



THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga Deney 7

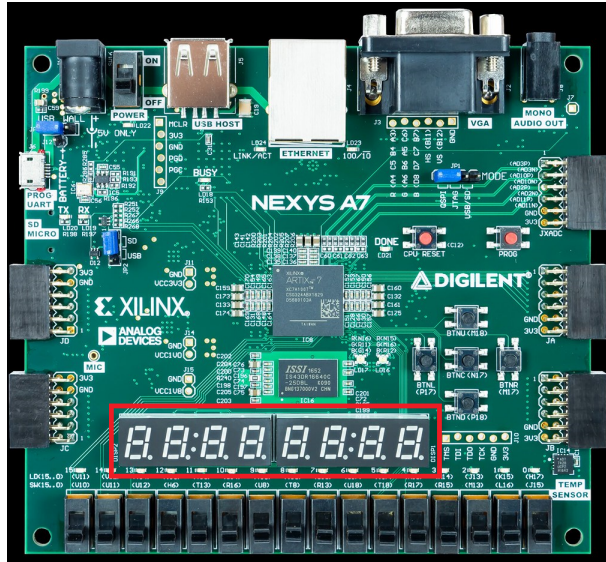
7-Kesimli Ekranlar

1. GİRİŞ

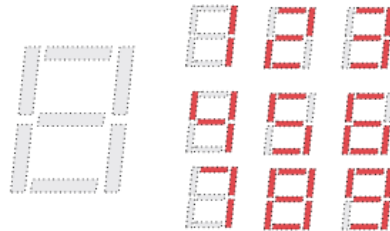
Bu deney RVfpga'in 7-kesimli ekranlarla çalışmak üzere nasıl genişletildiğini tanımlayıp 7-kesimli ekran denetleyicisini nasıl değiştirileceğini gösterir. Nexys A7 FPGA kartının sekiz 7-kesimli ekranı vardır. İlk nasıl çalıştıklarını tanımlayıp (Bölüm 2) ardından RVfpga'de içerilen 8-sayı 7-kesimli ekranın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

2. 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.

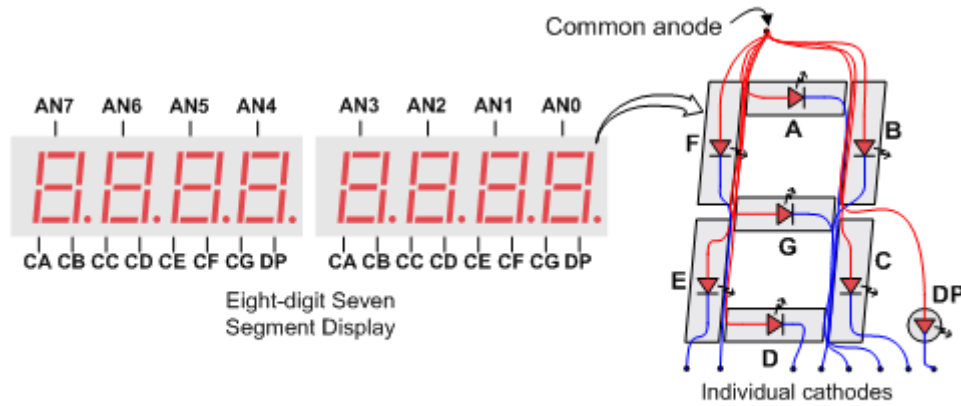


Figür 1. Nexys A7'daki 8-sayı 7-kesimli ekran



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 etinken 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 etinken 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.

3. 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.

A. 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 bakabilirsiniz <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)

4. 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.

5. 8-SAYI 7-KESİMLİ EKRAN DENETLEYİCİSİ: ALÇAK DÜZEYLİ GERÇEKLEŞTİRME, SİMÜLASYON

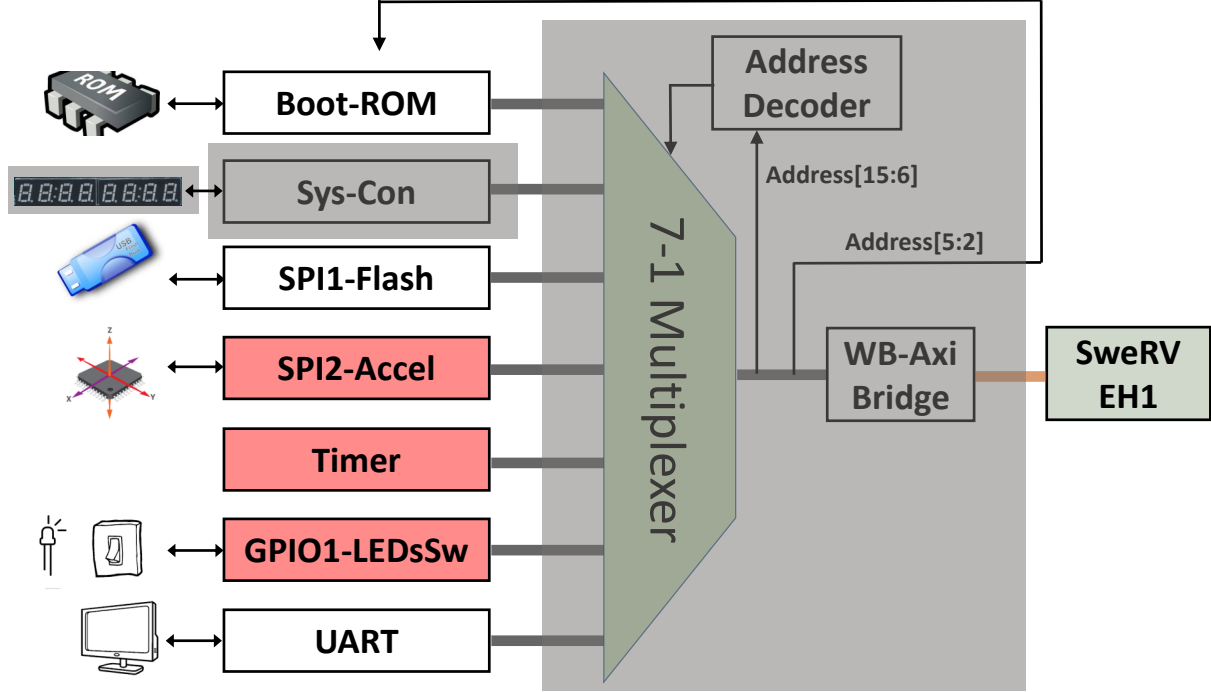
Buraya değiin 8-sayı 7-kesimli ekranın yalnızca kullanımını gösterdik. Bu bölümde alçak düzeyli gerçekleştirmesini tanımlayıp basit bir çevirici örneği yürütürken RVfpga'i simülasyonda çözümlüyoruz. Son olarak ise 8-sayı 7-kesimli ekranların değiştirildiği alıştırmalar sağlıyoruz.

A. 8-sayı 7-kesim ekran denetleyicisinin alçak düzeyli gerçekleştirilmesi

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 Denetleyicisinde 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/çıkış 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.

```

60 ##7 segment display
61 set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33 } [get_ports { CA }]; #IO_L24N_T3_A00_D16_14 Sch=ca
62 set_property -dict { PACKAGE_PIN R10 IOSTANDARD LVCMOS33 } [get_ports { CB }]; #IO_25_14 Sch=cb
63 set_property -dict { PACKAGE_PIN K16 IOSTANDARD LVCMOS33 } [get_ports { CC }]; #IO_25_15 Sch=cc
64 set_property -dict { PACKAGE_PIN K13 IOSTANDARD LVCMOS33 } [get_ports { CD }]; #IO_L17P_T2_A26_15 Sch=cd
65 set_property -dict { PACKAGE_PIN P15 IOSTANDARD LVCMOS33 } [get_ports { CE }]; #IO_L13P_T2_MRCC_14 Sch=ce
66 set_property -dict { PACKAGE_PIN T11 IOSTANDARD LVCMOS33 } [get_ports { CF }]; #IO_L19P_T3_A10_D26_14 Sch=cf
67 set_property -dict { PACKAGE_PIN L18 IOSTANDARD LVCMOS33 } [get_ports { CG }]; #IO_L4P_T0_D04_14 Sch=cg
68 #set_property -dict { PACKAGE_PIN H15 IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
69 set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_F0E_B_15 Sch=an[0]
70 set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
71 set_property -dict { PACKAGE_PIN T9 IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
72 set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
73 set_property -dict { PACKAGE_PIN P14 IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
74 set_property -dict { PACKAGE_PIN T14 IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
75 set_property -dict { PACKAGE_PIN K2 IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
76 set_property -dict { PACKAGE_PIN U13 IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]

```

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 bulabilirsiniz (CA...CG sinyallerini bu modül Digits_Bits [6:0] olarak adlandırmıştır).

```

25 module rvfpga
26     #(parameter bootrom_file = "")
27     (input wire      clk,
28      input wire      rstn,
29      output wire [12:0] ddr_a,
30      output wire [2:0] ddr_ba,
31      output wire      ddr_ras_n,
32      output wire      ddr_cas_n,
33      output wire      ddr_we_n,
34      output wire      ddr_cs_n,
35      output wire [1:0] ddr_dm,
36      inout wire [15:0] ddr_dq,
37      inout wire [1:0] ddr_dqs_p,
38      inout wire [1:0] ddr_dqs_n,
39      output wire      ddr_clk_p,
40      output wire      ddr_clk_n,
41      output wire      ddr_cke,
42      output wire      ddr_odt,
43      output wire      o_flash_cs_n,
44      output wire      o_flash_mosi,
45      input wire      i_flash_miso,
46      input wire      i_uart_rx,
47      output wire      o_uart_tx,
48      inout wire [15:0] i_sw,
49      output reg [15:0] o_led,
50      output reg [7:0] AN,
51      output reg      CA, CB, CC, CD, CE, CF, CG,
52      output wire      o_accel_cs_n,

```

```

256 .i_ram_init_error (litedram_init_error),
257 .io_data          ({i_sw[15:0], gpio_out[15:0]}),
258 .AN (AN),
259 .Digits_Bits      ({CA, CB, CC, CD, CE, CF, CG}),
260 .o_accel_sclk     (accel_sclk),

```

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.

```

225 swervolf_syscon syscon
226     (.i_clk      (clk),
227      .i_rst      (wb_rst),
228      .gpio_irq   (gpio_irq),
229      .ptc_irq    (ptc_irq),
230      .o_timer_irq (timer_irq),
231      .o_sw_irq3   (sw_irq3),
232      .o_sw_irq4   (sw_irq4),
233      .i_ram_init_done (i_ram_init_done),
234      .i_ram_init_error (i_ram_init_error),
235      .o_nmi_vec    (nmi_vec),
236      .o_nmi_int    (nmi_int),
237      .i_wb_adr     (wb_m2s_sys_adr[5:0]),
238      .i_wb_dat     (wb_m2s_sys_dat),
239      .i_wb_sel     (wb_m2s_sys_sel),
240      .i_wb_we      (wb_m2s_sys_we),
241      .i_wb_cyc     (wb_m2s_sys_cyc),
242      .i_wb_stb     (wb_m2s_sys_stb),
243      .o_wb_rdt     (wb_s2m_sys_dat),
244      .o_wb_ack     (wb_s2m_sys_ack),
245      .AN (AN),
246      .Digits_Bits (Digits_Bits));
247

```

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
```

2. 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 yapıp SoC'ye entegre edilir (Figür 8'e göz at).

```
276 // Eight-Digit 7 Segment Displays
277
278 reg [ 7:0] Enables_Reg;
279 reg [31:0] Digits_Reg;
280
281 SevSegDisplays_Controller SegDispl_Ctr(
282     .clk           (i_clk),
283     .rst_n         (i_rst),
284     .Enables_Reg   (Enables_Reg),
285     .Digits_Reg    (Digits_Reg),
286     .AN            (AN),
287     .Digits_Bits   (Digits_Bits)
288 );
289
290 endmodule
```

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.

3. 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 Figür 9Figür Figür 9 seçer. Örneğin, CPU tarafından oluşturulan adres 0x80001000-0x8000103F aralığındaysa, Sistem Denetleyicisi seçilir, böylece *wb_io_** sinyalleri *wb_sys_** ile bağlanır.

```

108 wb_mux
109 #(.num_slaves (7),
110   .MATCH_ADDR ({32'h00000000, 32'h00001000, 32'h00001040, 32'h00001100, 32'h00001200, 32'h00001400, 32'h00002000}),
111   .MATCH_MASK ({32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffff00}))
112 wb_mux_io
113 (.wb_clk_i (wb_clk_i),
114   .wb_rst_i (wb_rst_i),
115   .wbm_adr_i (wb_io_adr_i),
116   .wbm_dat_i (wb_io_dat_i),
117   .wbm_sel_i (wb_io_sel_i),
118   .wbm_we_i (wb_io_we_i),
119   .wbm_cyc_i (wb_io_cyc_i),
120   .wbm_stb_i (wb_io_stb_i),
121   .wbm_cti_i (wb_io_cti_i),
122   .wbm_bte_i (wb_io_bte_i),
123   .wbm_dat_o (wb_io_dat_o),
124   .wbm_ack_o (wb_io_ack_o),
125   .wbm_err_o (wb_io_err_o),
126   .wbm_rty_o (wb_io_rty_o),
127   .wbs_adr_o (wb_sys_adr_o, wb_spi_flash_adr_o, wb_spi_accel_adr_o, wb_ptc_adr_o, wb_gpio_adr_o, wb_uart_adr_o),
128   .wbs_dat_o (wb_rom_dat_o, wb_sys_dat_o, wb_spi_flash_dat_o, wb_spi_accel_dat_o, wb_ptc_dat_o, wb_gpio_dat_o, wb_uart_dat_o),
129   .wbs_sel_o (wb_rom_sel_o, wb_sys_sel_o, wb_spi_flash_sel_o, wb_spi_accel_sel_o, wb_ptc_sel_o, wb_gpio_sel_o, wb_uart_sel_o),
130   .wbs_we_o (wb_rom_we_o, wb_sys_we_o, wb_spi_flash_we_o, wb_spi_accel_we_o, wb_ptc_we_o, wb_gpio_we_o, wb_uart_we_o),
131   .wbs_cyc_o (wb_rom_cyc_o, wb_sys_cyc_o, wb_spi_flash_cyc_o, wb_spi_accel_cyc_o, wb_ptc_cyc_o, wb_gpio_cyc_o, wb_uart_cyc_o),
132   .wbs_stb_o (wb_rom_stb_o, wb_sys_stb_o, wb_spi_flash_stb_o, wb_spi_accel_stb_o, wb_ptc_stb_o, wb_gpio_stb_o, wb_uart_stb_o),
133   .wbs_cti_o (wb_rom_cti_o, wb_sys_cti_o, wb_spi_flash_cti_o, wb_spi_accel_cti_o, wb_ptc_cti_o, wb_gpio_cti_o, wb_uart_cti_o),
134   .wbs_bte_o (wb_rom_bte_o, wb_sys_bte_o, wb_spi_flash_bte_o, wb_spi_accel_bte_o, wb_ptc_bte_o, wb_gpio_bte_o, wb_uart_bte_o),
135   .wbs_dat_i (wb_rom_dat_i, wb_sys_dat_i, wb_spi_flash_dat_i, wb_spi_accel_dat_i, wb_ptc_dat_i, wb_gpio_dat_i, wb_uart_dat_i),
136   .wbs_ack_i (wb_rom_ack_i, wb_sys_ack_i, wb_spi_flash_ack_i, wb_spi_accel_ack_i, wb_ptc_ack_i, wb_gpio_ack_i, wb_uart_ack_i),
137   .wbs_err_i (wb_rom_err_i, wb_sys_err_i, wb_spi_flash_err_i, wb_spi_accel_err_i, wb_ptc_err_i, wb_gpio_err_i, wb_uart_err_i),
138   .wbs_rty_i (wb_rom_rty_i, wb_sys_rty_i, wb_spi_flash_rty_i, wb_spi_accel_rty_i, wb_ptc_rty_i, wb_gpio_rty_i, wb_uart_rty_i));
139
140 endmodule

```

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. Figür 10 *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).


```

219      14 : begin
220          if (i_wb_sel[0]) Enables_Reg[7:0] <= i_wb_dat[7:0];
221      end
222      15 : begin
223          if (i_wb_sel[0]) Digits_Reg[7:0] <= i_wb_dat[7:0];
224          if (i_wb_sel[1]) Digits_Reg[15:8] <= i_wb_dat[15:8];
225          if (i_wb_sel[2]) Digits_Reg[23:16] <= i_wb_dat[23:16];
226          if (i_wb_sel[3]) Digits_Reg[31:24] <= i_wb_dat[31:24];
227      end

```

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

B. 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 bulabilirsiniz:

[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl (C sürümünü ise burada bulabilirsiniz: [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, burada bulabilirsiniz:

[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.

2. VSCode/PlatformIO'yu bilgisayarında aç.
3. Üst çubukta *File* (Dosya) - *Open Folder...* (Klasörü Aç...) tıklayıp şu dizine git `[RVfpgaPath]/RVfpga/Labs/Lab7`
4. `71_7SegDispl` dizinini seç (açma, yalnızca seç), OK'a tıkla. Örnek PlatformIO'da açılacaktır.
5. `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:

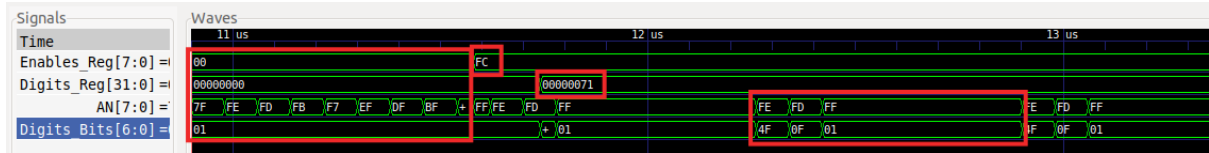
```
board_debug.verilator.binary =
[RVfpgaPath]/RVfpga/verilatorSIM/Vrvfpgasim
```
6. Sol menü şeridinde PlatformIO ikonuna tıklayarak simülasyonu çalıştır , ardından Project Tasks → env:swervolf_nexys → Platform genişletip Generate Trace (İz Oluştur) tıkla.

`trace.vcd` dosyası burada 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.

7. 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.**
8. Figür 13'te gösterilen simülasyonu çözümü. İlk olarak 7-kesimli ekranda gösterilen değerlerin hepsi 0 (ilk başta bütün sayılar *Enables_Reg*=0 olduğundan etkindir). Ardından en soldaki altı sayıyı *0xFC*'i *Enables_Reg*'a yazarak devre dışı bırakıyoruz (*s_b* yönergesi, Figür 12), en sağdaki iki sayıya *71* değerini *0x71*'i *Digits_Reg*'a yazarak iletiyoruz (*s_w* 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

9. İlerlemeden önce *COUNT_MAX*'ın değerini eski değerine döndürmeyi unutma (*COUNT_MAX*=20).

6. 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) olduğunu 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 kesimleri 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 \leftrightarrow 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.