gerçekleştirilmiştir: *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*.

*Enables\_Reg* bitlerin karşılık gelen sayının *ON* (0) ya da *OFF* (1) olduğunu belirleyen 8-bitlik bir yazmaçtır. *Digits\_Reg* ise 4-bitlik grupların karşılık gelen sayıda gösterilecek değeri belirlediği 32-bitlik bir yazmaçtır. Örneğin en sağdaki iki sayıda 71’i göstermek için programlamacı şu değerleri yazmaçlara atar:

* *Enables\_Reg = 0xFC* (en sağdaki iki sayı etkinleştirilmiş)
* *Digits\_Reg = 0x00000071* (değer = 71)

# TEMEL ALIŞTIRMALAR

# Alıştırma 1. Anahtarların değerini 7-kesimli ekranın en sağdaki dört sayısında göstern bir RISC-V çevirici programı ile bir C programı yaz.

# Alıştırma 2. “0-1-2-3-4-5-6-7-8” dizisini 8-sayı 7-kesimli ekranlarda sağdan sola doğru kayan biçimde gösteren bir RISC-V çevirici programı ile C programı yaz. Yani 0 ilk baştan en sağdaki sayıda görünmelidir. Sonrasında sola kayıp en sağda 1 olmalı, gibi gibi.

# 8-SAYI 7-KESİMLİ EKRAN DENETLEYİCİSİ: ALÇAK DÜZEYLİ GERÇEKLEŞTİRME, SİMÜLASYON

# Buraya değiin 8-sayı 7-kesimli ekranın yalnızca kullanımını gösterdik. Bu bölümde alçak düzeyli gerçekleştirmesini tanımlayıp basit bir çevirici örneği yürütürken RVfpga’i simülasyonda çözümlüyoruz. Son olarak ise 8-sayı 7-kesimli ekranların değiştirildiği alıştırmalar sağlıyoruz.

1. **8-sayı 7-kesim ekran denetleyicisinin alçak düzeyli gerçekleştirmesi**

Önceki genel amaçlı I/O (GPIO) deneylerindeki gibi 8-sayı 7-kesimli ekran denetleyicisinin çözümlemesini üç faza bölüyoruz:

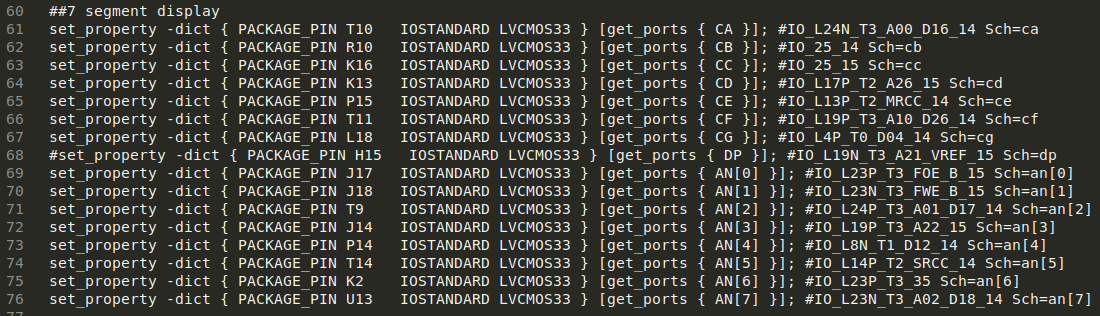
1. SoC ile karttaki I/O aygıtı arasındaki bağlantı (Figür 4’te sol taralı bölge);
2. Yeni denetleyicinin entegrasyonu, SoC’deki SweRVolf Sistem Denetleyicisnde içerilir (Figür 4’te orta taralı bölge);
3. Yeni denetleyici ile SweRV EH1 Çekirdeği arasındaki bağlantı (Figür 4’te sağ taralı bölge).



Figür 4. 3 fazda 8-sayı 7-kesimli ekran denetleyicisinin çözümü

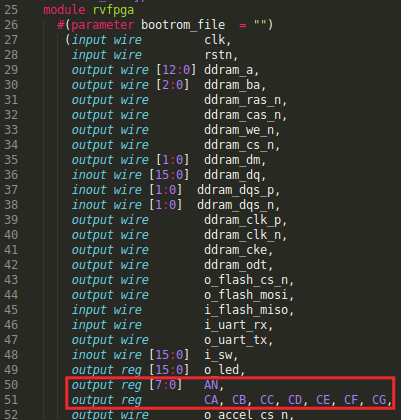
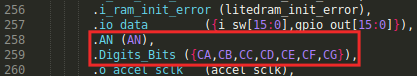
1. **LEDlerin/Anahtarların SoC’ye Bağlantısı**

Projenin kısıtlandırmalar dosyası (*[RVfpgaPath]/RVfpga/src/RVfpga.xdc*) girdi/çıktı SoC sinyalleri ile kart arasındaki bağlantıyı tanımlar. Nexys A7 FPGA kartındaki I/O aygıtlarının hepsi ayrı ayrı belirli bir FPGA ucuna bağlıdır. Sekiz anodu bağlayan sinyale (Figür 3’e göz at) *AN[i]* denir (*i* 0-7 aralığındadır), 8 sayının hepsinde yakın kesimlerin katotlarını bağlayan sinyallere (Figür 3’e göz at) *CA, CB, CC, CD, CE, CF*, *CG* denir. Figür 5 kısıtlandırmalar dosyasında bağlantıların tanımlandığı bölümü gösterir.



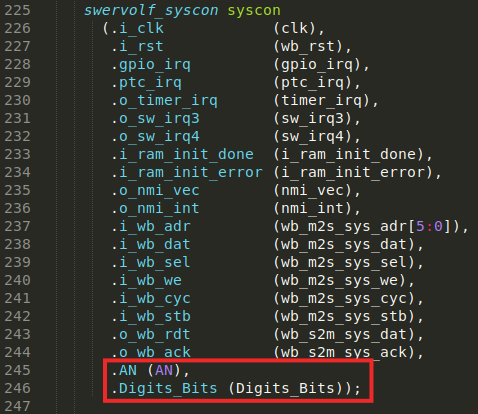
Figür 5. 8-sayı 7-kesimli ekran girdilerinin bağlantısı (dosya *RVfpga.xdc*).

Sistemimizin üst modülünün 50-51 numaralı satırlarında (modül RVfpga, [RVfpgaPath] /RVfpga/src/RVfpga.sv dosyasında uygulanmıştır) SoC'ye bağlı 8-sayı 7-kesimli ekran giriş sinyallerini bulabilirsiniz, AN [i] ve CA… CG (Figür 6'nın sol kısmı) ve bu modülün sonunda (Figür 6'nın sağ kısmı) swervolf\_core modülüne bağlantılarını bulabilirsin (CA…CG sinyallerini bu modül Digits\_Bits [6:0] olarak adlandırmıştır).

Figür 6. 8-sayı 7-kesimli ekranın SoC’ye bağlantısı (dosya: *rvfpga.sv*).

Son olarak, iki sinyal swervolf\_core modülünden, 8 basamaklı 7 segmentli ekran denetleyicisinin uygulandığı Sistem Denetleyici modülüne (swervolf\_syscon) (Figür 7) yerleştirilir.



Figür 7. 8-sayı 7-kesimli ekranın Sistem Denetleyicisine bağlantısı (dosya: *SweRVolfCore.v*).

**GÖREV:** Bu sinyalleri (CA-CG ve AN) kısıtlamalar dosyasından Sistem Denetleyici modülüne kadar izle (burada CA-CG, Digits\_Bits dizisiyle birleştirilir). Aşağıdaki dosyaları incelemen gerekecek:

*[RVfpgaPath]/RVfpga/src/RVfpga.xdc*

*[RVfpgaPath]/RVfpga/src/RVfpga.sv*

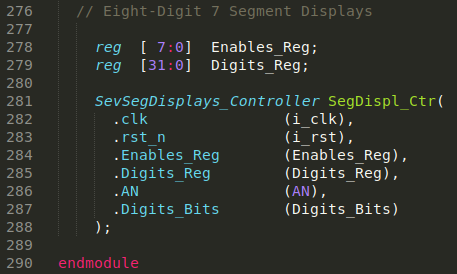
*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf\_core.v*

*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*

RI

1. **8-sayı 7-kesimli ekran denetleyicisinin SoC’ye entegrasyonu**

**swervolf\_syscon** (*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*) modülünün 276-288 satırlarında 8-sayı 7-kesimli ekran denetleyicisi somutlaması yapılıp SoC’ye entegre edilir (Figür 8’e göz at).



Figür 8. 8-sayı 7-kesimli ekran controller instantiation (file: *swervolf\_syscon.v*).

SevSegdisplays\_Controller modülü, saat sinyaline (i\_clk, clk olarak yeniden adlandırılır) ve sıfırlama sinyaline (i\_rst, rst\_n olarak yeniden adlandırılır) ek olarak, önceden açıklanan iki bellek eşlemeli kontrol kaydı olan iki giriş sinyali (Enables\_Reg ve Digits\_Reg) alır. Bu modül, karttaki 7 kesimli ekranlara bağlanan AN ve Digits\_Bits olmak üzere iki sinyal çıkarır. En sağdaki iki basamakta 71'i gösteren örnek için, SevSegdisplays\_Controller AN ve Basamak\_Bitleri sinyallerine aşağıdaki değerleri atayacaktır:

* 0’dan 2ms’e: Signal *AN[0]* is low to enable digit 0 (the right-most digit) to display. Signals *Digits\_Bits[5]* and *Digits\_Bits[4]* (that correspond to *CB* and *CC*) are also low to display “1” on digit 0 (the right-most digit). All other signals are high.
* 2’den 4ms’e: Signal *AN[1]* is low to enable digit 1 to display. *Digits\_Bits[6], Digits\_Bits[5]* and *Digits\_Bits[4]* (that correspond to *CA, CB,* and *CC*) are high to display “*7”* on digit 1. All other signals are high.
* 4’ten 16ms’e: *AN[2]*…*AN[7]* are high in 2 ms intervals so that they do not display values. The segments are also high for the remaining digits, digits 2-7.

**SevSegdisplays\_Controller** modülü şu dosyanın 295-366 satırlarında gerçekleştirilmiştir *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*. Şu alt üniteleri barındırır:

* 2ms’de bir *AN* ile *Digits\_Bits* sinyallerine gönderilecek değeri seçen iki çoklayıcı. Çoklayıcı **SevSegMux** modülünün içerisinde gerçekleştirilmiştir.
* For creating the 2ms period, we use a **counter** module provided in files *counter.sv* and *delta\_counter.sv*, both included in folder *[RVfpgaPath]/RVfpga/src/OtherSources/PulpPlatform/pulp-platform.org\_\_common\_cells\_1.16.4/src*. The counter is configured to count from 0 to 219, and the 3 most significant bits, which change approximately every 2ms, are used as the select signals for the two multiplexers described above.
* **SevenSegDecoder** modülü içerisinde bir çözücü gerçekleştirilir, ki bu verilen 4-bit onaltılık değer için kesim değerlerini çıkarır.

**GÖREVLER: SevSegdisplays\_Controller** modülünü detaylıca çözümle. Sonraki bölümde yapılan simülasyon sana bu görevde yardımcı olabilir. Gerekirse simülasyonu yeni sinyallerle genişletebilirsin.

1. **8-sayı 7-kesimli ekran denetleyicisi ile SweRV EH1 Çekirdeği arasındaki bağlantı**

Deney 6’da tanımlandığı gibi aygıt denetleyicileri SweRV EH1 Çekirdeğine bir çoklayıcı kullanarak bağlanır (Figür 4’e göz at). 7:1 çoklayıcının (Figür 9) şu dosyada gerçekleştirildiğini anımsa *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb\_intercon.v*, şu dosyanın 104-205 satırlarında somutlaması yapılır *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb\_intercon.vh*. İkinci dosya şuradaki **swervolf\_core** modülünün 168 satırında içerilir: *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf\_core.v*.

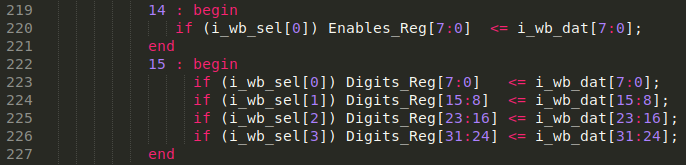
Çoklayıcı, adrese (satır 110-111) bağlı olarak, bir çevre biriminin (Figür 9'un 127-138 satırları) Wishbone Veri Yolu ile CPU'yu *(wb\_io\_\** sinyalleri – Figür 9'un 115-126 satırları) birbirine bağlayarak hangi çevre biriminin okunup yazılması gerektiğini seçer. Örneğin, CPU tarafından oluşturulan adres 0x80001000-0x8000103F aralığındaysa, Sistem Denetleyicisi seçilir, böylece *wb\_io\_\** sinyalleri wb\_sys\_\* ile bağlanır.



Figür 9. CPU’ya bağlı çevre birimini seçen 7-1 çoklayıcı (dosya: *wb\_intercon.v*).

Sistem Denetleyicisinde yer alan yazmaçlar, CPU tarafından oluşturulan adrese (i\_wb\_adr) (swervolf\_syscon 162-228) bağlı olarak, doğrudan Wishbone Veri Yolu *(i\_wb\_dat)* veri sinyaline bağlayarak CPU'dan yazılır.

**GÖREV:** Adreslerin Sistem Denetleyicisinde nasıl eşlendiğini anlamak için modül **swervolf\_syscon** 162-228 satırlarını incele. *Enables\_Reg*, *Digits\_Reg* yazmaçları anlamına gelen 219-227 (Figür 10) satırlarına odaklan (daha önce de belirttiğimiz gibi, bu iki yazmaç için atanan adresler sırasıyla 0x80001038, 0x8000103C).



Figür 10. 8-sayı 7-kesimli ekran ile Çekirdek arasındaki bağlantı (dosya *swervolf\_syscon.v*).

1. **Verilator Simulation**

Bu bölümde, işlemci bu çevre birimini çalıştıran basit bir örnek yürüttüğünde 8 sayı 7 kesimli ekran denetleyicisinin ana sinyallerini incelemek için RVfpgaSIM kullanıyoruz. Simülasyonda, en sağdaki iki basamağa 71 yazan Figür 11'deki örneği yürütürken AN ve Digits\_Bits sinyallerini analiz ediyoruz. Bu programı şu adreste bulabilirsin: [RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl (C sürümünü ise şurada bulabilirsin: [RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl\_C-Lang).

|  |
| --- |
| #define SegEn\_ADDR 0x80001038  #define SegDig\_ADDR 0x8000103C  .globl main  main:  li x1, SegEn\_ADDR  li x6, 0xFC  **sb x6, 0(x1)** # Enable the 7Segdisplays  li x1, SegDig\_ADDR  li x6, 0x71  **sw x6, 0(x1)** # Write the 7Segdisplays  next: beq zero, zero, next  .end |

Figür 11. 71\_7SegDispl.S örneği

Figür 12, 71\_7SegDispl.elf programının tersine çeviri sürümünü gösterir, ki bunu, PlatformIO’da derledikten sonra, şurada bulabilirsin: [RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys/firmware.dis

|  |
| --- |
| 00000090 <main>:  90: 800010b7 lui ra,0x80001  94: 03808093 addi ra,ra,56 # 80001038  98: 0fc00313 li t1,252  9c: 00608023 **sb t1,0(ra)**  a0: 800010b7 lui ra,0x80001  a4: 03c08093 addi ra,ra,60 # 8000103c  a8: 07100313 li t1,113  ac: 0060a023 **sw t1,0(ra)**  000000b0 <next>:  b0: 00000063 beqz zero,b0 <next> |

Figür 12. 71\_7SegDispl.S örneğinin tersine çeviri sürümü

# Simülasyonu çalıştırmak için sonraki adımları izle. (Simülasyonu çalıştırmamayı seçersen doğrudan 7. adıma geçebilirsin.)

1. Bu durumda, yalnızca simülasyon için, COUNT\_MAX (şurada satır 295 *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*) değerini 20’den 5’e değiştirerek saat periyotunu azaltmalısın; yoksa sonuçları görmek çok uzun sürer. COUNT\_MAX değerini değiştirip ardından RVfpgaSIM’i şu komutları yürüterek yeniden derle (bu GSG’de (İlk Kullanım Kılavuzu) açıklanmıştı):

cd [RVfpgaPath]/RVfpga/verilatorSIM

make clean

make

Yeni bir dosya olarak *Vrvfpgasim* (RVfpga simulation binary), şu dizinde oluşturulmuş olmalı *[RVfpgaPath]/RVfpga/verilatorSIM*.

**WINDOWS:** Windows kullanıyorsan, bu komutları Cygwin terminalinde yürütmelisin (ayrıntılı yönergeler için İlk Kullanım Kılavuzundaki Bölüm 6 ile Ek C'ye bak). C: Windows klasörünün Cygwin içinde şu adreste bulunabileceğini unutma: */cygdrive/c*.

**MacOS:** Detaylı yönergeler için İlk Kullanım Kılavuzunun Ek B’sine bak.

1. VSCode/PlatformIO’yu bilgisayarında aç.
2. Üst çubukta *File* (Dosya) - *Open Folder...* (Klasörü Aç...) tıklayıp şu dizine git *[RVfpgaPath]/RVfpga/Labs/Lab7*
3. *71\_7SegDispl* dizinini seç (açma, yalnızca seç), OK’a tıkla. Örnek PlatformIO’da açılacaktır.
4. *platformio.ini* dosyasını açıp RVfpga simülasyon ikilisine giden yolun doğru olup olmadığını denetle (**Error! Reference source not found.**) (önceki bölümde adım 3). GSG’den anımsayacağın üzere gibi şöyle görünmelidir:

board\_debug.verilator.binary = [RVfpgaPath]/RVfpga/verilatorSIM/Vrvfpgasim

1. Sol menü şeridinde PlatformIO ikonuna tıklayarak simülasyonu çalıştır , ardından Project Tasks → env:swervolf\_nexys → Platform genişletip Generate Trace (İz Oluştur) tıkla.

*trace.vcd* dosyası şurada oluşturulmuş olmalı *[RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys*, bunu GTKWave’de şu komutu yürüterek açabilirsin:

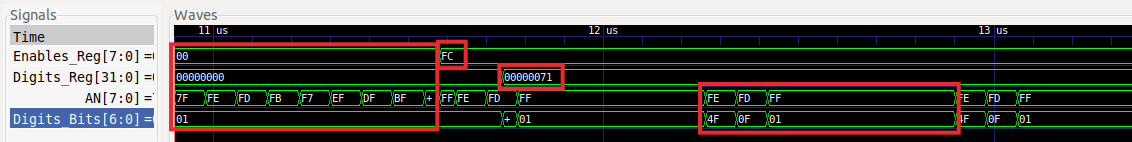
gtkwave [RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys/trace.vcd

**WINDOWS:** indirdiğin *gtkwave64* klasörü *bin* klasörünün içerisinde *gtkwave.exe* adında bir uygulama içerir. GTKWave’i o uygulamaya çift tıklayarak çalıştır. Uygulamanın üstünde **File (Dosya) – Open New Tab (Yeni Sekme Aç)** tıklayıp, *[RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys* klasöründe oluşturulan trace.vcd dosyasını aç.

1. Şu sinyalleri simülasyonda içer (sinyallerin yerini bulmak için adı verilen modüllere git):

* rvfpgasim – swervolf – syscon – SegDispl\_Ctr
  + Girdi sinyalleri: ***Enables\_Reg***, ***Digits\_Reg***.
  + Çıktı sinyalleri: ***AN***, ***Digits\_Bits***.

1. Figür 13’te gösterilen simülasyonu çözümle. İlk olarak 7-kesimli ekranda gösterilen değerlerin hepsi 0 (ilk başta bütün sayılar *Enables\_Reg*=0 olduğundan etkindir). Ardından en soldaki altı sayıyı *0xFC’ı* *Enables\_Reg’a* yazarak devre dışı bırakıyoruz (sb yönergesi, Figür **12**), en sağdaki iki sayıya *71* değerini *0x71’i* *Digits\_Reg’a* yazarak iletiyoruz (sw yönergesi, Figür **12**). Çıktı sinyallerindeki etki şöyledir (Figür 13’te gösterildiği gibi):
   * İlk periyotta: *AN=0xFE*, *Digits\_Bits=0x4F*, böylelikle en sağdaki sayı, sayı 0’ta, 1 gösterilir.
   * İkinci periyotta: *AN=0xFD*, *Digits\_Bits=0x0F*, böylelikle sonraki sayıda, sayı 1’de, 7 gösterilir.
   * Sonraki altı periyotta: *AN=0xFF*, *Digits\_Bits=0x01*, böylelikle en soldaki altı sayı kapanır.
   * Bu işlem ardından tekrarlar.



Figür 13. 8-sayı 7-kesim ekranların en sağdaki iki sayısında 71 değerini yaz

1. İlerlemeden önce COUNT\_MAX’ın değerini eski değerine döndürmeyi unutma (COUNT\_MAX=20).

# İLERİ DÜZEY ALIŞTIRMALAR

# Alıştırma 3. Bu deneyde tanımlanan denetleyiciyi 8-sayı 7-kesimli ekranın ON/OFF LEDlerin bütün kombinasyonlarını gösterebileceği biçimde değiştir.

* Artık bir etkinleştirme yazmacına gerek yoktur. Onun yerine 8 7-bit yazmaç gerekir. Şöyle adlandır: Segments\_Digit0 – Segments\_Digit7, 7-kesimli ekran başına bir yazmaç, bir bit karşılık gelen kesimin ON (0) ya da OFF (1) olduğnu belirler. Örneğin ilk yazmacın bütün bitleri 0’sa (Segments\_Digit0), en sağdaki sayının bütün kesimleri ON olacaktır, ancak ilk yazmacın bütün bitleri 1’se, en sağdaki yazının bütün kesitleri OFF olacaktır.
* Bu yeni iki yazmacı önceden kullandığımız adreslere eşleyebilirsin (öncelikle önceki iki yazmaç *Enables\_Reg’i*, *Digits\_Reg’i kaldır*):
  + Segments\_Digit0 🡨🡪 Adres 0x80001038
  + Segments\_Digit1 🡨🡪 Adres 0x80001039
  + …
  + Segments\_Digit7 🡨🡪 Adres 0x8000103F
* Önemli olarak artık 4-7 çözücüsü gerekli değildir (modül **SevenSegDecoder**), burada programın sağladığı bilgi çözülü biçimdedir.

# Alıştırma 4. 8-sayı 7-kesimli ekranda şunu yazdırmak için yeni denetleyiciyi kullan: “I SAY HI”. Önceden olduğu gibi programın RISC-V çevirici sürümünü de C sürümünü de gerçekleştir.