

RVfpga

Bilgisayar Mimarisini Anlamanın Eksiksiz Kursu

Sayılanlar

YAZARLAR

Prof. Sarah Harris
Prof. Daniel Chaver

DANIŞMAN

Prof. David Patterson

KATKIDA BULUNANLAR

Robert Owen
Zubair Kakakhel
Olof Kindgren
Prof. Luis Piñuel
Ivan Kravets
Valerii Koval
Ted Marena
Prof. Roy Kravitz

ORTAKLAR

Prof. Daniel León	Prof. Christian Tenllado	Gage Elerding
Prof. José Ignacio Gómez	Prof. Francisco Tirado	Prof. Brian Cruickshank
Prof. Katzalin Olcoz	Prof. Román Hermida	Deepen Parmar
Prof. Alberto del Barrio	Cathal McCabe	Thong Doan
Prof. Fernando Castro	Dan Hugo	Oliver Rew
Prof. Manuel Prieto	Braden Harwood	Niko Nikolay
Prof. Ataur Patwary	Prof. David Burnett	Guanyang He

Sponsorlar - Destekleyenler

Western Digital

 **Imagination**

 **CHIPS
ALLIANCE**

 **RISC-V**

 **DIGILENT**
A National Instruments Company

 **XILINX**
UNIVERSITY PROGRAM

 **Digi-Key**
ELECTRONICS

 **Esperanto**
TECHNOLOGIES

 **codasip**

 **硬禾学堂**

 **ANDES**
TECHNOLOGY

 **PLATFORMIO.ORG**

Bu içeriğin ilk çevirisini
Mehmet Oguz Derin
yapmıştır.

Görüş ile önerilerinizi
rvfpga@mehmetoguzderin.com
adresine iletebilir, çevirmene
[@mehmetoguzderin](https://twitter.com/mehmetoguzderin)
Twitter kullanıcı adından ulaşabilirsiniz.

Bu içeriğin ilk testi
Cerrahpaşa Tıp Fakültesi
öğrencisi
Hüseyin Bora Gürer
ile yapılmıştır.

Giriş

- RISC-V FPGA (**RVfpga**) yönergeler, araçlar, deneyler sağlayan bir öğretim paketi olarak şu konular üzerine yoğunlaşır:
 - Ticari bir RISC-V yongadaki sistem (SoC) bir **FPGA**'e nasıl **hedeflenir**
 - RISC-V SoC nasıl **programlanır**
 - RISC-V SoC'e nasıl **daha çok işlevsellik eklenir**
 - RISC-V çekirdek ile bellek hiyerarşisi nasıl **çözümlemlenip değiştirilir**
- Bu paket **Imagination Technologies** ile akademi, endüstri ortakları birlikteliğinde geliştirilmektedir.
- RVfpga Western Digital'ın RISC-V **SweRV EH1** çekirdeğine dayalı Chips Alliance'ın **SweRVolf SoC**'si ekseninde kurgulanmıştır.

RVfpga Tanıtımı

- **RVfpga Paketi** şunları sağlar:
 - kapsamlı, ücretsiz dağıtılan, eksiksiz bir **RISC-V** kursu
 - RISC-V işlemcilerle RISC-V ekosistemini öğrenmek için **uygulamalı, kolayca erişilebilir** bir yol
 - birçok üniversiteyle şirkette bulunan **düşük maliyetli FPGAlere** hedeflenmiş bir RISC-V sistemi.
- RVfpga kursunu bitirdikten sonra kullanıcılar anladıkları, nasıl kullanıp değiştireceklerini bildikleri, **çalışan bir RISC-V işlemcisi, SoC'si, ekosistemi** ile ayrılacaklar.

RVfpga Kurs içerikleri



RVfpga İçerikleri

- **İlk Kullanım Kılavuzu**

- Hızlı Başlangıç Kılavuzu
- RISC-V Mimarisi ile RVfpga'ın tanıtımı
- Araçları Kurma (VSCode, PlatformIO, Vivado, Verilator, Whisper)
- RVfpga'i Donanımda, Simülasyonda Çalıştırma

RVfpga kursun **da** FPGA'e hedeflenen RISC-V SoC'sinin **de** adıdır.

- **Deneyler**

- **1-10:** RVfpga'i Vivado'da Kurgulama, RVfpga'i Programlama, RVfpga'i çevre birimleri ekleyerek Genişletme (Kasım 2020'de yayınlandı)
- **11-20:** RVfpga'in RISC-V çekirdeği ile bellek sistemini çözümleyip değiştirme (2021 dördüncü çeyrekte yayınlanacak)

RVfpga Kursu

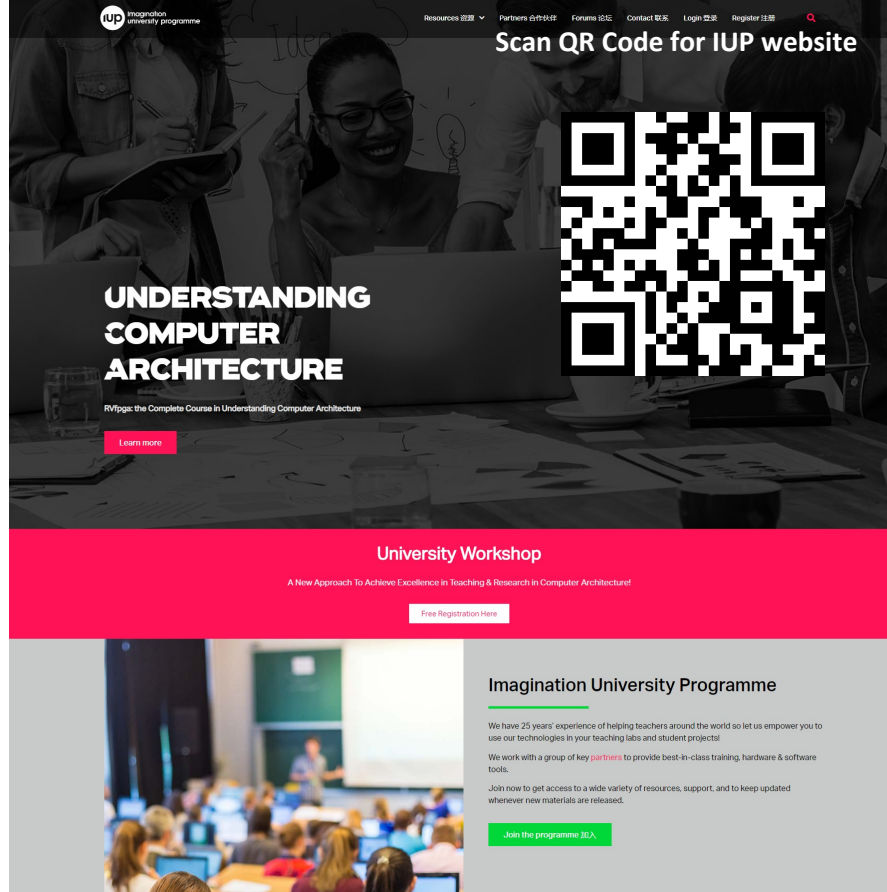
- **1-2 Dönemlik Kurs**

- Lisans (1-10 arası Deneyler)
- Yüksek lisans/üst dönemler (11-20 arası Deneyler)

- **Beklenen Ön Bilgi**

- **Dijital tasarımın, yüksek-düzeyle programlamanın (C önerilir), yönerge kümesi mimarisinin, çevirici programlamanın, işlemci mikromimarisinin, bellek sistemlerinin** (bu içerik *Digital Design and Computer Architecture: RISC-V Edition*, Harris & Harris, © Elsevier kapsamındadır, beklenen yayın dönemi: 2021 yazı) temelleri
- Bu konular RVfpga kursu boyunca uygulamalı öğrenimle genişletilip sağlanılacaktır

RVfpga'e Nasıl Ulaşılır



Imagination University Programme Website

- **Imagination University Programme'a (IUP) Kayıt Ol** – dünya çapındaki öğretmenler, araştırmacılar, öğrenciler için:
<https://university.imgtec.com>
 - Yayınların **güncellemelerini**, **bildirimlerini** al
 - İçerikleri **isteyip indir**
 - **Destek Forumları**: müfredat/öğretim tartışmaları için PowerVR, RVfpga, & AI Forumları; IUP Forum
- **Sosyal Medya**:
 - **Robert Owen, IUP Direktörü**: @UniPgm
 - **Imagination Technologies**: @ImaginationTech
 - **WeChat & Weibo**: ImaginationTech

RVfpga Gerekli Yazılım ile Donanım

YAZILIM

Xilinx **Vivado** 2019.2 WebPACK

PlatformIO – Microsoft’un **Visual Studio Code**’unun bir eklentisi – Chips Alliance platformu ile, şunları içerir: RISC-V Toolchain, OpenOCD, Verilator HDL Simulator, WD Whisper yönerge kümesi simülatörü (ISS)

DONANIM*

Digilent’in **Nexys A7** / Nexys 4 DDR FPGA Kartı

*Deneylerin hepsi yalnızca simülasyonda bitirilebilir; yani bu donanım önerilir ancak gerekmez.

RISC-V ÇEKİRDEK & SOC

Çekirdek: Western Digital’in **SweRV EH1**’i

SoC: Chips Alliance’in **SweRVolf**’u

Bütünüyle ücretsiz, yalnızca FPGA kartı \$265 (akademik ücret: \$199) tutar

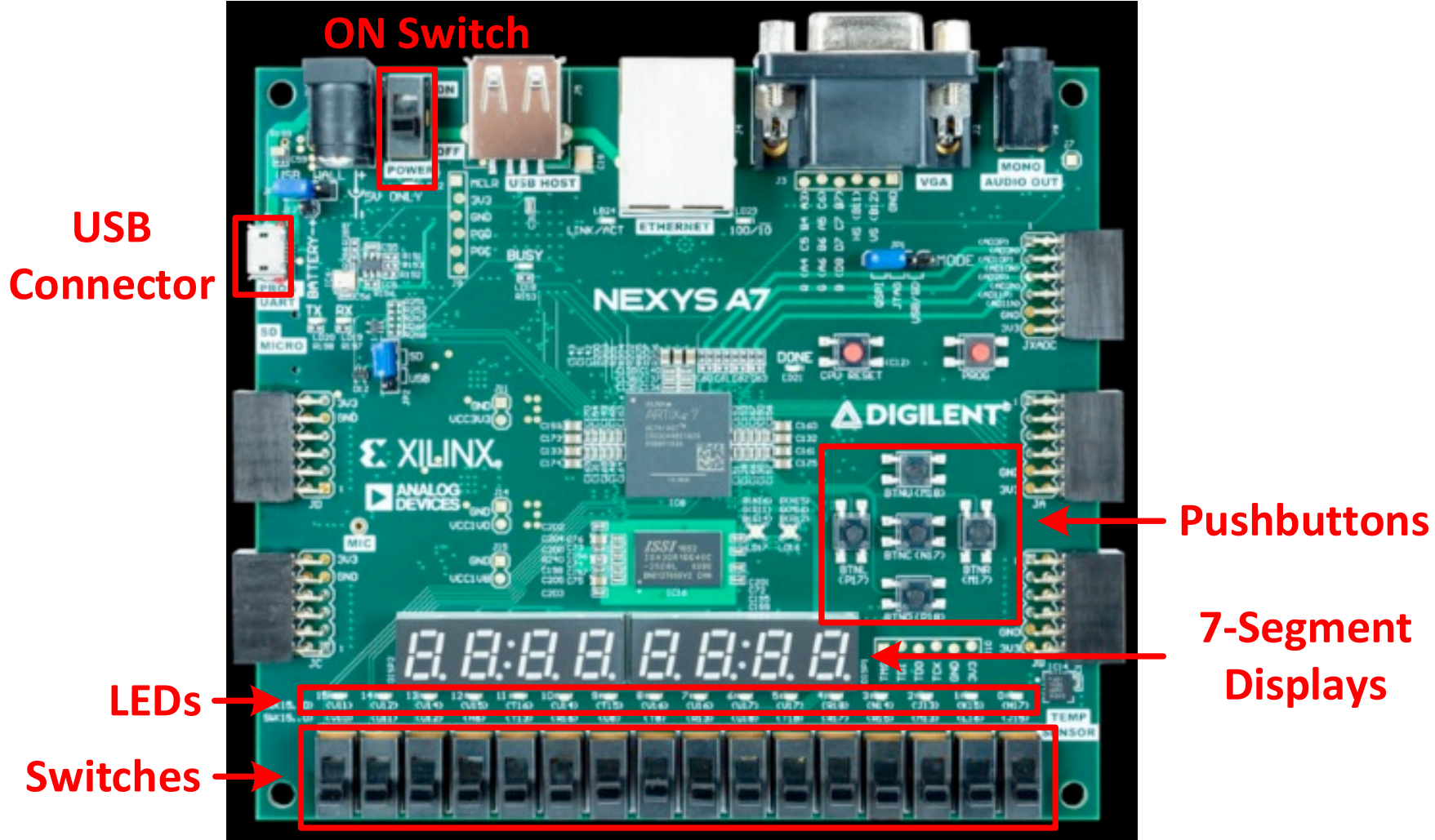
Desteklenen Platformlar

- **İşletim Sistemleri**
 - **Ubuntu 18.04** (sonraki sürümler de çalışacaktır)
 - **Windows 10**
 - **macOS**

RVfpga Yazılım Araçları

- **Xilinx'in Vivado IDE'si**
 - RVfpga kaynak dosyaları (Verilog / SystemVerilog) ile RVfpga'in hiyerarşisini gör
 - Nexys A7 kartına hedeflenmiş RVfpga için veri dosyası (FPGA yapılandırma dosyası) oluştur
- **PlatformIO** – bir Visual Studio Code (VSCode) eklentisi
 - RVfpga'i Nexys A7 kartına indir
 - RVfpga'de C ile çevirici programlarını derle, indir, çalıştır, ayıkla
- **Verilator** – bir HDL (donanım tanım dili) simülatörü
 - RVfpga'in iç sinyallerini çözümlemek için RVfpga'i HDL (alçak) düzeyde simüle et

Nexys A7-100T FPGA Kartı



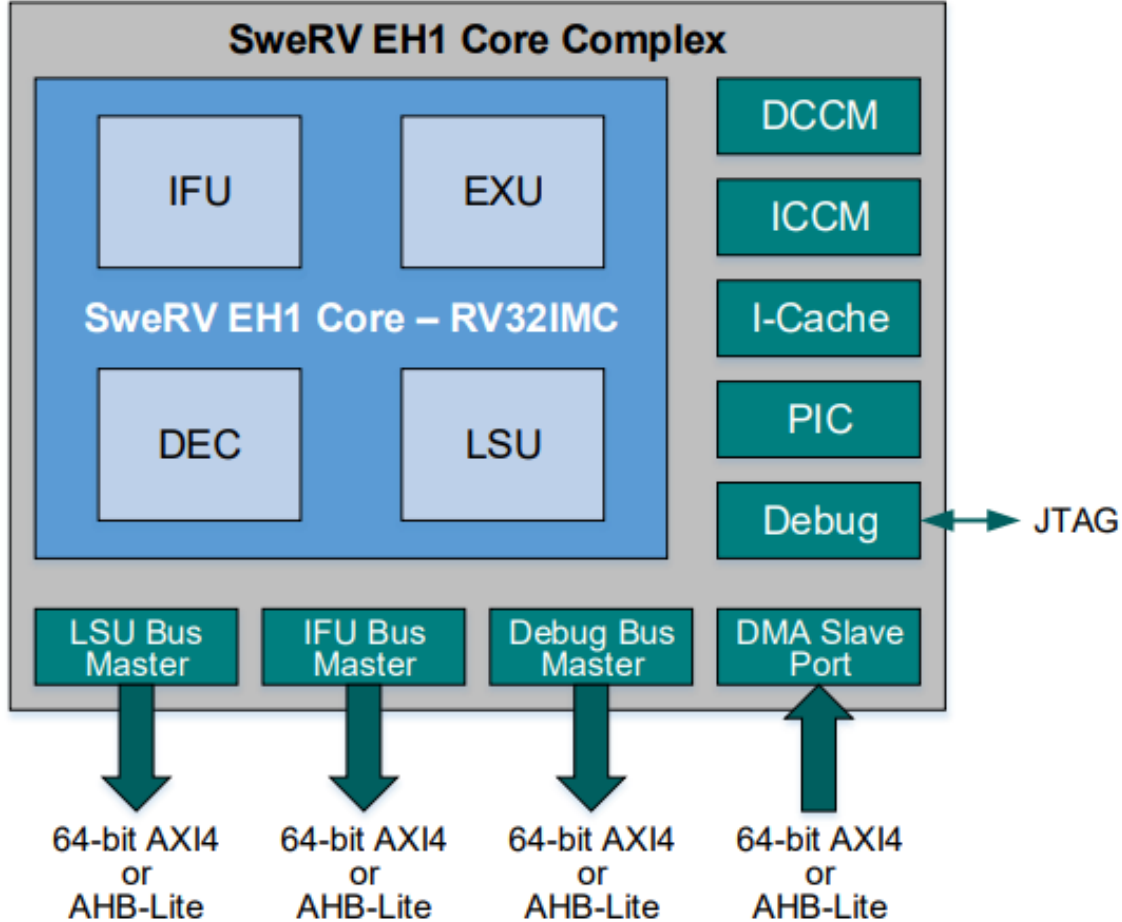
- **Artix-7** alanda programlanabilir geçit dizisi (FPGA) içerir
- **Çevre birimleri** (örneğin LEDler, anahtarlar, düğmeler, 7-kesimli ekranlar, ivmeölçer, sıcaklık sensörü, mikrofon gibi) içerir
- **digilentinc.com** ile diğer sağlayıcılardan alınabilir

Kartın şuradan figürü: <https://reference.digilentinc.com/>

RISC-V Çekirdekleri ile SoCleri



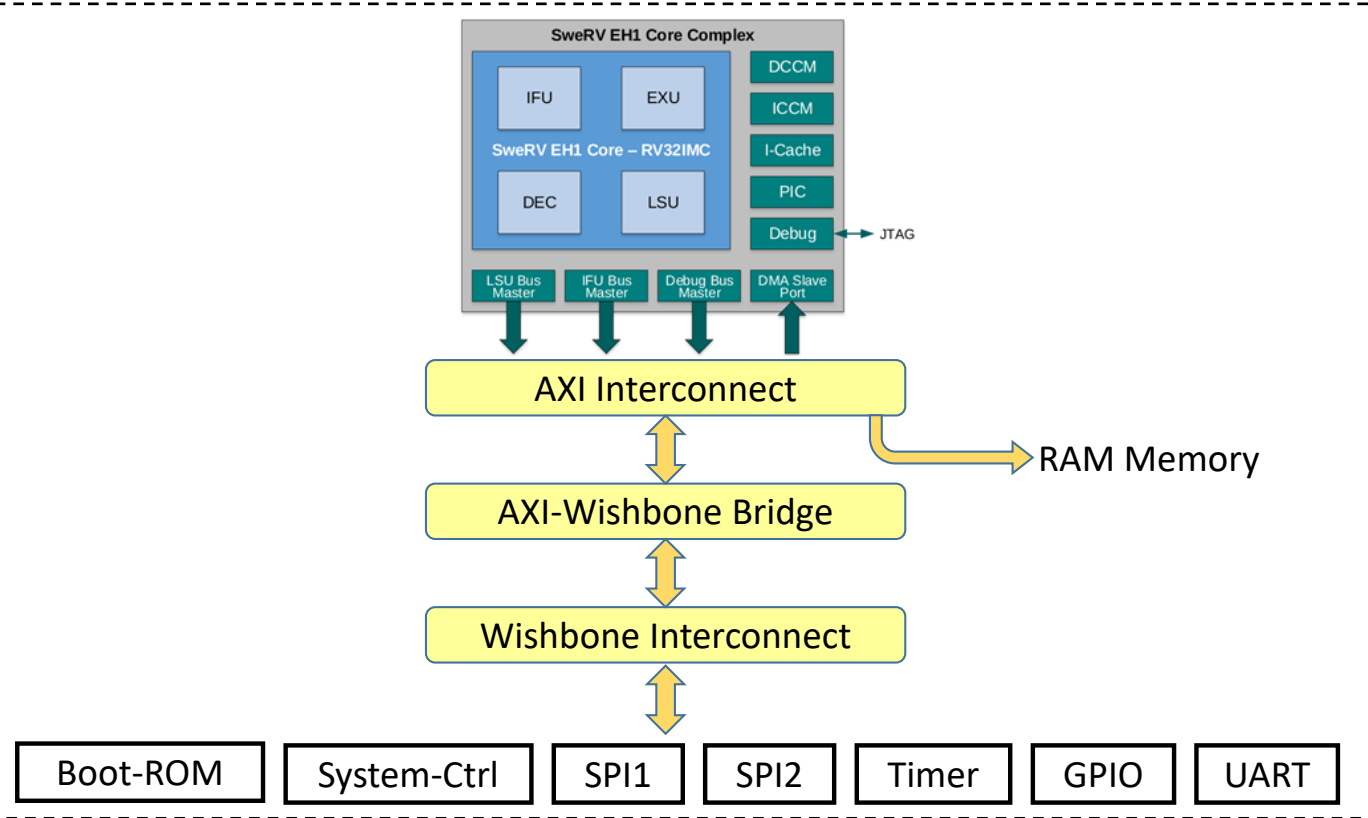
SweRV EH1 Çekirdeği



- Western Digital'den açık-kaynak çekirdek
- 32-bit (RV32IMC) süperskaler çekirdek, çift-yayın 9-aşama boruhatlı
- Çekirdeğe sıkı bağlı ayrı yönerge ile veri bellekleri (ICCM ile DCCM)
- Eşlikli ya da ECC korumalı 4-yön küme-ilişkisel I\$ (I-Cache)
- Programlanabilir Kesinti Denetleyicisi
- RISC-V Ayıklama spesifikasyonu ile uyumlu Çekirdek Ayıklama Ünitesi
- Sistem Veri Yolu: AXI4 ya da AHB-Lite

Figür şuradan: https://github.com/chipsalliance/Cores-SweRV/blob/master/docs/RISC-V_SweRV_EH1_PRM.pdf

Geniřletilmiř SweRVolf SoC

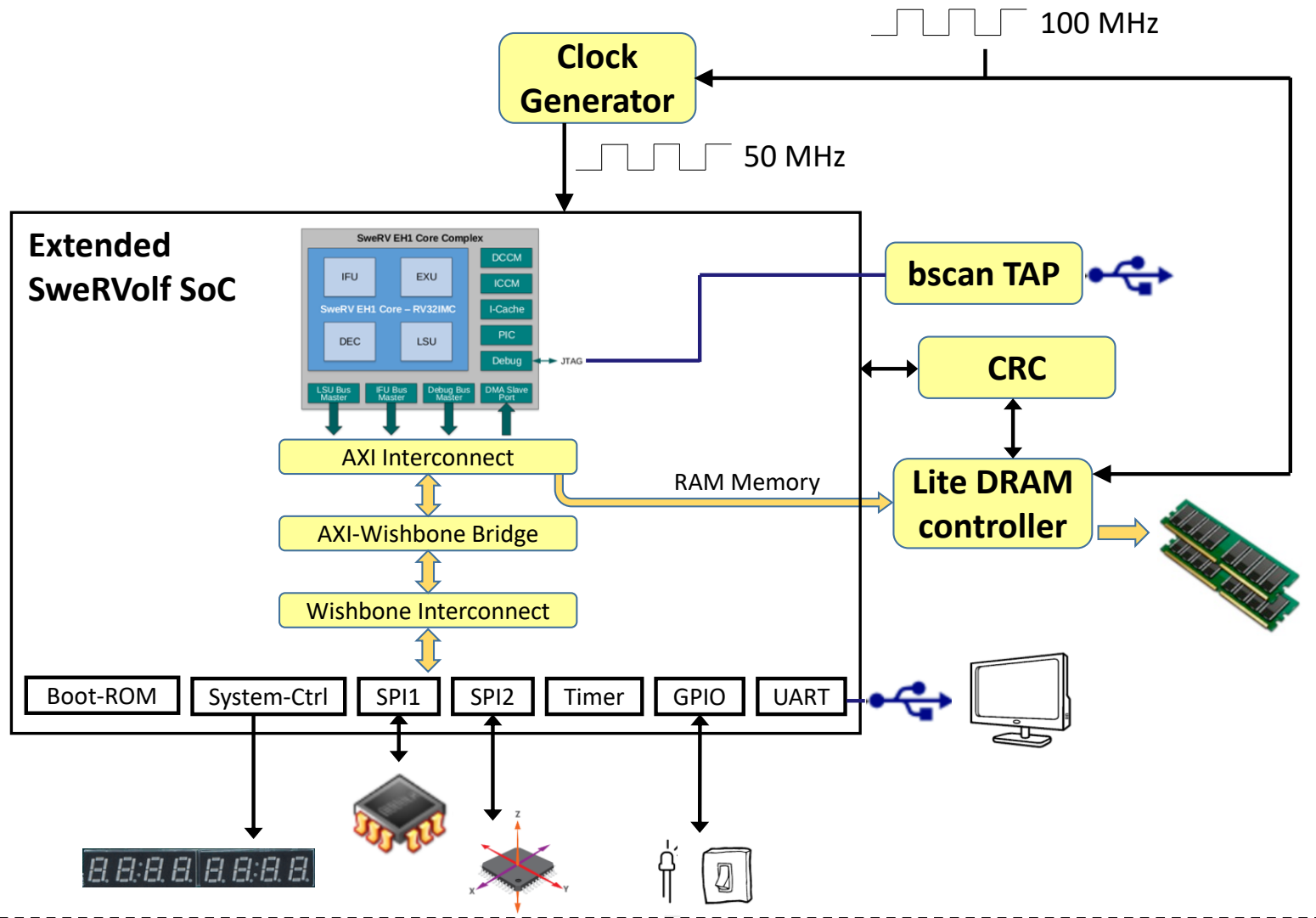


Geniřletilmiř
SweRVolf Bellek
Eřlenimi

Sistem	Adres
Boot ROM	0x80000000 - 0x80000FFF
Sistem Denetleyicisi	0x80001000 - 0x8000103F
SPI1	0x80001040 - 0x8000107F
SPI2	0x80001100 - 0x8000113F
Zamanlayıcı	0x80001200 - 0x8000123F
GPIO	0x80001400 - 0x8000143F
UART	0x80002000 - 0x80002FFF

- Chips Alliance'dan aık-kaynak yongadaki-sistem (SoC)
- SweRVolf, SweRV EH1 ekirdeęini kullanır. SweRVolf bir Boot (onykleme) ROM, UART, bir Sistem Denetleyicisi, bir SPI denetleyicisi (SPI1) ierir.
- RVfpga, SweRVolf'u bir bařka SPI denetleyicisi (SPI2), bir GPIO (Genel Amalı Girdi/ıktı), 8-sayı 7-Kesimli Ekran, bir zamanlayıcı ile geniřletir.
- SweRV ekirdeęi bir AXI veri yolu kullanır, evre birimleri bir Wishbone veri yolu kullanır, dolayısıyla SoC'de AXI'dan Wishbone'a Kpr de vardır

RVfpga



- **RVfpga:** Nexys A7 FPGA kartına hedeflenmiş, ek çevre birimli Genişletilmiş SweRVolf SoC:

- **Çekirdek & Sistem:**

- Genişletilmiş SweRVolf SoC
- Lite DRAM denetleyici
- Saat Oluşturucu, Saat Alanı, JTAG portu için BSCON mantığı

- Nexys A7 FPGA kartında kullanılan **Çevre Birimleri:**

- DDR2 bellek
- USB bağlantısıyla UART
- SPI Flash bellek
- 16 LED, 16 anahtar
- SPI İvmeölçer
- 8-sayı 7-kesimli ekran

RVfpga Eklentileri

- SweRVolf SoC'si 6-10 arası Deneylerde **daha da genişletilir:**
 - Karttaki Nexys A7 **düğmeleriyle** arayüz yapmak için **GPIO** denetleyicisi
 - **7-kesimli ekran** denetleyicisinin değiştirilmesi
 - Karttaki **üç renkli LEDleri** kullanmak için yeni **zamanlayıcı** modülleri
 - Yeni **dış kesinti kaynakları**

RVfpga Deneylerinin Tanıtımı



RVfpga Deneyleri

1-10 arası Deneyler (Kasım 2020'de yayınlandı)

- Vivado Projesi, Programlama
- I/O Sistemleri

11-20 arası Deneyler (2021 dördüncü çeyrekte yayınlanacak)

- RISC-V Çekirdeği
- RISC-V Bellek Sistemleri

Uygulamalı tasarımla daha iyi öğrenilmesi için bütün deneyler RVfpga'ı kullanma ya da değiştirme ya da kullanıp değiştirme **alıştırmaları** içerir.

RVfpga Deneyler 1-10

RVfpga kaynak koduna nasıl bakılıp (Verilog/SV) FPGA'e hedefleneceğini (1), RVfpga için C ile çevirici programlarının nasıl yazılacağını (2-5), çevre birimleri eklemek için RVfpga'ın nasıl değiştirileceğini (6-10) gösterir.

Programlama

- Deney 0: RVfpga Deneylerinin Tanıtımı
- Deney 1: Bir Vivado Projesi Oluşturma
- Deney 2: C Programlaması
- Deney 3: RISC-V Çevirici Dili
- Deney 4: İşlev Çağrılar
- Deney 5: Görüntü İşleme: C & Çevirici
- Deney 6: Girdi/Çıkıya Giriş
- Deney 7: 7-Kesimli Ekranlar
- Deney 8: Zamanlayıcılar
- Deney 9: Kesinti-güdümlü Girdi/Çıktı
- Deney 10: Dizisel Veri Yolları

Girdi/Çıktı Sistemleri

RVfpga Deneyler 1-5: Vivado Proje & Programlama

- **Deney 1: Bir Vivado Projesi Oluşturma:** RISC-V sistemini (RVfpga) bir FPGA kartına hedefleyip RVfpga'i Verilator'da simüle etmek için bir Vivado projesi kurgula
- **Deney 2: C Programlaması:** PlatformIO'da bir C programı yazıp RVfpga'de çalıştır / ayıkla. Ayrıca terminale yazdırma gibi işlemleri destekleme için Western Digital'in Kart Destek ile Platform Destek Paketlerine (BSP ile PSP) giriş yap.
- **Deney 3: RISC-V Çevirici Dili:** PlatformIO'da bir RISC-V çevirici programı yazıp RVfpga'de çalıştır / ayıkla
- **Deney 4: İşlev Çağrılar:** İşlev çağrılarına, C kütüphanelerine, RISC-V çağırma uzlaşımına giriş
- **Deney 5: Görüntü İşleme: C & Çevirici:** Çevirici kodunu C koduyla birlikte göm



RVfpga Deneyler 6-10: Girdi/Çıktı & RVfpga Çevre Birimleri

- **Deney 6: Girdi/Çıktıya Giriş:** Bellek eşlenmiş Girdi/Çıktı ile RVfpga'nın açık kaynak GPIO modülüne giriş
- **Deney 7: 7-Kesimli Ekranlar:** 7-kesimli ekran çözücüsü kurgulayıp RVfpga sistemine entegre et
- **Deney 8: Zamanlayıcılar:** Zamanlayıcılarla bir Zamanlayıcı denetleyicisini anlayıp kullan
- **Deney 9: Kesinti-güdümlü Girdi/Çıktı:** RVfpga kesinti desteğiyle kesinti-güdümlü Girdi/Çıktının kullanımına giriş
- **Deney 10: Dizisel Veri Yolları:** Dizisel arayüzlere giriş (SPI, I2C, UART). SPI arayüzünü kullanan karttaki ivmeölçeri kullan

RVfpga Deneyler 11-15: RISC-V Çekirdeği

- **2021 dördüncü çeyrekte yayınlanacak**
- Çekirdek yapısını anlama
- Yönerge akışını boruhattı üzerinden anlama (Aritmetik/Mantık, Bellek, Sıçramalar, Dallandırmalar)
- Sakıncaları anlayıp başa çıkma
- Yeni yönergeleri gerçekleştirip FPGA kartında yürütme
- Dallandırma kestiricisini anlayıp değiştirme
- Süperskaler işlemeyi anlama

RVfpga Deneyler 16-20: RISC-V Bellek Sistemleri

- **2021 dördüncü çeyrekte yayınlanacak**
- Önbellek yakalamasıyla/ kaçıışıyla bellek hiyerarşisinin işlemini anlama
- Önbelleği değiştirme: değişik önbellek boyutlarını, yapılandırmalarını, yönetim politikalarını gerçekleştirme
- Önbellek denetleyicisini anlama
- Bellekleri anlama: ICCM (yönerge sıkı bağlı bellek) ile DCCM (veri sıkı bağlı bellek)

RVfpga Süreci

RVfpga'e Erişilebilirlik

Kasım 2020	RVfpga İlk Kullanım Kılavuzu RVfpga Deneyler 1-10
2021 4. Çeyrek	RVfpga Deneyler 11-20
Mart 2021	Yüksek lisans düzeyinde SoC Tasarım Kursu
Diller	İngilizce & Çince & Türkçe (sonra İspanyolca & Japonca)
Ders Kitabı	<i>Digital Design & Computer Architecture: RISC-V Edition 2021</i> by Sarah Harris and David Harris

- **Hedef Kitle**

- Elektrik mühendisliği, bilgisayar bilimi, bilgisayar mühendisliği lisans öğrencileri
- RISC-V mimarisini öğrenmeye ilgili Akademi & Endüstri profesyonelleri

- **Imagination University Programme'ın (IUP) Yaptıkları: MIPSfpga Programını Geliştirdi:**

- Nisan 2015'te çıktı
- 800 üniversiteyle etkileşime geçti
- Elektra Best Educational Support Award, Europe 2015 ödülünü kazandı

RVfpga

Hızlı Başlangıç Kılavuzu



Hızlı Başlangıç Kılavuzu Tanıtımı


- VSCode & PlatformIO kur
- RVfpga'de **Örnek Programı** çalıştır

PlatformIO & VSCode Kur

- **VSCode** kur
 - <https://code.visualstudio.com/Download>
 - Ubuntu ile macOS için **Python** kur (bu adım Windows için gerekmez)
- VSCode içerisinde **PlatformIO** eklentisini kur
- Nexys A7 Kart **sürücülerini** kur (RVfpga İlk Kullanım Kılavuzu yönergelerine göz at)

RVfpga'i Karta İndirip Programı Çalıştır

- **PlatformIO'da:**

- Anahtarların değerini LEDlere yazan **örnek programı** aç. Program şurada:
[RVfpgaPath]\RVfpga\examples\LedsSwitches_C-Lang
- PlatformIO ilk değerlendirme dosyasında (platformio.ini) **RVfpga veri dosyasının** dizin yerini güncelle – bir diğer deyişle, şu satırı platformio.ini'ye ekle: *board_build.bitstream_file = [RVfpgaPath]/RVfpga/src/rvfpga.bit*
- **RVfpga SoC'yi** Nexys A7 Kartına **İndir** (Project Tasks → env:swervolf_nexys → Platform → Upload Bitstream)
- Çalıştır/Ayıkla butonuna basarak **programı RVfpga'de derle, indir, çalıştır**, buton şu: 

[RVfpgaPath] RVfpga klasörünün makinadaki konumudur. Bu klasör Imagination University Programme'den gelen RVfpga paketinde sağlanmıştı.

RVfpga Deney Tanımları



Deney 1: Vivado Projesi



RVfpga Deney 1: RVfpga Vivado Projesi

- **Vivado**, RVfpga'in kaynak (Verilog) kodunu görüntüleme, değiştirme, sentezleme için bir Xilinx aracıdır.
- RVfpga'in kaynak kodu şuradadır:
[RVfpgaPath]/RVfpga/src
- RVfpga'in kaynak kodunu içeren bir **Vivado projesi** oluşturun. Nexys A7 kartına hedeflenmiş RVfpga sentezleyip FPGA'yi RVfpga olarak yapılandırma bilgisi içeren bir **veri dosyası** (veri akışı dosyası da denir) oluşturun.
- Ayrıca RVfpga'in kaynak kodunun simülasyonunu yapıp, iç sinyalleri incelemek için **Verilator**'ü - bir HDL simülatörü - de kullanabilirsiniz (Verilator'ün kullanım yönergeleri için RVfpga İlk Kullanım Kılavuzuna göz at).
- Vivado ile Verilator **Deneyler 6-10** arasında RVfpga SoC'sini değiştirip simülasyonunu yapmak için çok kullanılacaktır.

Deney 2:

C Programlaması



RVfpga Deney 2: C Programlaması

- PlatformIO projesi oluştur
- Örnek C programını projeye ekle
- RVfpga'i Nexys A7 kartına indir
- C programını RVfpga'e indirip programı çalıştır/ayıkla
- Deneyin sonundaki **alıştırmaların** hepsini ya da birkaçını bitir

RVfpga Deney 2: Örnek C Programı

```
// bellek-eşlenmiş girdi/çıktı adresleri
```

```
#define GPIO_SWs      0x80001400
```

```
#define GPIO_LEDs     0x80001404
```

```
#define GPIO_INOUT    0x80001408
```

Bu program anahtarların
değerlerini LEDlere yazar.

```
#define READ_GPIO(dir) (*(volatile unsigned *)dir)
```

```
#define WRITE_GPIO(dir, value) { (*(volatile unsigned *)dir) = (value); }
```

```
int main ( void )
```

```
{
```

```
    int En_Value=0xFFFF, switches_value;
```

```
    WRITE_GPIO(GPIO_INOUT, En_Value);
```

```
    while (1) {
```

```
        switches_value = READ_GPIO(GPIO_SWs);    // anahtarlardaki değerleri oku
```

```
        switches_value = switches_value >> 16;   // alt 16 bitlere kaydır
```

```
        WRITE_GPIO(GPIO_LEDs, switches_value);   // anahtar değerini LEDlerde göster
```

```
    }
```

```
    return(0);
```

```
}
```


RVfpga Deney 2: Bellek-Eşlenmiş I/O Adresleri

Aygıt	Bellek-Eşlenmiş I/O Adresi
Anahtarlar (Nexys A7 kartında 16)	0x80001400 (üst 16 bitler)
LEDler (Nexys A7 kartında 16)	0x80001404 (alt 16 bitler)
GPIO Girdi/Çıktısı (1 = çıktı, 0 = girdi)	0x80001408

RVfpga Deney 2: Western Digital'in BSP & PSP'si

- Western Digital şunları sağlar:
 - **PSP**: işlemci destek paketi
 - **BSP**: kart destek paketi
- Bunlar belirli işlemci (SweRV EH1 core) ile kart (Nexys A7 FPGA kartı) için yaygın işlevleri sağlar.
 - **Örnek:** `printfNexys` (C'deki `printf` işlevi gibi)

RVfpga Deney 2: Terminale Yazdırma için UART

```
#if defined(D_NEXYS_A7)
#include <bsp_printf.h>
#include <bsp_mem_map.h>
#include <bsp_version.h>
#else
PRE_COMPILED_MSG("no platform was defined")
#endif
#include <psp_api.h>
#define DELAY 10000000

int main(void) {
    int i, j = 0;

    // UART'a ilk değerini ver
    uartInit();
    while (1) {
        printfNexys("Hello RVfpga users! Iteration: %d\n", j);
        for (i=0; i < DELAY; i++) ; // printf'ler arasında geciktir
        j++;
    }
}
```

- Şu satırı **platform.ini** dosyasına ekle:

monitor_speed = 115200

- Program çalışmaya başladıktan sonra, pencerenin aşağısındaki şu butona basarak **PlatformIO terminalini aç**:



Deney 3:

RISC-V Çeviricisi



RVfpga Deney 3: RISC-V Çeviricisi

- RISC-V Çevirici Dili Tanıtımı
- PlatformIO projesi oluştur
- Örnek RISC-V çevirici programını projeye ekle
- RVfpga'i Nexys A7 kartına indir
- RISC-V çevirici programını RVfpga'e indirip programı çalıştır/ayıkla
- Deneyin sonundaki **alıştırmaların** hepsini ya da birkaçını bitir

RVfpga Deney 3: RISC-V Çevirici Yönergeleri

Yaygın RISC-V Çevirici Yönergeleri/Sözde Yönergeleri

RISC-V Çeviricisi	Tanım	İşlem
add s0, s1, s2	Ekle	$s0 = s1 + s2$
sub s0, s1, s2	Çıkar	$s0 = s1 - s2$
addi t3, t1, -10	Anlıkla ekle	$t3 = t1 - 10$
mul t0, t2, t3	32-bit çarp	$t0 = t2 * t3$
div s9, t5, t6	Bölüm	$t9 = t5 / t6$
rem s4, s1, s2	Kalan	$s4 = s1 \% s2$
and t0, t1, t2	Bitsel AND yap	$t0 = t1 \& t2$
or t0, t1, t5	Bitsel OR yap	$t0 = t1 t5$
xor s3, s4, s5	Bitsel XOR yap	$s3 = s4 \wedge s5$
andi t1, t2, 0xFFB	Anlıkla bitsel AND yap	$t1 = t2 \& 0xFFFFFBB$
ori t0, t1, 0x2C	Anlıkla bitsel OR yap	$t0 = t1 0x2C$
xori s3, s4, 0xABC	Anlıkla bitsel XOR yap	$s3 = s4 \wedge 0xFFFFFABC$
sll t0, t1, t2	Sola mantıksal kaydır	$t0 = t1 \ll t2$
srl t0, t1, t5	Sağa mantıksal kaydır	$t0 = t1 \gg t5$
sra s3, s4, s5	Sağa aritmetik kaydır	$s3 = s4 \ggg s5$
slli t1, t2, 30	Anlıkla sola mantıksal kaydır	$t1 = t2 \ll 30$
srlt t0, t1, 5	Anlıkla sağa mantıksal kaydır	$t0 = t1 \gg 5$
srait s3, s4, 31	Anlıkla sağa aritmetik kaydır	$s3 = s4 \ggg 31$

RVfpga Deney 3: RISC-V Çevirici Yönergeleri

Yaygın RISC-V Çevirici Yönergeleri/Sözde Yönergeleri (devam)

RISC-V Çeviricisi	Tanım	İşlem
lw s7, 0x2C(t1)	Sözcüğü yükle	$s7 = \text{memory}[t1+0x2C]$
lh s5, 0x5A(s3)	Yarım-sözcüğü yükle	$s5 = \text{SignExt}(\text{memory}[s3+0x5A]_{15:0})$
lb s1, -3(t4)	Baytı yükle	$s1 = \text{SignExt}(\text{memory}[t4-3]_{7:0})$
sw t2, 0x7C(t1)	Sözcüğü depola	$\text{memory}[t1+0x7C] = t2$
sh t3, 22(s3)	Yarım-sözcüğü depola	$\text{memory}[s3+22]_{15:0} = t3_{15:0}$
sb t4, 5(s4)	Baytı depola	$\text{memory}[s4+5]_{7:0} = t4_{7:0}$
beq s1, s2, L1	Eşitse dallandır	if ($s1==s2$), PC = L1
bne t3, t4, Loop	Eşit değilse dallandır	if ($s1!=s2$), PC = Loop
blt t4, t5, L3	Azsa dallandır	if ($t4 < t5$), PC = L3
bge s8, s9, Done	Çok ya da eşitse dallandır	if ($s8 \geq s9$), PC = Done
li s1, 0xABCDEF12	Anlığı yükle	$s1 = 0xABCDEF12$
la s1, A	Adresi yükle	$s1 = \text{Memory address where variable A is stored}$
nop	İşlem yok	no operation
mv s3, s7	Taşı	$s3 = s7$
not t1, t2	Değil (Tersine Döndür)	$t1 = \sim t2$
neg s1, s3	Olumsuzla	$s1 = -s3$
j Label	Sıçra	PC = Label
jal L7	Sıçrayıp bağla	PC = L7; ra = PC + 4
jr s1	Yazmaça sıçra	PC = s1

RVfpga Deney 3: RISC-V Yazmaçları

32 tane 32-bit yazmaç

Ad	Yazmaç Numarası	Kullanım
zero	x0	Sabit değer 0
ra	x1	Dönüş adresi
sp	x2	Yığın göstergesi
gp	x3	Genel gösterge
tp	x4	İş parçacığı göstergesi
t0-2	x5-7	Geçici değişkenler
s0/fp	x8	Kaydedilmiş değişken / Çerçeve göstergesi
s1	x9	Kaydedilmiş değişken
a0-1	x10-11	İşlev argümanları / Dönüş değerleri
a2-7	x12-17	İşlev argümanları
s2-11	x18-27	Kaydedilmiş değişkenler
t3-6	x28-31	Geçici değişkenler

RVfpga Deney 3: Örnek RISC-V Çevirici Programı

```
• // bellek-eşlenmiş girdi/çıkıtı adresleri
• # GPIO_SWs      = 0x80001400
• # GPIO_LEDs     = 0x80001404
• # GPIO_INOUT    = 0x80001408
•
• .globl main
• main:
•
• main:
•     li t0, 0x80001400    # genel-amaçlı girdi/çıkıtı bellek-eşlenmiş yazmaçların taban adresi
•     li t1, 0xFFFF       # genel-amaçlı girdi/çıkıtların yönünü ayarla
•                          # üst yarım = anahtarlar (girdiler)      (=0)
•                          # alt yarım = çıkıtlar (LEDler)          (=1)
•     sw t1, 8(t0)         # GPIO_INOUT = 0xFFFF
•
• repeat:
•     lw  t1, 0(t0)        # anahtarları oku: t1 = GPIO_SWs
•     srli t1, t1, 16       # değeri 16 bit sağa kaydır
•     sw  t1, 4(t0)        # değeri LEDlere yaz: GPIO_LEDs = t1
•     j   repeat           # döngüyü tekrarla
```

Bu program anahtarların değerini LEDlere yazar.



Deney 4:

İşlev Çağrıları



RVfpga Deney 4: İşlev Çağrılar

- **İşlev çağrılar** olan C programları yaz
 - İşlevlere *prosedür (işlem)* de denir
- **C kütüphaneleri** kullanarak
- RISC-V (Prosedür) **Çağırma Uzlaşımı**

RVfpga Deney 4: İşlevli Örnek Program

```
// bellek-eşlenmiş I/O adresleri
#define GPIO_SWs      0x80001400
#define GPIO_LEDs     0x80001404
#define GPIO_INOUT    0x80001408
#define READ_GPIO(dir) (*(volatile unsigned *)dir)
#define WRITE_GPIO(dir, value) { (*(volatile unsigned *)dir) = (value); }

void IOsetup();
unsigned int getSwitchVal();
void writeValtoLEDs(unsigned int val);

int main ( void ) {
    unsigned int switches_val;

    IOsetup();
    while (1) {
        switches_val = getSwitchVal();
        writeValtoLEDs(switches_val);
    }

    return(0);
}
```

RVfpga Deney 4: İşlevli Örnek Program

```
void IOsetup()  
{  
    int En_Value=0xFFFF;  
    WRITE_GPIO(GPIO_INOUT, En_Value);  
}  
  
unsigned int getSwitchVal()  
{  
    unsigned int val;  
  
    val = READ_GPIO(GPIO_SWs);    // anahtarlardaki değeri oku  
    val = val >> 16;    // alt 16 bitlere kaydır  
    return val;  
}  
  
void writeValtoLEDs(unsigned int val)  
{  
    WRITE_GPIO(GPIO_LEDs, val);    // değeri LEDlerde göster  
}
```

RVfpga Deney 4: C Kütüphaneleri

- **Kütüphaneler**
 - Yaygın kullanılan işlevler koleksiyonu
 - Yaygın kullanılan işlevlere kolayca erişilebilmesi için sağlanır (programlama süresinden kurtarır)
- **Örnek C kütüphaneleri:**
 - **math.h** (matematik kütüphanesi): sqrt (karekök), cos (kosinüs), gibi işlevler içerir
 - **stdio.h** (standart Girdi/Çıktı kütüphanesi): ekrana değerler yazdırmak (printf), kullanıcıdan değerler okumak (scanf), gibisi için işlevler içerir
 - **stdlib.h** (standart kütüphane): rastgele sayılar oluşturmak için işlevler içerir (rand).
 - **Daha nicesi...** (google'da C kütüphanelerini arat)

RVfpga Deney 4: C Kütüphaneli Örnek Program

```
#include <stdlib.h>
```

```
...
```

```
int main(void) {  
    unsigned int val;  
    volatile unsigned int i;  
  
    IOsetup();  
    while (1) {  
        val = rand() % 65536;  
        writeValtoLEDs(val);  
        for (i = 0; i < DELAY; i++)  
            ;  
    }  
    return(0);  
}
```

Bu program 0 ile 65535 arasından seçilen rastgele bir sayıyı LEDlere yazar.

RVfpga Deney 4: RISC-V Çağırma Uzlaşımı

- **Bir işlevi çağır**
`jal function_label`
- **Bir işlevden dönüş yap**
`jr ra`
- **Argümanlar**
 - şu yazmaçlara koyulur: `a0–a7`
- **Dönüş değeri**
 - şu yazmaça koyulur: `a0`

RVfpga Deney 4: RISC-V Çağrı Uzlaşımı Örneği

C Kodu

```
int main() {  
    ...  
    int y = y + func1(1, 2, 3)  
    y++;  
    ...  
}  
  
int func1(int a, int b, int c) {  
    int sum;  
    sum = a + b + c;  
    return sum;  
}
```

RISC-V Çeviricisi

```
# y is in s0  
main:  
    ...  
    addi a0, zero, 1 # put values in argument registers  
    addi a1, zero, 2  
    addi a2, zero, 3  
    jal  func1       # call function func1  
    add  s0, s0, a0   # y = y + return value  
    addi s0, s0, 1    # y = y++  
    ...  
  
# sum is in s0  
func1:  
    add s0, a0, a1    # sum = a + b  
    add s0, s0, a2    # sum = a + b + c  
    addi a0, s0, 0    # return value = sum  
    jr   ra          # return
```

RVfpga Deney 4: Yığın

- Bellekte yazmaç değerlerini kaydetmek için kullanılan **üzerine yazılabilen boşluk**
- Yığın göstergesi (sp) yığının üstünün adresini tutar
- **Yığın** bellekte **aşağı doğru büyür**. Yani, örneğin, yığında 4 sözcük (16 bayt) boşluk açmak için şu kod kullanılır:

```
addi sp, sp, -16
```

- **Yazmaçların iki türü:**
 - **Korunur yazmaçlar:** yazmaç içeriği işlev çağrıları arasında **korunmalıdır** (diğer deyişle işlev çağrısının öncesinde değer neyse sonrasında da o olmalıdır)
 - **Korunmaz yazmaçlar:** yazmaç içeriği işlev çağrıları arasında **korunmamalıdır** (diğer deyişle işlev çağrısının öncesinde değer neyse sonrasında da o olması gerekmez)
 - Kaydedilmiş yazmaçlar ($s0-s11$), dönüş adresi yazmacı (ra), yığın göstergesi (sp) **korunur** yazmaçlardır. Diğer yazmaçlar korunmaz.

RVfpga Deney 4: Korunur / Korunmaz Yazmaçlar

Ad	Yazmaç Numarası	Kullanım	Korunur
zero	x0	Sabit değer 0	-
ra	x1	Dönüş adresi	Korunur
sp	x2	Yığın göstergesi	Korunur
gp	x3	Genel gösterge	-
tp	x4	İş parçacığı göstergesi	-
t0-2	x5-7	Geçici değişkenler	Korunmaz
s0/fp	x8	Kaydedilmiş değişken / Çerçeve göstergesi	Korunur
s1	x9	Kaydedilmiş değişken	Korunur
a0-1	x10-11	İşlev argümanları / Dönüş değerleri	Korunmaz
a2-7	x12-17	İşlev argümanları	Korunmaz
s2-11	x18-27	Kaydedilmiş değişkenler	Korunur
t3-6	x28-31	Geçici değişkenler	Korunmaz

RVfpga Deney 4: Yığın – Gözden Geçirilmiş Kod

C Kodu

```
int main() {  
    ...  
    int y = y + func1(1, 2, 3)  
    y++;  
    ...  
}  
  
int func1(int a, int b, int c) {  
    int sum;  
  
    sum = a + b + c;  
    return sum;  
}
```

RISC-V Çeviricisi

```
# y is in s0  
main: ...  
    addi a0, zero, 1    # put values in argument registers  
    addi a1, zero, 2  
    addi a2, zero, 3  
    jal  func1          # call function func1  
    add  s0, s0, a0      # y = y + return value  
    addi s0, s0, 1      # y = y++  
    ...  
  
# sum is in s0  
func1: addi sp, sp, -4 # make room on stack  
       sw  s0, 0(sp) # save s0 on stack  
    add s0, a0, a1      # sum = a + b  
    add s0, s0, a2      # sum = a + b + c  
    addi a0, s0, 0      # return value = sum  
    lw  s0, 0(sp) # restore s0 from stack  
    addi sp, sp, 4 # restore stack pointer  
    jr   ra             # return
```

Deney 5: C ile Çevirici



RVfpga Deney 5: C ile Çeviriciyi Birleştirme

- **Örnek:** Görüntü işleme programı
- Kimi işlev C dilinde kimisi çevirici dilinde yazılı

RVfpga Deney 5: Görüntü İşleme Programı

- Renkli görüntüyü grili görüntüye dönüştür



RVfpga Deney 5: Görüntü İşleme Programı

- Pikseller üç 8-bit renk olarak depolanır: **R** = kırmızı, **G** = yeşil, **B** = mavi
- İstenilen renk **R**, **G**, **B** değerlerini **çeşitlendirerek** oluşturulabilir
- **Görüntüyü 8-bit griliye (grey) dönüştürmek için** pikseller şöyle dönüştürülür:

$$\text{grey} = (306 * \text{R} + 601 * \text{G} + 117 * \text{B}) \gg 10$$

- RGB ağırlıkları 1024 eder ($306 + 601 + 117 = 1024$), dolayısıyla 8-bit aralığına dönmek için (0-255), sonuç 1024 ile bölünür (diğer deyişle sağa 10 bit kaydırılır: $\gg 10$)
- Algoritma üzerine daha çok detay için şuraya göz at:

<https://www.mathworks.com/help/matlab/ref/rgb2gray.html>

RVfpga Deney 5: Çevirici İşlevi

`.globl ColourToGrey_Pixel` ← `.globl` yönlendirmesi `ColourToGrey_Pixel` işlevini
projedeki bütün dosyalara görünür kılar.

`ColourToGrey_Pixel:`

```
    li x28, 306          # a0 = R * 306
    mul a0, a0, x28
    li x28, 601          # a1 = G * 601
    mul a1, a1, x28
    li x28, 117          # a2 = B * 117
    mul a2, a2, x28
    add a0, a0, a1        # grey = a0 + a1 + a2
    add a0, a0, a2
    srl a0, a0, 10        # grey = grey / 1024
    ret                  # return
.end
```

$$\text{grey} = (306 * \text{R} + 601 * \text{G} + 117 * \text{B}) \gg 10$$

RVfpga Deney 5: yapılarla diziler

```
typedef struct {
    unsigned char R;
    unsigned char G;
    unsigned char B;
} RGB;

extern unsigned char VanGogh_128x128[]; // 1D array of individual RGB values
RGB ColourImage[N][M]; // 2D array of RGB struct (colour image)
unsigned char GreyImage[N][M]; // 2D array of greyscale image

// VanGogh_128.c
unsigned char VanGogh_128x128[] = { 157, // R (pixel [0][0])
                                     182, // G (pixel [0][0])
                                     161, // B (pixel [0][0])
                                     171, // R (pixel [0][1])
                                     195, // G (pixel [0][1])
                                     173, // B (pixel [0][1])
                                     173, // R (pixel [0][2])
                                     ... }
```

RVfpga Deney 5: Main (Ana) İşlevi

```
int main(void) {
    // Create an N x M matrix using the input image
    initColourImage(ColourImage);

    // Transform Colour Image to Grey Image
    ColourToGrey(ColourImage, GreyImage);
    ...
}

void ColourToGrey(RGB Colour[N][M], unsigned char Grey[N][M]) {
    int i, j;

    for (i=0; i<N; i++)
        for (j=0; j<M; j++)
            Grey[i][j] = ColourToGrey_Pixel(Colour[i][j].R, Colour[i][j].G,
                                             Colour[i][j].B);
}
```

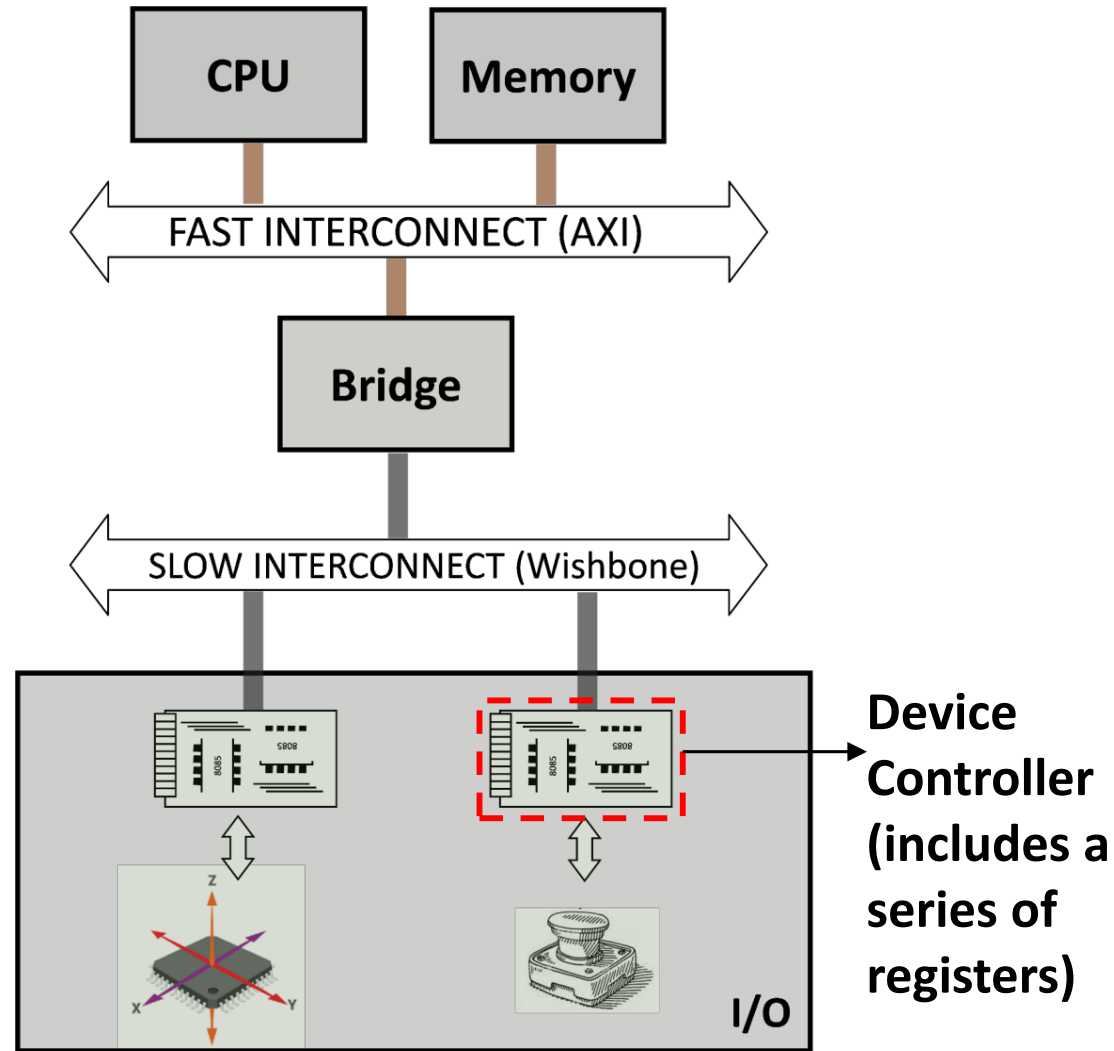
Deney 6: I/O'ya Giriş



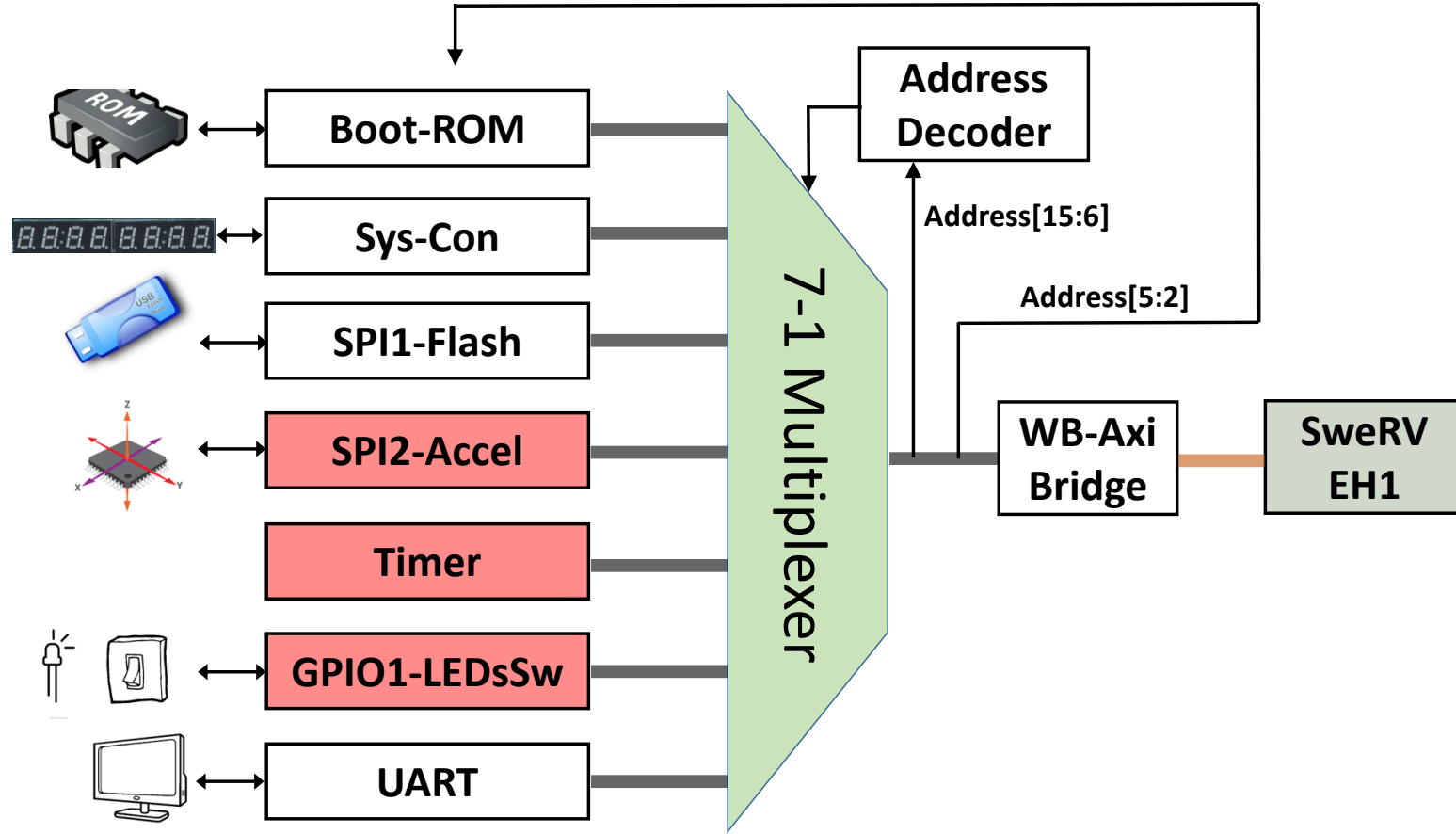
RVfpga Deney 6: Girdi/Çıktıya Giriş

- Girdi/çıkıtı (I/O) sistemleri – *çevre birimleri* de denir
- Genel amaçlı Girdi/Çıkıtı (GPIO)
- GPIO denetleyicileri

RVfpga Deney 6: I/O'lu, Genel Yapılı İşlemci



RVfpga Deney 6: I/O'lu İşlemci



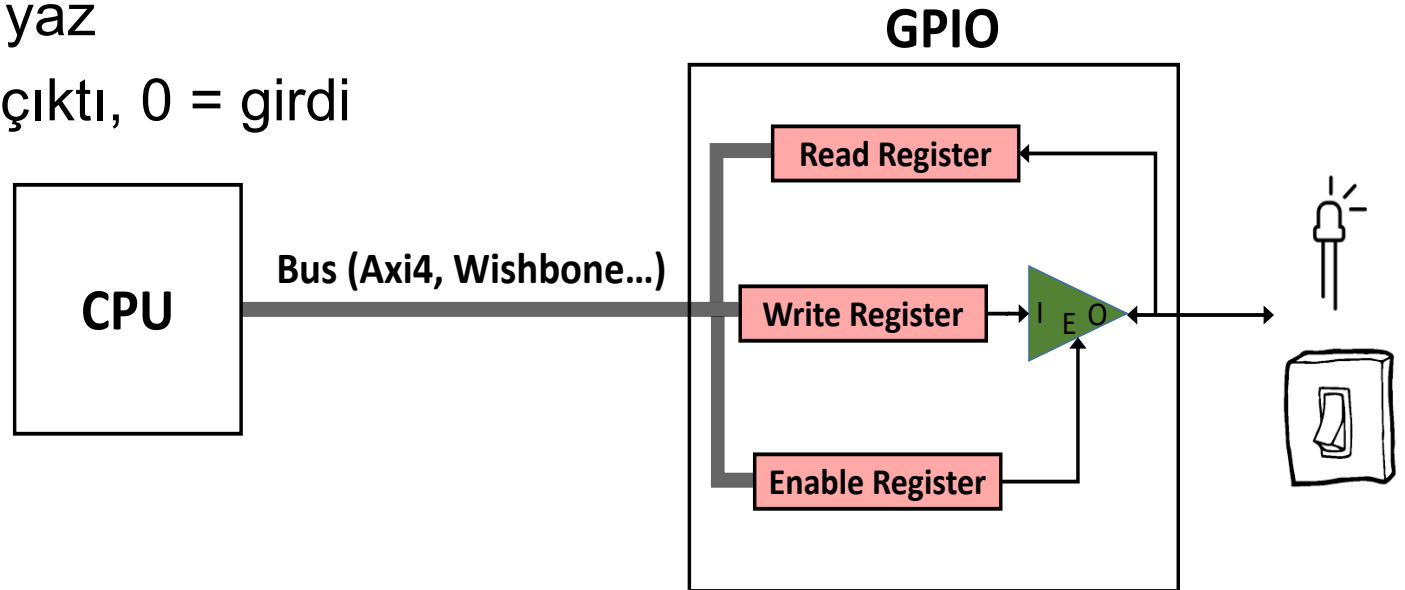
Çevre Birimleri

- **SweRVolf çevre birimleri:**
 - Boot (Önyükleme) ROM
 - Sistem Denetleyicisi
 - SPI1 Flash Bellek
 - UART
- **RVfpga'in ekledikleri:**
 - GPIO LEDler, anahtarlar
 - Zamanlayıcı
 - SPI2 İvmeölçeri
 - 7-kesimli ekranlar (Sistem Denetleyicisi içerisinde: Sys-Con)

RVfpga Deney 6: Genel Amaçlı I/O (GPIO)

- **Genel amaçlı Girdi/Çıktı:**
 - İşlemcinin çevre birimlerine (anahtarlar, LEDler gibi) bağlı uçları okuma/yazma sağlar
 - Üç durum (tri-state) kullanılarak uçlar girdi ya da çıktı olarak yapılandırılabilir
- **Üç bellek-eşlenmiş yazmaç:**
 - **Okuma Yazmacı:** uçtan değeri oku
 - **Yazma Yazmacı:** uca değeri yaz
 - **Etkinleştirme Yazmacı:** 1 = çıktı, 0 = girdi

Peripherals



RVfpga Deney 6: Bellek-Eşlenmiş Yazmaçlar

Yazmaç	Bellek-Eşlenmiş Adres
Okuma Yazmacı	0x80001400
Yazma Yazmacı	0x80001404
Etkinleştirme Yazmacı	0x80001408

- **GPIO'nun 15:0 bitlerini çıktı, 31:16 bitlerini girdi olarak yapılandır:**

```
li t0, 0x80001400 # t0 = 0x80001400
```

```
li t1, 0xFFFF # 1 = çıktı, 0 = girdi
```

```
sw t1, 8(t0) # [15:0] = çıktıları, [31:16] = girdiler
```

- **I/O okuma:**

```
lw t2, 0(t0) # t2 = GPIO uçlarının değeri
```

- **I/O yazma:**

```
sw t3, 4(t0) # GPIO uçları = t3
```

RVfpga Deney 6: RVfpga GPIO Modülü

- **OpenCores'dan Genel Amaçlı Girdi/Çıktı Modülü**

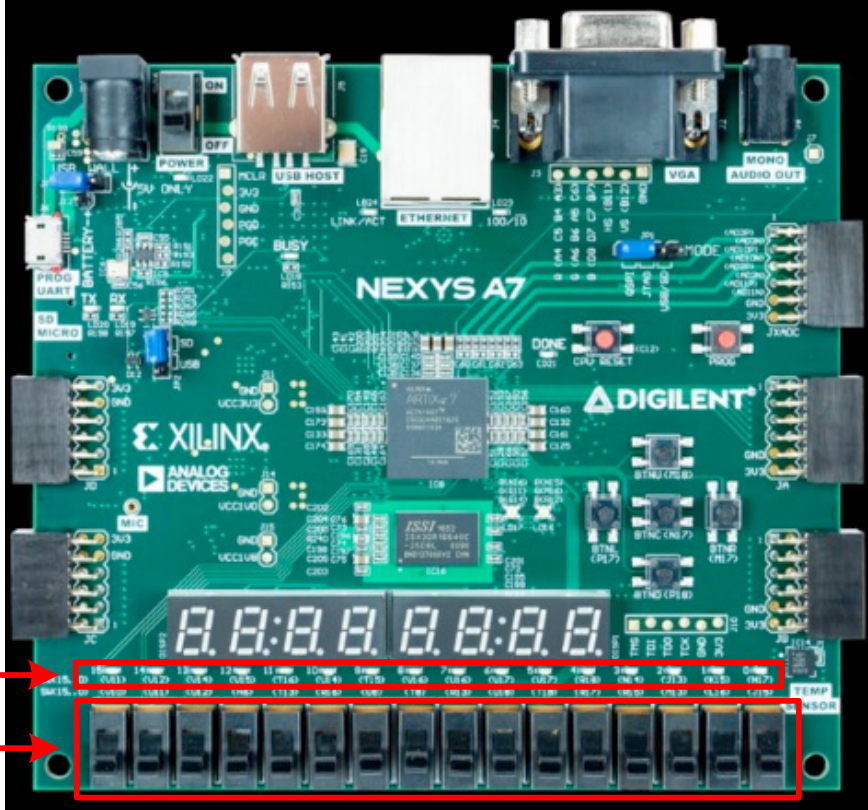
<https://opencores.org/projects/gpio>

- **32 GPIO ucuna kadar kullanılabilir**

- Uçlar birbirinden bağımsız olarak girdi (etkinleştirme = 0) ya da çıktı (etkinleştirme = 1) olarak yapılandırılabilir
- Yapılandırma program süresince değişebilir

Yazmaç	Bellek-Eşlenmiş Adres
Okuma Yazmacı	0x80001400
Yazma Yazmacı	0x80001404
Etkinleştirme Yazmacı	0x80001408

RVfpga Deney 6: Bellek-Eşlenmiş Yazmaçlar



kartın figürü şurada <https://reference.digilentinc.com/>

LEDlerle Anahtarları GPIO uçlarına eşleme:

- LEDler: uçlar [15:0] (işlemcinin çıktıları)
- Anahtarlar (Switches): uçlar [31:16] (işlemcinin girdileri)

GPIO'yu Yapılandır:

- Etkinleştirme Yazmacı = **0x0000FFFF** (1 = çıktı, 0 = girdi)
li t0, 0x80001400
li t1, 0xFFFF
sw t1, 8(t0) # Etkinleştirme Yazmacı = 0xFFFF

LEDlere Yaz:

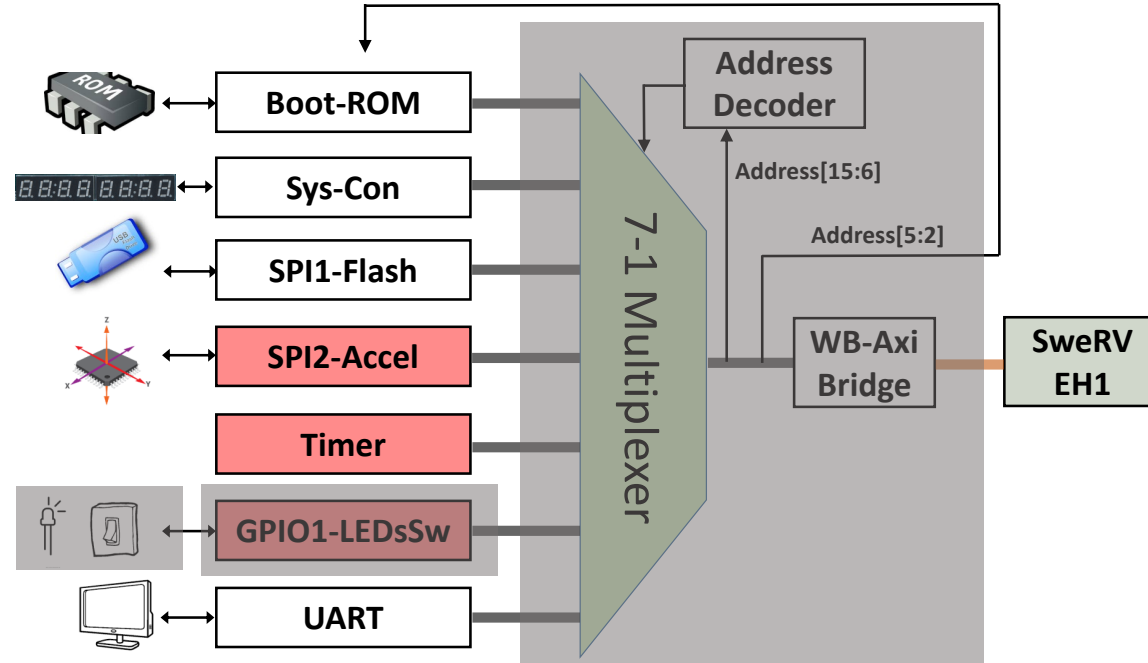
- [15:0] içerisindeki değeri 0x80001404 adresine yaz
sw t3, 4(t0) # LEDs = [t3]_{15:0}

Anahtarları Oku:

- 0x80001400 adresindeki [31:16] bitlerinde anahtarları oku
- Değeri alt 16 bitlere koymak için 16 bit sağa kaydır
lw t5, 0(t0) # [t5]_{31:16} = anahtar değerleri
srli t5, t5, 16 # [t5]_{15:0} = anahtar değerleri

RVfpga Deney 6: GPIO Alçak Düzeyli Gerçekleştirilmesi

- 3 ana parçaya bölünmüştür
 - RVfpga'in karttaki LEDlere/Anahtarlara dış bağlantısı (sol taralı bölge)
 - GPIO modülünün RVfpga'e entegrasyonu (orta taralı bölge)
 - GPIO ile SweRV EH1 arasındaki bağlantı (sağ taralı bölge)



RVfpga Deney 6: Dış bağlantı

Dosya **rvfpga.xdc**: i_sw[15:0]'in karttaki anahtarlarla, o_led[15:0]'in karttaki LEDlerle bağlantısını tanımlar

```
26 set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { i_sw[0] }]
27 set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [get_ports { i_sw[1] }]
28 set_property -dict { PACKAGE_PIN M13 IOSTANDARD LVCMOS33 } [get_ports { i_sw[2] }]
29 set_property -dict { PACKAGE_PIN R15 IOSTANDARD LVCMOS33 } [get_ports { i_sw[3] }]
30 set_property -dict { PACKAGE_PIN R17 IOSTANDARD LVCMOS33 } [get_ports { i_sw[4] }]
31 set_property -dict { PACKAGE_PIN T18 IOSTANDARD LVCMOS33 } [get_ports { i_sw[5] }]
32 set_property -dict { PACKAGE_PIN U18 IOSTANDARD LVCMOS33 } [get_ports { i_sw[6] }]
33 set_property -dict { PACKAGE_PIN R13 IOSTANDARD LVCMOS33 } [get_ports { i_sw[7] }]
34 set_property -dict { PACKAGE_PIN T8 IOSTANDARD LVCMOS18 } [get_ports { i_sw[8] }]
35 set_property -dict { PACKAGE_PIN U8 IOSTANDARD LVCMOS18 } [get_ports { i_sw[9] }]
36 set_property -dict { PACKAGE_PIN R16 IOSTANDARD LVCMOS33 } [get_ports { i_sw[10] }]
37 set_property -dict { PACKAGE_PIN T13 IOSTANDARD LVCMOS33 } [get_ports { i_sw[11] }]
38 set_property -dict { PACKAGE_PIN H6 IOSTANDARD LVCMOS33 } [get_ports { i_sw[12] }]
39 set_property -dict { PACKAGE_PIN U12 IOSTANDARD LVCMOS33 } [get_ports { i_sw[13] }]
40 set_property -dict { PACKAGE_PIN U11 IOSTANDARD LVCMOS33 } [get_ports { i_sw[14] }]
41 set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { i_sw[15] }]
42
43 set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { o_led[0] }]
44 set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { o_led[1] }]
45 set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports { o_led[2] }]
46 set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [get_ports { o_led[3] }]
47 set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports { o_led[4] }]
48 set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { o_led[5] }]
49 set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports { o_led[6] }]
50 set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports { o_led[7] }]
51 set_property -dict { PACKAGE_PIN V16 IOSTANDARD LVCMOS33 } [get_ports { o_led[8] }]
52 set_property -dict { PACKAGE_PIN T15 IOSTANDARD LVCMOS33 } [get_ports { o_led[9] }]
53 set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33 } [get_ports { o_led[10] }]
54 set_property -dict { PACKAGE_PIN T16 IOSTANDARD LVCMOS33 } [get_ports { o_led[11] }]
55 set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33 } [get_ports { o_led[12] }]
56 set_property -dict { PACKAGE_PIN V14 IOSTANDARD LVCMOS33 } [get_ports { o_led[13] }]
57 set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33 } [get_ports { o_led[14] }]
58 set_property -dict { PACKAGE_PIN V11 IOSTANDARD LVCMOS33 } [get_ports { o_led[15] }]
```


RVfpga Deney 6: RVfpga'e Entegrasyon

Dosya **swervolf_core.v**: Üç-durum arabellekleri ile GPIO modül somutlaması

```
bidirec gpio0 (.oe(en_gpio[0]), .inp(o_gpio[0]), .outp(i_gpio[0]), .bidir(io_data[0]));
bidirec gpio1 (.oe(en_gpio[1]), .inp(o_gpio[1]), .outp(i_gpio[1]), .bidir(io_data[1]));
bidirec gpio2 (.oe(en_gpio[2]), .inp(o_gpio[2]), .outp(i_gpio[2]), .bidir(io_data[2]));
bidirec gpio3 (.oe(en_gpio[3]), .inp(o_gpio[3]), .outp(i_gpio[3]), .bidir(io_data[3]));
bidirec gpio4 (.oe(en_gpio[4]), .inp(o_gpio[4]), .outp(i_gpio[4]), .bidir(io_data[4]));
bidirec gpio5 (.oe(en_gpio[5]), .inp(o_gpio[5]), .outp(i_gpio[5]), .bidir(io_data[5]));
bidirec gpio6 (.oe(en_gpio[6]), .inp(o_gpio[6]), .outp(i_gpio[6]), .bidir(io_data[6]));
bidirec gpio7 (.oe(en_gpio[7]), .inp(o_gpio[7]), .outp(i_gpio[7]), .bidir(io_data[7]));
bidirec gpio8 (.oe(en_gpio[8]), .inp(o_gpio[8]), .outp(i_gpio[8]), .bidir(io_data[8]));
bidirec gpio9 (.oe(en_gpio[9]), .inp(o_gpio[9]), .outp(i_gpio[9]), .bidir(io_data[9]));
bidirec gpio10 (.oe(en_gpio[10]), .inp(o_gpio[10]), .outp(i_gpio[10]), .bidir(io_data[10]));
bidirec gpio11 (.oe(en_gpio[11]), .inp(o_gpio[11]), .outp(i_gpio[11]), .bidir(io_data[11]));
bidirec gpio12 (.oe(en_gpio[12]), .inp(o_gpio[12]), .outp(i_gpio[12]), .bidir(io_data[12]));
bidirec gpio13 (.oe(en_gpio[13]), .inp(o_gpio[13]), .outp(i_gpio[13]), .bidir(io_data[13]));
bidirec gpio14 (.oe(en_gpio[14]), .inp(o_gpio[14]), .outp(i_gpio[14]), .bidir(io_data[14]));
bidirec gpio15 (.oe(en_gpio[15]), .inp(o_gpio[15]), .outp(i_gpio[15]), .bidir(io_data[15]));
bidirec gpio16 (.oe(en_gpio[16]), .inp(o_gpio[16]), .outp(i_gpio[16]), .bidir(io_data[16]));
bidirec gpio17 (.oe(en_gpio[17]), .inp(o_gpio[17]), .outp(i_gpio[17]), .bidir(io_data[17]));
bidirec gpio18 (.oe(en_gpio[18]), .inp(o_gpio[18]), .outp(i_gpio[18]), .bidir(io_data[18]));
bidirec gpio19 (.oe(en_gpio[19]), .inp(o_gpio[19]), .outp(i_gpio[19]), .bidir(io_data[19]));
bidirec gpio20 (.oe(en_gpio[20]), .inp(o_gpio[20]), .outp(i_gpio[20]), .bidir(io_data[20]));
bidirec gpio21 (.oe(en_gpio[21]), .inp(o_gpio[21]), .outp(i_gpio[21]), .bidir(io_data[21]));
bidirec gpio22 (.oe(en_gpio[22]), .inp(o_gpio[22]), .outp(i_gpio[22]), .bidir(io_data[22]));
bidirec gpio23 (.oe(en_gpio[23]), .inp(o_gpio[23]), .outp(i_gpio[23]), .bidir(io_data[23]));
bidirec gpio24 (.oe(en_gpio[24]), .inp(o_gpio[24]), .outp(i_gpio[24]), .bidir(io_data[24]));
bidirec gpio25 (.oe(en_gpio[25]), .inp(o_gpio[25]), .outp(i_gpio[25]), .bidir(io_data[25]));
bidirec gpio26 (.oe(en_gpio[26]), .inp(o_gpio[26]), .outp(i_gpio[26]), .bidir(io_data[26]));
bidirec gpio27 (.oe(en_gpio[27]), .inp(o_gpio[27]), .outp(i_gpio[27]), .bidir(io_data[27]));
bidirec gpio28 (.oe(en_gpio[28]), .inp(o_gpio[28]), .outp(i_gpio[28]), .bidir(io_data[28]));
bidirec gpio29 (.oe(en_gpio[29]), .inp(o_gpio[29]), .outp(i_gpio[29]), .bidir(io_data[29]));
bidirec gpio30 (.oe(en_gpio[30]), .inp(o_gpio[30]), .outp(i_gpio[30]), .bidir(io_data[30]));
bidirec gpio31 (.oe(en_gpio[31]), .inp(o_gpio[31]), .outp(i_gpio[31]), .bidir(io_data[31]));
```

```
gpio_top gpio_module(
    .wb_clk_i      (clk),
    .wb_rst_i      (wb_rst),
    .wb_cyc_i      (wb_m2s_gpio_cyc),
    .wb_adr_i      ({2'b0,wb_m2s_gpio_adr[5:2],2'b0}),
    .wb_dat_i      (wb_m2s_gpio_dat),
    .wb_sel_i      (4'b1111),
    .wb_we_i       (wb_m2s_gpio_we),
    .wb_stb_i      (wb_m2s_gpio_stb),
    .wb_dat_o      (wb_s2m_gpio_dat),
    .wb_ack_o      (wb_s2m_gpio_ack),
    .wb_err_o      (wb_s2m_gpio_err),
    .wb_inta_o     (gpio_irq),
    // External GPIO Interface
    .ext_pad_i     (i_gpio[31:0]),
    .ext_pad_o     (o_gpio[31:0]),
    .ext_padoe_o   (en_gpio));
```

RVfpga Deney 6: SweRV EH1 ile Bağlantı

Dosya wb_intercon.v: 7-1 Çoklayıcı gerçekleştirilmesi

```
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'hffff000, 32'hfffffc0, 32'hfffffc0, 32'hfffffc0, 32'hfffffc0, 32'hfffffc0, 32'hffff000}))
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_rom_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
```

CPU/Controller Signals

Peripheral Signals

Deney 7:

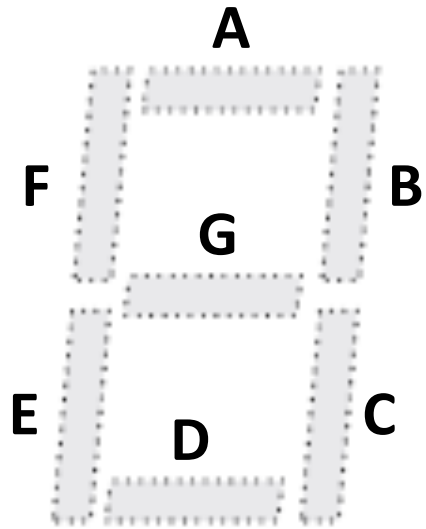
7-Kesimli Ekranlar



RVfpga Deney 7: 7-Kesimli Ekranlar

- 7-kesimli ekranların tanıtımı
- 7-kesimli ekran donanımı

RVfpga Deney 7: 7-Kesimli Ekranların Tanıtımı



- 7 LED kesimi: A-G
- Belirli sayıyı oluşturmak için kesimler yakılır
 - 1: kesimler B, C
 - 2: kesimler A, B, D, E, G
 - 3: kesimler A, B, C, D, G
 - gibi.

RVfpga Deney 7: Nexys A7'daki 7-Kesimli Ekranlar

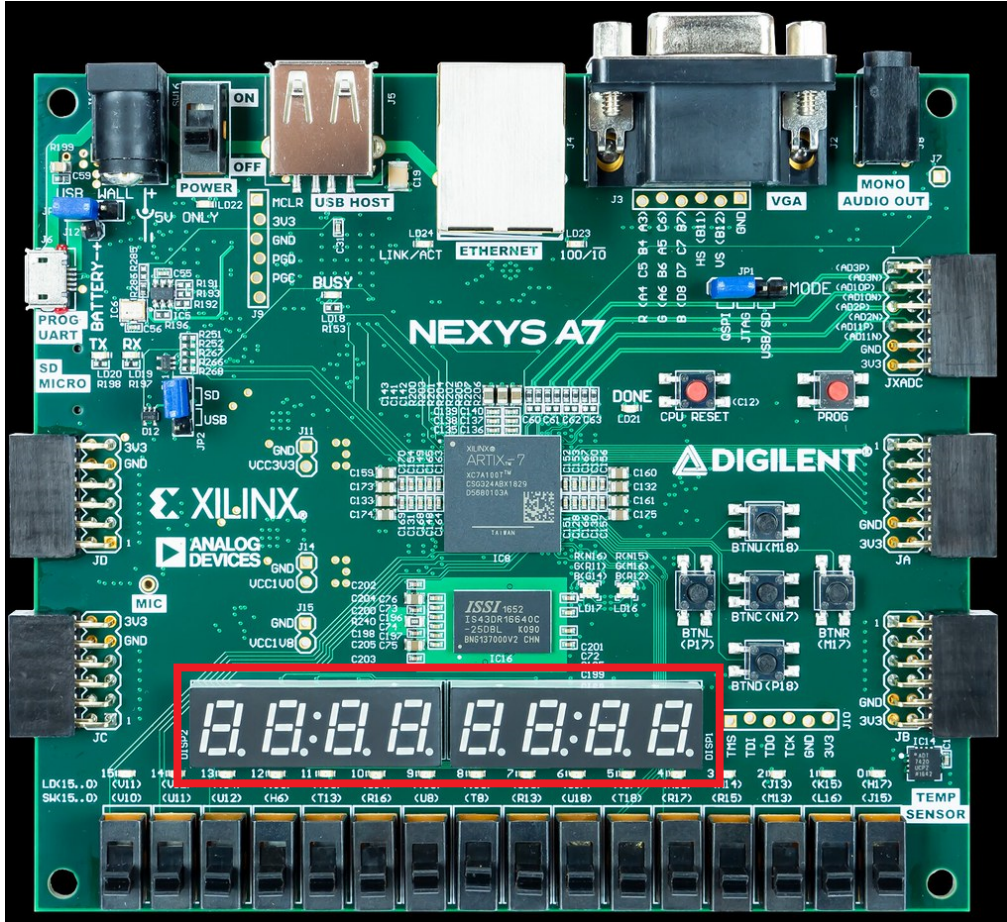


figure of board from <https://reference.digilentinc.com/>

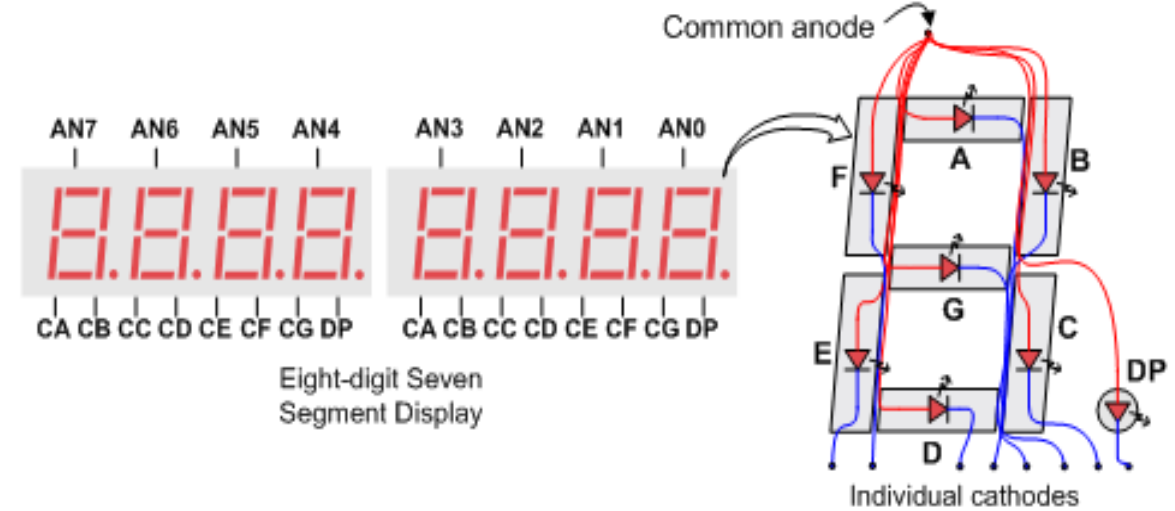
- **8-sayı 7-kesim ekran**
- **Bellek-eşlenmiş erişim:**
 - **Enables_Reg:** 0x80001038
 - **Digits_Reg:** 0x8000103C
- **Etkinleştirmeler alçakta-sağlamalıdır**
- **Örnek:** En sağdaki iki sayıda 71'i göster:
 - **Enables_Reg** = 0xFC (0b11111100: en sağdaki iki sayıyı etkinleştir)
 - **Digits_Reg** = 0x71
 - **Çevirici:**

```
li t0, 0x80001038
li t1, 0xFC
li t2, 0x71
sw t1, 0(t0)
sw t2, 4(t0)
```

RVfpga Deney 7: 7-Kesimli Ekran Donanımı

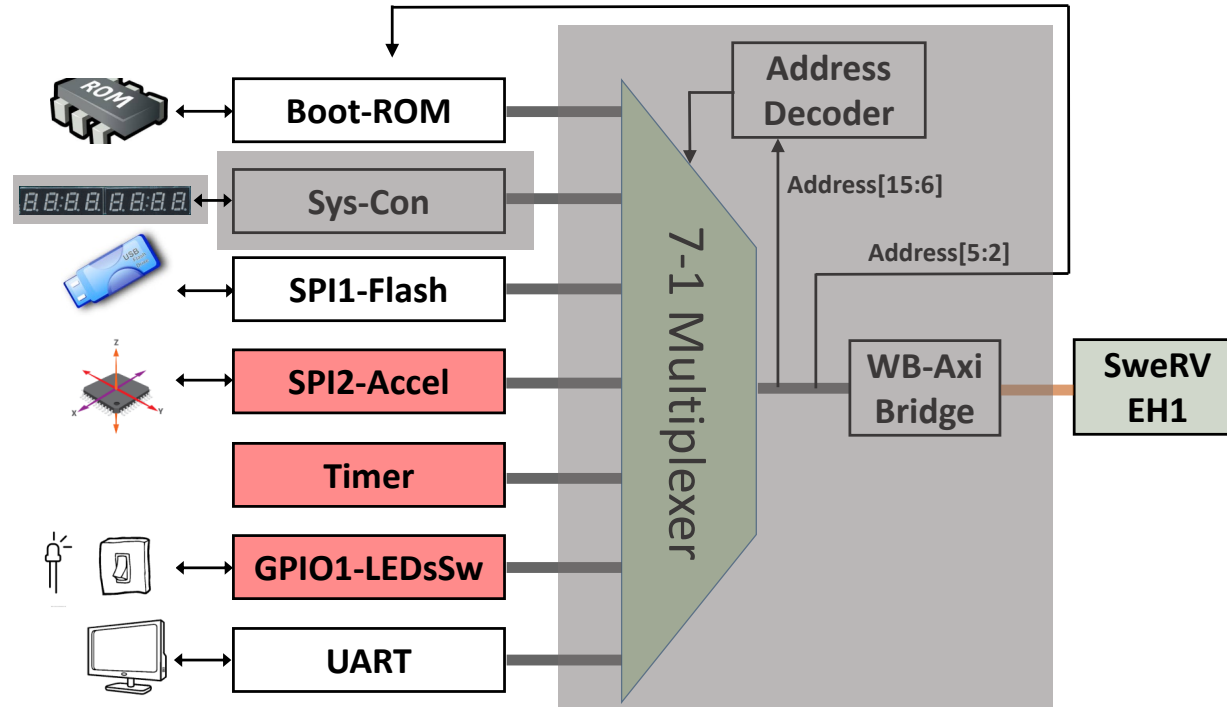
- Bütün sayılar birer **yaygın anot**dur (o sayının LEDlerinin anotları birbirine bağlıdır)
 - Anot sinyalleri **etkinleştirme** olarak davranır (AN0 - AN7)
 - Sayıyı etkinleştirmek için **düşüğe** sür (AN0 - AN7 LED'e baselenmeden önce tersleyiciden geçer (gösterilmemiştir))
- Sayılar için bütün **kesimler birbirlerine** bağlıdır
 - Kesimler açmak için **düşüğe** sürülür
 - AN0 - AN7 sinyallerinin **zaman-çoklayıcılanması** sayılarda eşsiz değerlerin gösterilmesini sağlar
 - Bir sayının AN sinyali (AN0 - AN7) parlaması için **1-16ms'de** bir düşüğe gitmelidir

8-Sayı 7-Kesimli Ekranlar



RVfpga Deney 7: 7-Kes. Ekr. Alçak-Düzey Gerçekleştirme

- 3 ana parçaya bölünmüştür
 - RVfpga'in karttaki 7-kesimli ekranlara dış bağlantısı (sol taralı bölge)
 - 7-kesimli ekran modülünün RVfpga'e entegrasyonu (orta taralı bölge)
 - 7-kesimli ekranlarla SweRV EH1 bağlantısı (sağ taralı bölge)



RVfpga Deney 7: Dış bağlantı

Dosya **rvfpga.xdc**: CA-CG (SoC'de Digits_Bits[i] denir) ile AN[i]'nin karttaki 7-kesimli ekranlarla bağlantısını tanımlar

```
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]
77
```

RVfpga Deney 7: RVfpga'ye Entegrasyonu

- Dosya **swervolf_syscon.v**: 7-kesimli ekran denetleyicisinin somutlaması. Modül, saat sinyaliyle (i_clk) sıfırlama sinyalinin (i_rst) üzerine, iki girdi sinyali de - **Enables_Reg** ile **Digits_Reg** - alır, ki bunlar önceden tanımlanmış olan iki bellek-eşlenmiş denetleme yazmaçlarıdır. Bu modül iki sinyal çıktısı verir - **AN** ile **Digits_Bits** - ki bunlar karttaki 7-kesimli ekranlara bağlıdır.

```
// Eight-Digit 7 Segment Displays

reg  [ 7:0]  Enables_Reg;
reg  [31:0]  Digits_Reg;

SevSegDisplays_Controller SegDispl_Ctr(
    .clk          (i_clk),
    .rst_n        (i_rst),
    .Enables_Reg  (Enables_Reg),
    .Digits_Reg   (Digits_Reg),
    .AN           (AN),
    .Digits_Bits  (Digits_Bits)
);
```

RVfpga Deney 7: RVfpga'ye Entegrasyonu

- **SevSegDisplays_Controller** da bu dosyada gerçekleştirilmiştir. Şu alt üniteleri barındırır:
 - 2ms'de bir AN ile Digits_Bits sinyallerine gönderilecek değeri seçen iki çoklayıcı (modül **SevSegMux**).
 - 2ms'lik periyot oluşturan bir sayaç (modül **counter**).
 - Verilen 4-bit onaltılık değer için kesim değerlerini çıkaran bir çözücü (modül **SevenSegDecoder**).

Deney 8:

Zamanlayıcılar



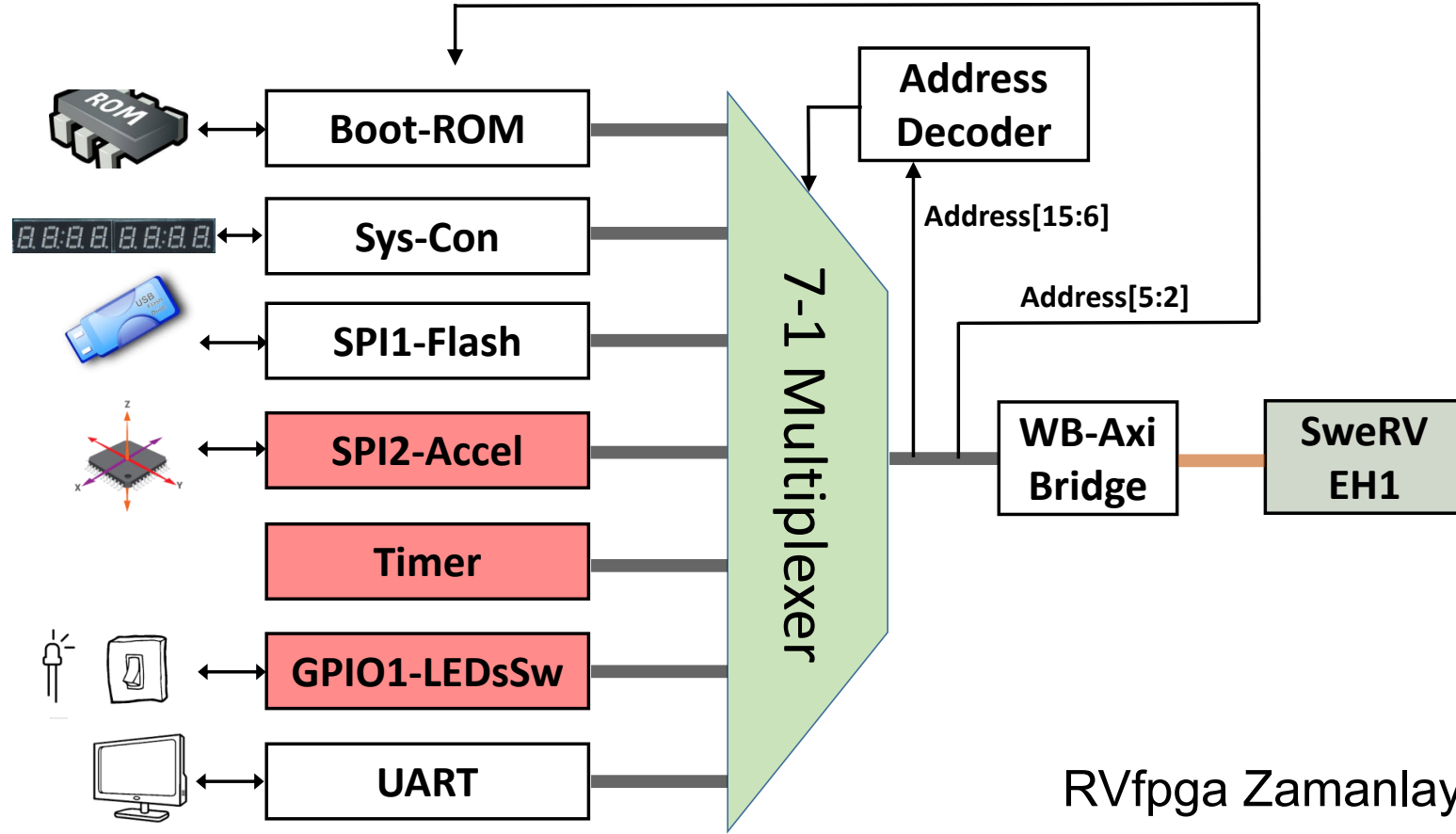
RVfpga Deney 8: Zamanlayıcılar

- Zamanlayıcıların tanıtımı
- Zamanlayıcı Yazmaçları
- Zamanlayıcı Örnekleri

RVfpga Deney 8: Zamanlayıcılar

- Zamanlayıcılar **bir sayacı** sabit bir sıklıkta **arttırır ya da azaltır**
- **Mikrodenetleyicilerde**, SoClerde (yongadaki sistem) yaygın olarak bulunur
- **Kesin zamanlama** oluşturmak için kullanılır

RVfpga Deney 8: Zamanlayıcılar



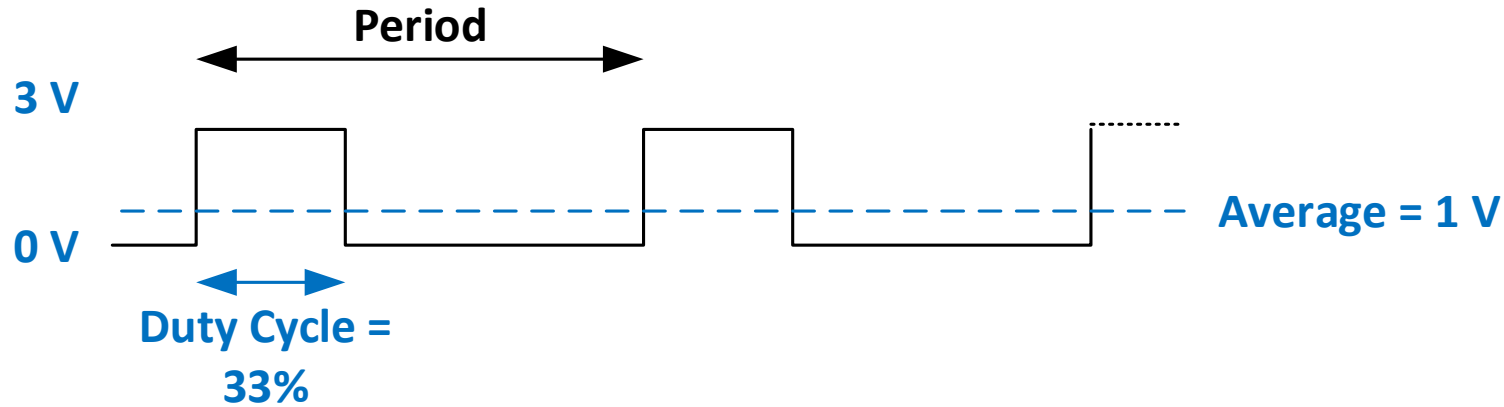
Çevre Birimleri

RVfpga Zamanlayıcı modülü OpenCores'dandır:

<https://opencores.org/projects/ptc>

RVfpga Deney 8: Zamanlayıcı (PTC) Modülü

- Zamanlayıcı modülü (**PTC** modülü de denir) şunun için kullanılır:
 - Timer (Zamanlayıcı) / Counter (Sayaç): saat sinyallerinin kenarlarını sayar (ya da başka sinyallerin kenarlarını, etkinlikler de denir)
 - Pulse-width modulation (Darbe-genişlik modülasyonu) (PWM):
 - Bir çıktının vary high duration'ı (*duty cycle* (*görev dönümü*)) denir
 - Analog voltajın dijital yaklaşımını yapmak için kullanılır
- **PWM örneği:** 33% görev dönümü (sinyal sürenin 1/3'ünde yüksek). Yüksek düzeyi 3 V ise, analog voltaj (sinyalin ortalama voltajı) $3\text{ V} * 0.33 = 1\text{ V}$ eder



RVfpga Deney 8: Zamanlayıcı (PTC) Yazmaçlar

Adı	Adresi	Geniřlięi	Eriřim	Tanım
RPTC_CNTR	0x80001200	1-32	R/W	Ana PTC sayacı
RPTC_HRC	0x80001204	1-32	R/W	PTC HI Referans/Yakalama yazmacı
RPTC_LRC	0x80001208	1-32	R/W	PTC LO Referans/Yakalama yazmacı
RPTC_CTRL	0x8000120C	9	R/W	Denetleme yazmacı

- **RPTC_CNTR:** Sayaç (sayacın değeri)
- **RPTC_HRC:** Yüksek referans yakalama - PWM modunda (sıfırlama sonrası) çıktının yükseęe gitmesi gereken dönüm sayısını gösterir
- **RPTC_LRC:** Düşük referans yakalama - sayaç/zamanlayıcı modunda sayma bittiğinde (sıfırlama sonrası) dönüm sayısını gösterir; (sıfırlama sonrası) PWM modunda çıktının düşüęe gitmesi gereken saat dönümü sayısını gösterir.
- **RPTC_CTRL:** Denetleme yazmacı

RVfpga Deney 8: Zamanlayıcı (PTC) Denetleme Yazmacı

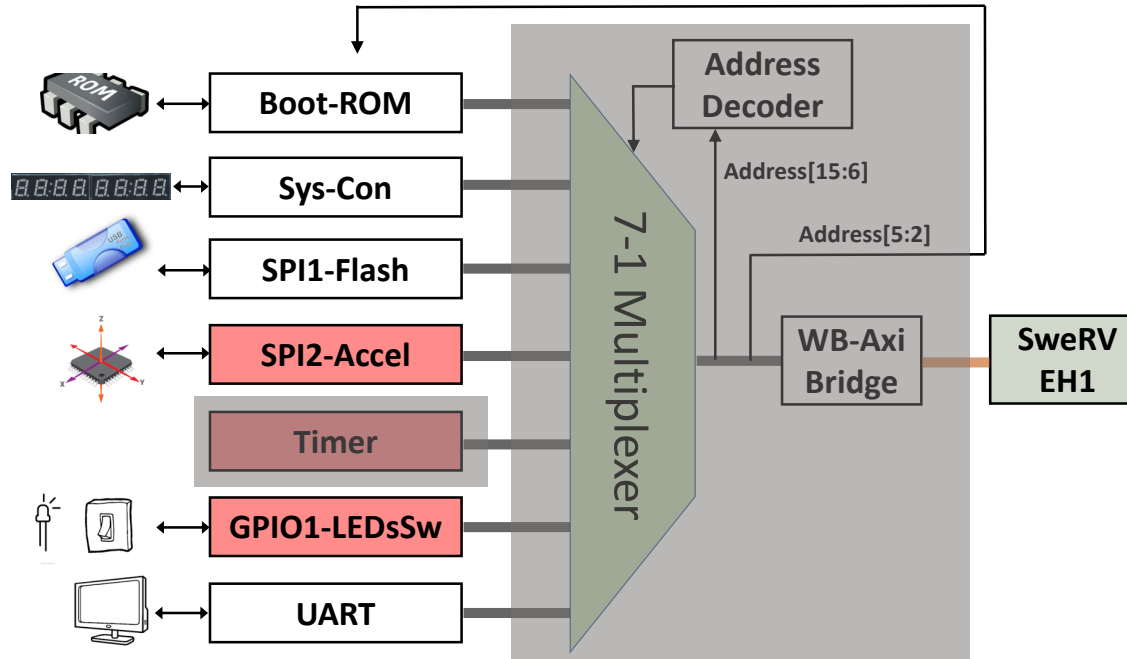
Bit	Erişim	Sıfır	Adı & Tanımı
0	R/W	0	EN: Ayarlanınca RPTC_CNTR artar.
1	R/W	0	ECLK: Saat sinyalini seçer: dış saat, ptc_ecgt (1) üzerinden, ya da sistem saati(0).
2	R/W	0	NEC: Negatif/pozitif kenarı seçmeyle dış saatin düşük/yüksek periyotunu seçme için kullanılır (ptc_ecgt).
3	R/W	0	OE: PWM çıktı sürücüsünü etkinleştirir.
4	R/W	0	SINGLE: Ayarlanınca RPTC_CNTR, RPTC_LRC değerine ulaştıktan sonra arttırılmaz. Boşaltılınca RPTC_CNTR, RPTC_LRC yazmacındaki değere ulaşınca yeniden başlatılır.
5	R/W	0	INTE: Ayarlanınca RPTC_CNTR değeri RPTC_LRC ya da RPTC_HRC değerine eşitlenince PTC bir kesinti sağlar. Sinyal boşaltılınca kesintiler maskelenir.
6	R/W	0	INT: Okununca bu bit bekleyen kesintiyi gösterir. Ayarlıysa bir kesinti beklemededir. Bu bit '1' ile yazılınca kesinti isteği boşaltılmıştır.
7	R/W	0	CNTRRST: Ayarlanınca RPTC_CNTR sıfırlanır. Boşaltılınca sayaç normal işlem yapar.
8	R/W	0	CAPTE: Ayarlanınca RPTC_CNTR, RPTC_LRC ya da RPTC_HRC yazmaçları içerisine yakalanır. Sıfırlanınca yakalama işlevi maskelenir.

RVfpga Deney 8: Zamanlayıcı Örneği

- **RPTC_LRC'i** sayılacak dönümlerin sayısına **ayarla**
- **Zamanlayıcıyı yapılandırmak için denetleme bitlerini (RPTC_CTRL) ayarla:**
 - **Sayacı sıfırla, kesintileri sıfırla:** **RPTC_CTRL = 0xC0** (0b011000000): CNTRRST (bit 7) = 1: sayaç sıfırlanır (RPTC_CNTR = 0); INT (bit 6) = 1: kesinti isteği sıfırlanır.
 - **Sayaçla kesintileri etkinleştir:** **RPTC_CTRL = 0x21** (0b000100001): EN (bit 0) = 1: sayaç etkinleştirilir, dolayısıyla RPTC_CNTR artar; INTE (bit 5) = 1: PTC, RPTC_CNTR == RPTC_LRC olunca kesinti sağlar.
- Program denetleme yazmacında kesinti bitini (**INT, RPTC_CTRL'nin bit 6'sıdır**) 1 olana kadar okur (RPTC_CNTR == RPTC_LRC olduğunu gösterir).
- Bu algoritma **kesintileri kullanmaz**, ancak kesinti bitini (INT, RPTC_CTRL'nin bit 6'sı) doğru saat dönüm sayısına erişildiğini belirlemek için kullanır. Kesinti kullanımını Deney 9'da gösteriyoruz.

RVfpga Deney 8: Zamanlayıcı Alçak Düzeyli Gerçekleştirilmesi

- 2 ana parçaya bölünmüştür
 - (Dış bağlantı yok)
 - Zamanlayıcı modülünün RVfpga'ye entegrasyonu (orta taralı bölge)
 - Zamanlayıcıyla SweRV EH1 arasındaki bağlantı (sağ taralı bölge)



RVfpga Deney 8: RVfpga'e Entegrasyon

Dosya `swervolf_core.v`: PTC modül somutlama

```
// PTC
wire          ptc_irq;

ptc_top timer_ptc(
    .wb_clk_i      (clk),
    .wb_rst_i      (wb_rst),
    .wb_cyc_i      (wb_m2s_ptc_cyc),
    .wb_adr_i      ({2'b0,wb_m2s_ptc_adr[5:2],2'b0}),
    .wb_dat_i      (wb_m2s_ptc_dat),
    .wb_sel_i      (4'b1111),
    .wb_we_i      (wb_m2s_ptc_we),
    .wb_stb_i      (wb_m2s_ptc_stb),
    .wb_dat_o      (wb_s2m_ptc_dat),
    .wb_ack_o      (wb_s2m_ptc_ack),
    .wb_err_o      (wb_s2m_ptc_err),
    .wb_inta_o     (ptc_irq),
    // External PTC Interface
    .gate_clk_pad_i (),
    .capt_pad_i  (),
    .pwm_pad_o  (),
    .oen_padoen_o ()
);
```

Deney 9:

Kesinti-Güdümlü I/O



RVfpga Deney 9: Kesinti-Güdümlü I/O

- Kesinti-güdümlü Girdi/Çıktı vs. Programlanmış Girdi/Çıktı
- RVfpga'in Kesinti Denetleyicisi
- Western Digital'in Çevre Birimi Destek ile Kart Destek Paketlerini (PSP ile BSP) kullanarak kesintiler nasıl yapılandırılır
- Kesinti Örneği

RVfpga Deney 9: Kesinti-Güdümlü I/O Giriş

- **Programlanmış Girdi/Çıktı:**

- İstenilen değer görülene kadar program sürekli bir değeri yoklar (örneğin anahtarı).
- Örneğin bu yöntem önceki deneylerde anahtarları okumak için kullanılmıştı.
- Bu bir değeri sürekli yoklayarak işlemciyi zorlar – başka işe yarar çalışmalar yapmak yerine.

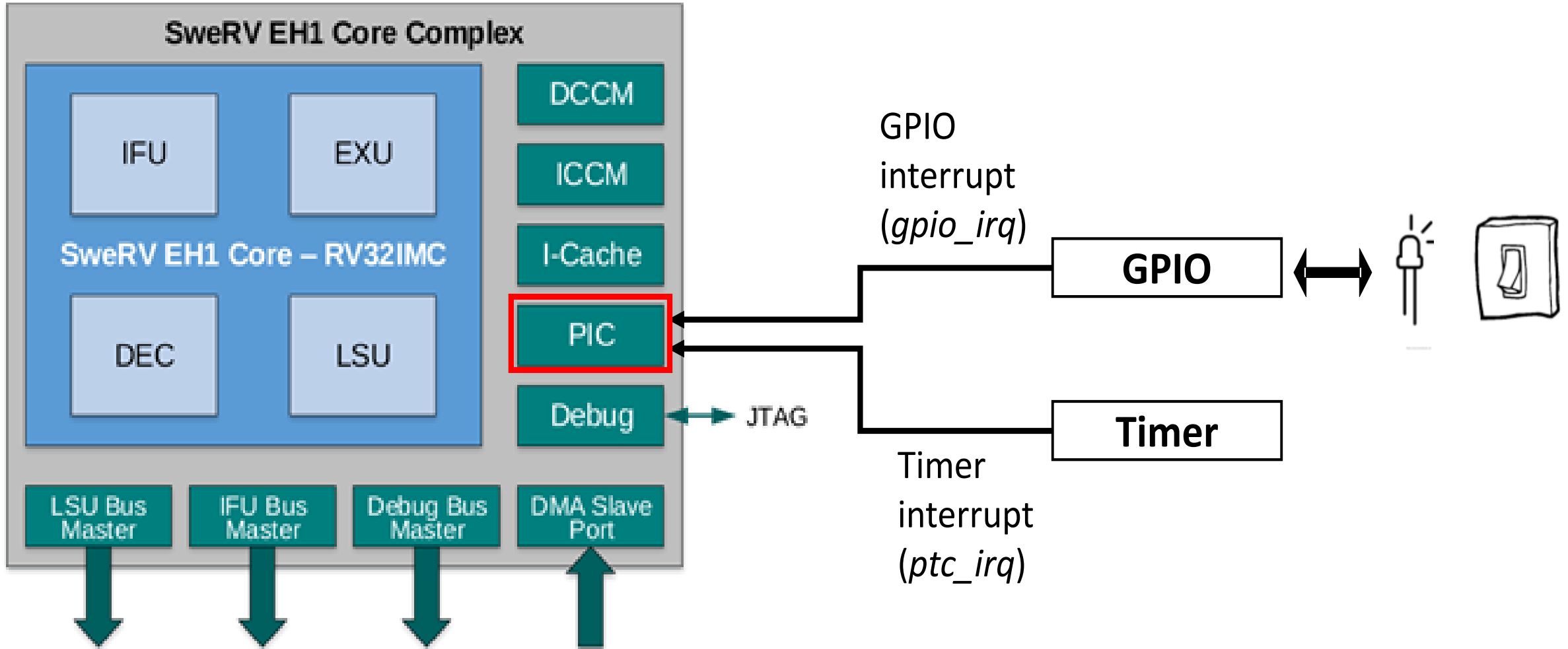
- **Kesinti-güdümlü Girdi/Çıktı:**

- Bir etkinlik (uç sağlaması gibi) işlemcinin kesinti hizmet yordamına (ISR, kesinti *yöneticisi* de denir) sıçramasına neden olur, bu da plansız bir işlev çağırısı gibidir. ISR kesintiyi yönetir - örneğin anahtarların değerini okur - ardından normal programa döner.
- Bu olay olana kadar işlemci anlamlı işler yapmayı sürdürebilir.

RVfpga Deney 9: Kesintileri Yönetme

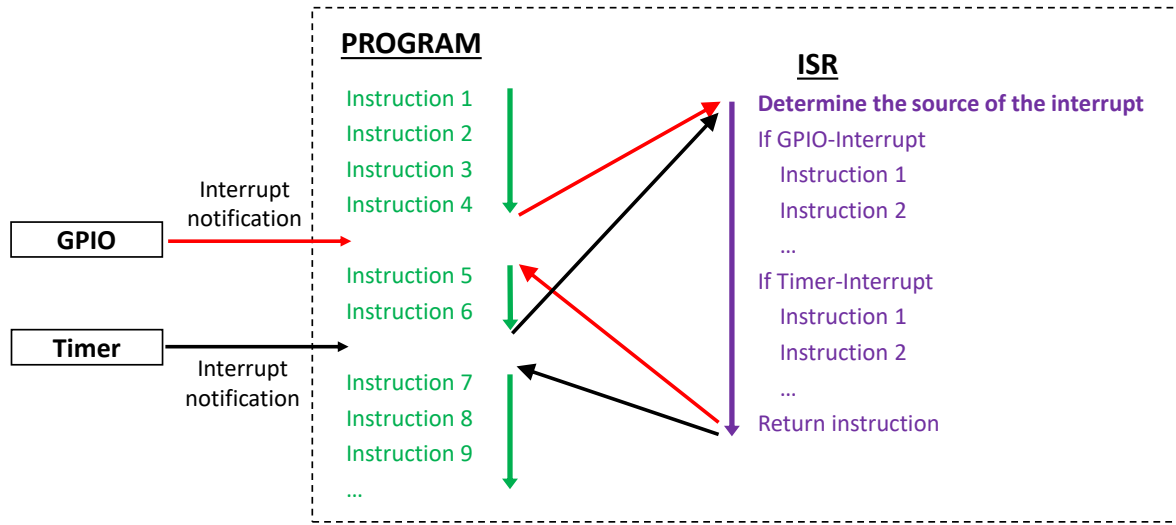
- **Donanım** ya da **yazılım** kesintilere neden olabilir
- Bu deneyde **donanım kesintilerine** odaklanıyoruz
- SweRV EH1 çekirdeği kesintileri RISC-V'in PLIC (Platform-level interrupt controller (Platform düzeyli kesinti denetleyicisi)) spesifikasyonuna göre yönetir. Buna Programmable Interrupt Controller (Programlanabilir Kesinti Denetleyicisi) (**PIC**) denir. Şunları vardır:
 - 255 kesinti kaynağı
 - 15 öncelik düzeyi

RVfpga Deney 9: Kesinti Donanımı

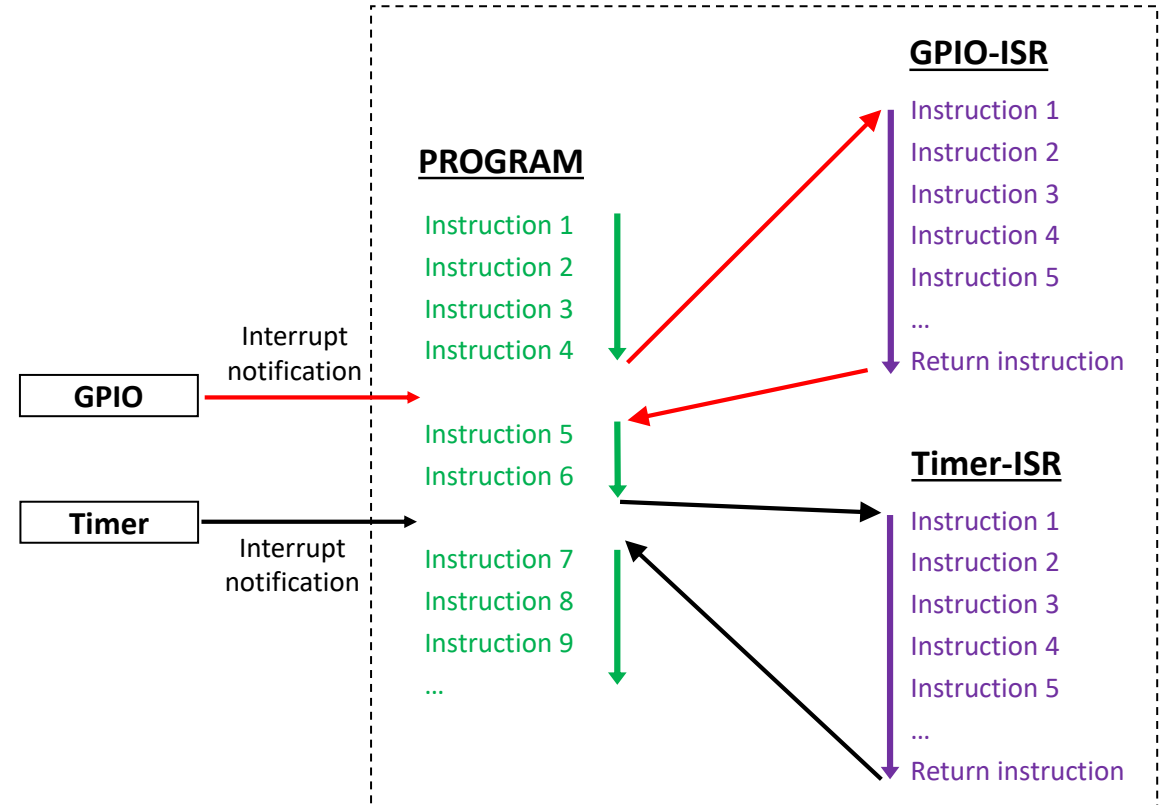


RVfpga Deney 9: Bir-vektör vs. Çoklu-vektör modu

Bir-vektör modu örneği:



Çoklu-vektör modu örneği:



RVfpga Deney 9: Kesintileri Yönetme

- WD'in PSP/BSP'sini kullanarak:
 - WD'in PSP/BSP'sini kullanarak kesintileri ilk değerlendir
 - 255 kesintiden birini ya da daha çoğunu ilk değerlendirip ISR'ın adını sağla
 - Kesintiyi tetikleyecek çevre birimi sinyalini kesinti ucuyla bağla.
 - Bütün kesintileri etkinleştir
 - Dış kesintileri etkinleştir

RVfpga Deney 9: Kesinti Örneği

- Switch[0]'ın değerini okumak için kesintileri kullan – yalnızca yükselen kenarda (0→1 geçişi)

Adı	Adresi	Genişliği	Erişim	Tanım
RGPIO_IN	0x80001400	1-32	R	GPIO girdi veri
RGPIO_OUT	0x80001404	1-32	R/W	GPIO çıktı veri
RGPIO_OE	0x80001408	1-32	R/W	GPIO çıktı sürücü etkinleştir
RGPIO_INTE	0x8000140C	1-32	R/W	Kesinti etkinleştir
RGPIO_PTRIG	0x80001410	1-32	R/W	Kesintiyi tetikleyen etkinlik türü
RGPIO_AUX	0x80001414	1-32	R/W	GPIO çıktılarına yardımcı girdileri çokla
RGPIO_CTRL	0x80001418	2	R/W	Denetleme yazmacı
RGPIO_INTS	0x8000141C	1-32	R/W	Kesinti durumu
RGPIO_ECLK	0x80001420	1-32	R/W	RGPIO_IN sürgülemesi için gpio_eclk'yi etkinleştir
RGPIO_NEC	0x80001424	1-32	R/W	gpio_eclk'nin etkin kenarını seç

Bütün kod için şuraya bak: [RVfpgaPath]/RVfpga/Labs/Lab9/LED-Switch_7SegDispl_Interrupts_C-Lang.c

RVfpga Deney 9: Kesinti Örneği

- Kesintiler için GPIO yazmaçlarını ayarlama:
 - RGPIO_INTE = 0x10000 (Switch[0] için kesintiyi etkinleştir)
 - RGPIO_PTRIG = 0x10000 (Switch[0] yükselen kenarında kesinti tetiklenir)
 - RGPIO_INTS = 0x0 (bütün kesintileri boşaltır)
 - RGPIO_CTRL = 0x1 (GPIO kesintilerini etkinleştirir)

RVfpga Deney 9: Kesinti Örneği

- GPIO ISR:

```
void GPIO_ISR(void) {
    unsigned int i;

    /* Invert LED value */
    i = M_PSP_READ_REGISTER_32(GPIO_LEDS);      /* RGPIO_OUT */
    i = !i;                                       /* Invert the LEDs */
    i = i & 0x1;                                  /* Only keep right-most LED */
    M_PSP_WRITE_REGISTER_32(GPIO_LEDS, i)       /* RGPIO_OUT */

    /* Clear GPIO interrupt */
    M_PSP_WRITE_REGISTER_32(RGPIO_INTS, 0x0); /* RGPIO_INTS */

    /* Stop the generation of this interrupt (IRQ4) */
    bspClearExtInterrupt(4);
}
```

RVfpga Deney 9: Kesinti Örneği

- Kesinti 4'ü (IRQ4) anahtarın kesintisiyle bağla, hizmet yordamını GPIO_ISR olarak ayarla
- Bellek-eşlenmiş yazmaç $0x80001018 = 0x1$: GPIO kesintisini RVfpga donanımında IRQ4'e bağlar
- Genel kesintileri etkinleştir

Deney 10:

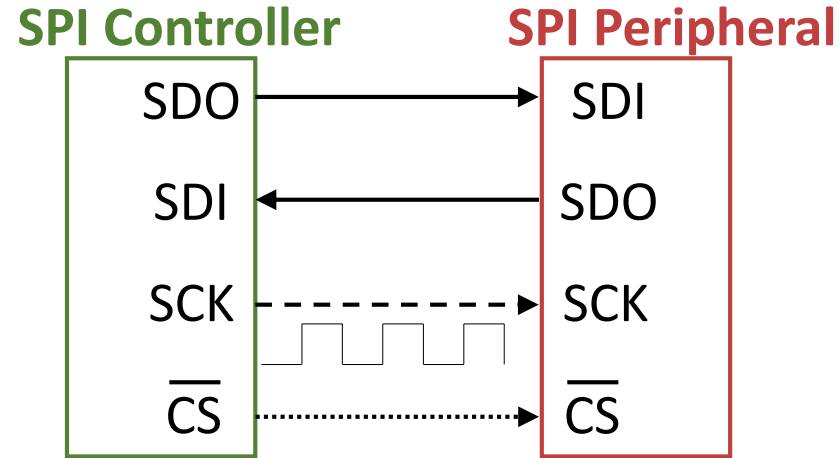
Dizisel Veri Yolları



RVfpga Deney 10: Dizisel Veri Yolları

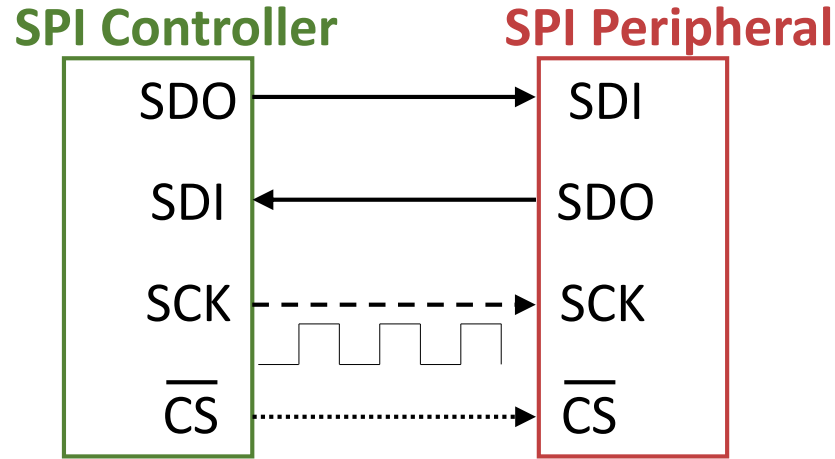
- Dizisel veri yolları **bitleri birer birer** yollar
 - Buna karşın paralel veri yolları **bitleri çoklu** yollar
- **Yaygın dizisel veri yolları**
 - **UART** (evrensel eş zamansız alıcı/verici)
 - **SPI** (dizisel çevre birimi arayüzü)
 - **I2C** (entegre devreler arası protokol)
- Bu deneyde **SPI'a** odaklanıyoruz

RVfpga Deney 10: Dizisel Veri Yolları

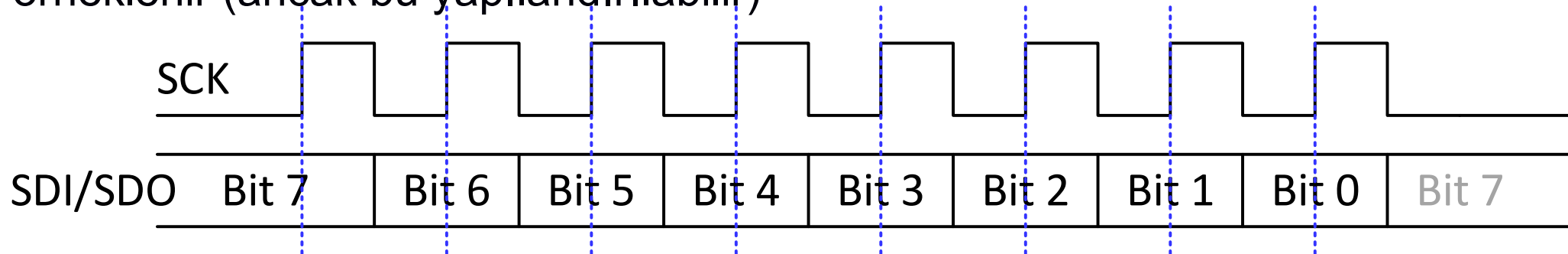


- **Denetleyici:** saat yollar, veri yollar & alır
- **Çevre Birimi:** saat alır, veri yollar & alır
- **Sinyaller:**
 - **SDO:** Dizisel Veri Dışarı
 - **SDI:** Dizisel Veri İçeri
 - **SCK:** SPI saati
 - **CSbar:** düşük sağlamalı yonga seçi

RVfpga Deney 10: Dizisel Veri Yolları



- **SCK boşta**
- Denetleyici, **SCK'de kenar** yolladığında denetleyici de çevre birimi de **veriyi örnekleyip yollar**. Veri, düşen kenarda değişip (yollanıp) yükselen kenarda örneklenir (ancak bu yapılandırılabilir)



RVfpga Deney 10: RVfpga'in SPI Modülü

- RVfpga'in SPI modülü OpenCores'dandır

https://opencores.org/projects/simple_spi

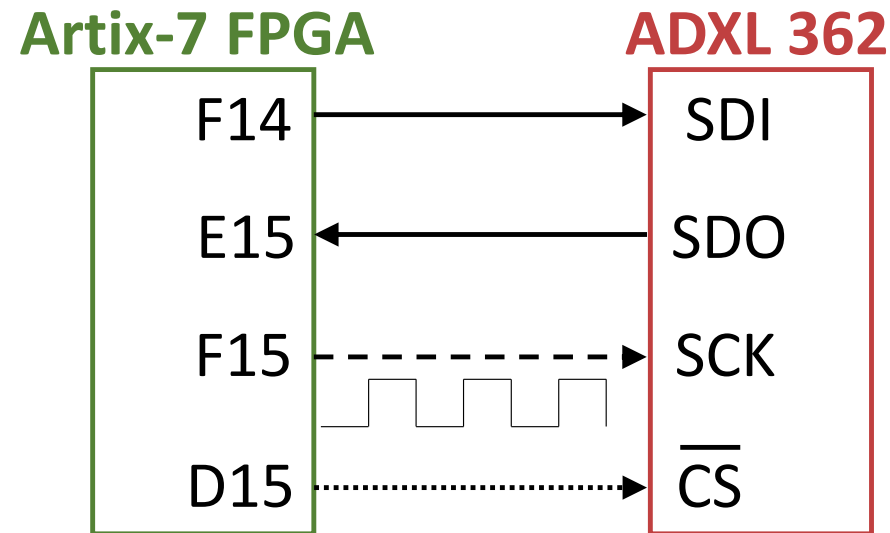
- 4-giriş okuma yazma arabellekleri
- SPI Yazmaçları:

Adı	Adresi	Geniřlięi	Eriřim	Tanım
SPCR	0x80001100	8	R/W	Denetleme yazmacı
SPSR	0x80001108	8	R/W	Durum yazmacı
SPDR	0x80001110	8	R/W	Veri yazmacı
SPER	0x80001118	8	R/W	Eklentiler yazmacı
SPCS	0x80001120	8	R/W	CS yazmacı

RVfpga Deney 10: ADXL362 ivmeölçer

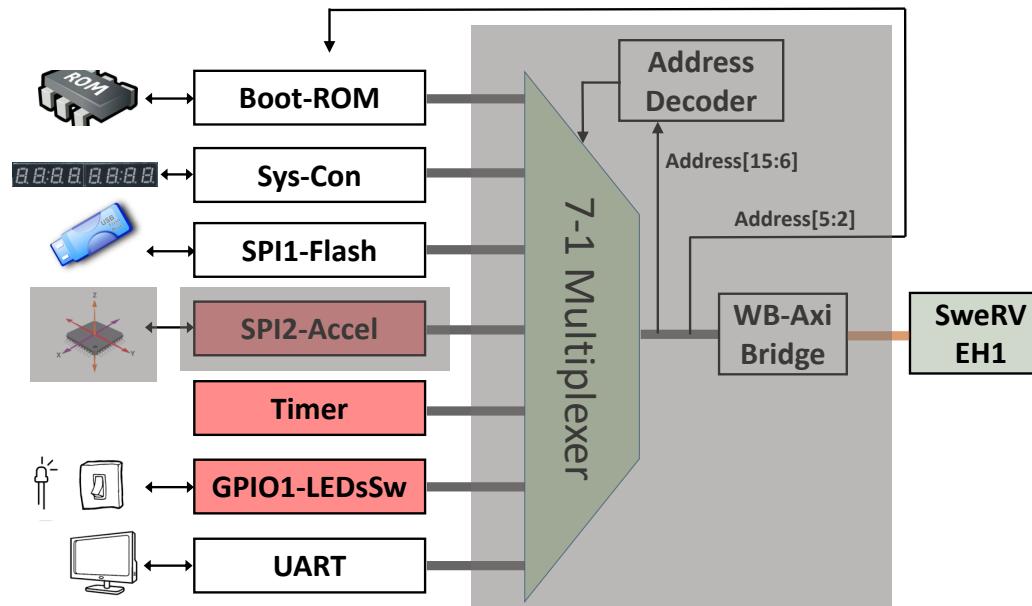
- Nexys A7 kartı bir Analog Devices ADXL362 ivmeölçer içerir. Bütün bilgilere şuradan erişebilirsiniz:

<https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL362.pdf>



RVfpga Deney 6: İvmeölçerin Alçak Düzeyli Gerçekleştirilmesi

- **3 ana parçaya bölünmüştür**
 - RVfpga'in karttaki ivmeölçerle dış bağlantısı (sol taralı bölge)
 - Yeni SPI modülünün RVfpga'e entegrasyonu (orta taralı bölge)
 - İvmeölçer ile SweRV EH1 arasında bağlantı (sağ taralı bölge)



RVfpga Deney 10: Dış Bağlantı

Dosya **rvfpga.xdc**: SoC'de kullanılan SPI sinyallerinin karşılık gelen karttaki ivmeölçer uçlarıyla bağlantısını tanımlar

```
78 ##Accelerometer
79 set_property -dict { PACKAGE_PIN E15   IOSTANDARD LVCMOS33 } [get_ports { i_accel_miso }]; #IO_L11P_T1_SRCC_15 Sch=acl_miso
80 set_property -dict { PACKAGE_PIN F14   IOSTANDARD LVCMOS33 } [get_ports { o_accel_mosi }]; #IO_L5N_T0_AD9N_15 Sch=acl_mosi
81 set_property -dict { PACKAGE_PIN F15   IOSTANDARD LVCMOS33 } [get_ports { accel_sclk }]; #IO_L14P_T2_SRCC_15 Sch=acl_sclk
82 set_property -dict { PACKAGE_PIN D15   IOSTANDARD LVCMOS33 } [get_ports { o_accel_cs_n }];
```

RVfpga Deney 10: RVfpga'ye Entegrasyon

Dosya **swervolf_core.v**: Üç-durumlu arabellekler ile GPIO modül somutlaması

```
simple_spi spi2
// Wishbone slave interface
.clk_i    (clk),
.rst_i    (wb_rst),
.adr_i    (wb_m2s_spi_accel_adr[2] ? 3'd0 : wb_m2s_spi_accel_adr[5:3]),
.dat_i    (wb_m2s_spi_accel_dat[7:0]),
.we_i    (wb_m2s_spi_accel_we),
.cyc_i    (wb_m2s_spi_accel_cyc),
.stb_i    (wb_m2s_spi_accel_stb),
.dat_o    (spi2_rdt),
.ack_o    (wb_s2m_spi_accel_ack),
.inta_o   (spi2_irq),
// SPI interface
.sck_o    (o_accel_sclk),
.ss_o     (o_accel_cs_n),
.mosi_o   (o_accel_mosi),
.miso_i   (i_accel_miso));
```