





THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga İlk Kullanım Kılavuzu

Sayılanlar



imagination
university programme



AUTHORS
Prof. Sarah Harris
Prof. Daniel [Chaver](#)

ADVISER
Prof. David Patterson

CONTRIBUTORS
Robert Owen
[Zubair Kakakhel](#)
[Olof Kindgren](#)
Prof. Luis [Piñuel](#)
[Ivan Kravets](#)
[Valerii Koval](#)
[Ted Marena](#)
Prof. Roy [Kravitz](#)

ASSOCIATES
Prof. Daniel León
Prof. José Ignacio Gómez
Prof. [Katzalin Olcoz](#)
Prof. Alberto del Barrio
Prof. Fernando Castro
Prof. Manuel Prieto
Prof. [Ataur Patwary](#)

[Prof. Christian Tenllado](#)
Prof. Francisco Tirado
Prof. Román Hermida
[Cathal McCabe](#)
Dan Hugo
[Braden Harwood](#)
[Prof. David Burnett](#)

[Gage Elerding](#)
[Prof. Brian Cruickshank](#)
[Deepen Parmar](#)
[Thong Doan](#)
Oliver Rew
[Niko Nikolay](#)
[Guanyang He](#)

Sponsors and Supporters



YAZARLAR

- Prof. Sarah Harris (<https://www.linkedin.com/in/sarah-harris-12720697/>)
- Prof. Daniel Chaver (<https://www.linkedin.com/in/daniel-chaver-a5056a156/>)

DANIŞMAN

- Prof. David Patterson (<https://www.linkedin.com/in/dave-patterson-408225/>)

KATKIDA BULUNANLAR

- Robert Owen (<https://www.linkedin.com/in/robert-owen-4335931/>)
- Zubair Kakakhel (<https://www.linkedin.com/in/zubairlk/>)
- Olof Kindgren (<https://www.linkedin.com/in/olofkindgren/>)
- Prof. Luis Piñuel (<https://www.linkedin.com/in/lpinuel/>)
- Ivan Kravets (<https://www.linkedin.com/in/ivankravets/>)
- Valerii Koval (<https://www.linkedin.com/in/valeros/>)
- Ted Marena (<https://www.linkedin.com/in/tedmarena/>)
- Prof. Roy Kravitz (<https://www.linkedin.com/in/roy-kravitz-4725963/>)

ORTAKLAR

- Prof. Daniel León (<https://www.linkedin.com/in/danileon-ufv/>)
- Prof. José Ignacio Gómez (<https://www.linkedin.com/in/jos%C3%A9-ignacio-gomez-182b981/>)
- Prof. Katzalin Olcoz (<https://www.linkedin.com/in/katzalin-olcoz-herrero-5724b0200/>)
- Prof. Alberto del Barrio (<https://www.linkedin.com/in/alberto-antonio-del-barrio-garc%C3%ADa-1a85586a/>)
- Prof. Fernando Castro (<https://www.linkedin.com/in/fernando-castro-5993103a/>)
- Prof. Manuel Prieto (<https://www.linkedin.com/in/manuel-prieto-matias-02470b8b/>)
- Prof. Christian Tenllado (<https://www.linkedin.com/in/christian-tenllado-31578659/>)
- Prof. Francisco Tirado (<https://www.linkedin.com/in/francisco-tirado-fern%C3%A1ndez-40a45570/>)
- Prof. Román Hermida (<https://www.linkedin.com/in/roman-hermida-correa-a4175645/>)
- Cathal McCabe (<https://www.linkedin.com/in/cathalmccabe/>)
- Dan Hugo (<https://www.linkedin.com/in/danhugo/>)
- Braden Harwood (<https://www.linkedin.com/in/braden-harwood/>)
- David Burnett (<https://www.linkedin.com/in/david-burnett-3b03778/>)
- Gage Elerding (<https://www.linkedin.com/in/gage-elerding-052b16106/>)
- Brian Cruickshank (<https://www.linkedin.com/in/bcruiksh/>)
- Deepen Parmar (<https://www.linkedin.com/in/deepen-parmar/>)
- Thong Doan (<https://www.linkedin.com/in/thong-doan/>)
- Oliver Rew (<https://www.linkedin.com/in/oliver-rew/>)
- Niko Nikolay (<https://www.linkedin.com/in/roy-kravitz-4725963/>)
- Guanyang He (<https://www.linkedin.com/in/guanyang-he-5775ba109/>)
- Prof. Ataur Patwary (<https://www.linkedin.com/in/ataurpatwary/>)

Bu içeriğin ilk çevirisini
Mehmet Oguz Derin
yapmıştır.

Görüş ile önerilerinizi
rvfpga@mehmetoguzderin.com
adresine iletebilir, çevirmene
@mehmetoguzderin
Twitter kullanıcı adından ulaşabilirsiniz.

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

İçindekiler

Sayılanlar	2
1. GİRİŞ	5
2. HIZLI BAŞLANGIÇ KILAVUZU	8
3. RISC-V MİMARİSİNİN TANITIMI	17
4. RVFPGA TANITIMI	19
5. YAZILIM ARAÇLARINI KURMA	38
6. RVFPGA'İ ÇALIŞTIRMA, PROGRAMLAMA	44
7. VERILATOR'DE SİMÜLASYON	74
8. WHISPER'DA SİMÜLASYON	80
9. EKLER	82



1. GİRİŞ

RISC-V FPGA, RVfpga olarak da yazılır, ticari bir RISC-V işlemcisini bir alanda programlanabilir geçit dizisine (FPGA) hedefleyip ardından bilgisayar mimarisi, dijital tasarım, gömülü sistemler, programlama üzerine öğrenmek için işlemciyi kullanıp genişletme için yönergeler, araçlar, deneyler içeren bir pakettir. RVfpga terimi, kendisini tanıttığımız kılavuz ile deneyler boyunca kullandığımız RISC-V yongadaki-sisteminin (SoC) de adıdır.

RVfpga İlk Kullanım Kılavuzunda kısaca şu bölümler vardır:

- **Hızlı Başlangıç Kılavuzu** (Bölüm 2)
- **Arkaplan, Tanıtım**
 - **RISC-V Mimarisi** (Bölüm 3)
 - **RVfpga** (Bölüm 4)
- **RVfpga’i Donanımda Kullanma**
 - **Yazılım Araçlarını Kurma** (Bölüm 5)
 - **RVfpga’i Çalıştırma, Programlama** (Bölüm 6)
- **RVfpga’in Simülasyonunu Yapma**
 - **Verilator Kullanarak, bir HDL Simülatörü** (Bölüm 7)
 - **Whisper Kullanarak, Western Digital’in Yönerge Kümesi Simülatörü** (Bölüm 8)
- **Ekler**
 - **Yerli RISC-V araç zinciri ile OpenOCD kullanma** (Ek A)
 - **PlatformIO’yu kullanmak için Windows’ta sürücülerini kurma** (Ek B)
 - **Windows’ta Verilator ile GTKWave’i kurma** (Ek C)
 - **MacOS’ta Verilator ile GTKWave’i kurma** (Ek D)
 - **RVfpga’i bir FPGA’e indirmek için Vivado’yu kullanma** (Ek E)
 - **Örnek: RVfpga’i bir endüstriyel IoT uygulamasında kullanma** (Ek F)

Hızlı Başlangıç Kılavuzu (Bölüm 2) RVfpga için gerekli minimal yazılım kurulumunu tanımlayıp ardından RVfpga’e yalın bir örnek programı nasıl kurup yürüteceğini gösterir. RVfpga’i daha bütünüyle anlamak için Bölüm 2’yi atlayıp Bölüm 3’te başlayan eksiksiz kılavuzla başla.

Bölüm 3, 4 RISC-V bilgisayar mimarisi ile RVfpga’e, bu kurs boyunca kullanacağın RISC-V yongadaki-sistem (SoC), kısa bir giriş yapar. RVfpga Western Digital’in (WD’nin) açık-kaynak RISC-V SweRV EH1 çekirdeğini kullanan SweRVolf SoC’sini kullanır. Bölüm 4 RVfpga ile RVfpga sistemini oluşturan Verilog dosyalarının düzenini tanımlar (Bölüm 4.D).

Kalan bölümler RVfpga’in donanım ile simülasyonda nasıl kullanılacağını gösterir. Bölüm 5 RVfpga’i kullanmak için gereken yazılımların nasıl kurulacağını gösterir. Bölüm 6 PlatformIO’nun RVfpga SoC’sini Nexys A7 FPGA kartına indirme (Bölüm 6.A) ile birkaç örnek programı RVfpga’e indirip çalıştırma (Bölüm 6.B-6.H) için nasıl kullanılacağını gösterir. Bölüm 7, 8 RVfpga’in Verilator (Bölüm 7), açık-kaynak HDL simülatörü, ile Whisper’da (Bölüm 8), Western Digital’in RISC-V Yönerge Kümesi Simülatörü (ISS), nasıl simülasyonunun yapılacağını gösterir.

Son olarak ekler RVfpga’in Linux komut istemcisinde nasıl kullanılacağını (Ek A), Windows ile MacOS makinelerde gerekli sürücülerle yazılımların nasıl indirileceğini (Ek B-D), Vivado’yu kullanarak RVfpga’in bir FPGA’e nasıl indirileceğini (Ek E) gösterir. Son ek, Ek F, RVfpga’in endüstri IoT uygulamasında nasıl kullanılacağını gösterir (Ek F).

Tablo 1 RVfpga için gerekli yazılım ile donanımı listeler. Bu kılavuz o araçlarla donanımın Ubuntu 18.04 işletim sisteminde (OS) nasıl indirilip kullanılacağını gösterir. Diğer işletim

sistemleri (Windows ile MacOS gibi) neredeyse birebir adımları izler. Yönlendirmeler ayrıldığına belirli yönergeleri **Windows** ile **macOS** parlatmasıyla yerleştiriyoruz.

Önemli: eğer Nexys A7 FPGA kartına erişimin yoksa deneyler Whisper, Western Digital'in yönerge kümesi simülatörü (ISS), ile Verilator, açık-kaynak HDL simülatörü, kullanarak simülasyonda bitirilebilir. Bu durumda Vivado'yu (Bölüm 5.A) kurman gerekmez; yalnızca VSCode/PlatformIO (Bölüm 2.A'da açıklandığı gibi) ile Verilator/GTKWave (Bölüm 5.C'de açıklandığı gibi) kurman gerekir.

Tablo 1. RVfpga için Gerekli Yazılım ile Donanım

Yazılım		
Adı	Websitesi	Ücreti
Vivado 2019.2 WebPACK	https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2019-2.html	ücretsiz
VSCode	https://code.visualstudio.com/Download	ücretsiz
PlatformIO	https://platformio.org/ Installed within VSCode	ücretsiz
Verilator (bir HDL simülatörü) ile GTKWave	https://github.com/verilator/verilator http://gtkwave.sourceforge.net/	ücretsiz
Whisper (Western Digital'in RISC-V Yönerge Kümesi Simülatörü)	https://github.com/chipsalliance/SweRV-ISS Installed within PlatformIO	ücretsiz
RISC-V Araç Zinciri ile OpenOCD	https://github.com/riscv/riscv-gnu-toolchain https://github.com/riscv/riscv-openocd Installed within PlatformIO	ücretsiz
Donanım		
Adı	Websitesi	Ücreti
Nexys A7 FPGA Kartı*	https://store.digilentinc.com/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/	\$265 (akademik ücret: \$199)
RISC-V Çekirdeği ile Yongadaki-Sistemi (SoC)**		
Adı	Websitesi	Ücreti
Western Digital'in SweRV EH1 Çekirdeği	https://github.com/chipsalliance/Cores-SweRV	free
SweRVolf	https://github.com/chipsalliance/Cores-SweRVolf	free

* All of the steps described in this guide also work on Digilent's Nexys4 DDR FPGA board.

** Provided with the RVfpga download from Imagination Technologies

ÖNEMLİ: Başlamadan önce Imagination'un University Programme'ından indirdiğin **RVfpga** klasörünü Ubuntu/Windows/macOS makinene kopyala. Bu RVfpga klasörünü koyduğun yerin mutlak yoluna [RVfpgaPath] diyeceğiz. RVfpga beş klasör barındırır: (1) **examples:** bu kılavuzu kullanırken çalıştıracağın örnek programlar, (2) **src:** RVfpga için kaynak kod (Verilog ile SystemVerilog), (3) **verilatorSIM:** SoC'nin simülasyonunu Verilator'de çalıştırmak

için senaryolar (script) barındırır, (4) **driversLinux_NexysA7**: Nexys A7 FPGA kartı için Linux sürücülerini barındırır, (5) **Labs**: RVfpga 1-10 arası Deneyler süresince kullanacağın programları barındırır.

Beklenen Ön Bilgi:

RVfpga kursunu, bu RVfpga İlk Başlangıç Kılavuzu ile RVfpga Deneylerini içeren, bitirmeden önce kullanıcıların şu konuları en azından temelden anlaması beklenir:

- Dijital mantık tasarımı
- Yüksek-düzeyle programlama (C önerilir)
- Çevirici programlama
- Yönerge kümesi mimarisi
- İşlemci mikromimarisi
- Bellek sistemleri

Bu konular *Digital Design and Computer Architecture: RISC-V Edition*, Harris & Harris, © Morgan Kaufmann, 2021 yazında yayınlanması beklenmektedir, ders kitabının kapsamındadır. *Computer Organization and Design RISC-V Edition*, Patterson & Hennessy, © Morgan Kaufmann 2017 gibi diğer ders kitapları bu konuların bir bölümünü kapsamına alır.

2. HIZLI BAŞLANGIÇ KILAVUZU

Bu bölüm RVfpga'i kullanmak için gereken minimal araçların nasıl kurulacağını gösterip ardından PlatformIO'nun RVfpga'i Nexys A7 FPGA kartına indirme, RVfpga'de program çalıştırma için nasıl kullanılacağını gösterir. FPGA kartını alman gerekecek (Tablo 1'e göz at). Bu adımlar kartın daha eski sürümü olan Nexys4-DDR FPGA kartı için de çalışır.

A. Minimal kurulum: VSCode, PlatformIO, Nexys A7 kart sürücüler

B. RVfpga'i FPGA'e indirip RVfpga'de program çalıştırma

Aşağıdaki yönergeler Ubuntu 18.04 sistemi içindir. Windows 10 ile MacOS işletim sistemleri için de çalışır – yönergeler Ubuntu'dan ayrıldığında **Windows** ile **macOS** özel yönergeleri içeren kutular yerleştiriyoruz. Ubuntu kullanıyorsan o kutuları görmezden gelebilirsin. Yollar ileriye eğik çizgilerle (/) Linux yolu olarak yazılı, Windows için yol genelde aynı ancak geriye eğik çizgildir (\).

A. Minimal kurulum: VSCode, PlatformIO, Nexys A7 kart sürücüler

Bu adımda RVfpga'i kullanmak için gereken minimum yazılımla sürücüyü kuracaksın. Öncelikle programlama ortamını kuracaksın, ardından Nexys A7 FPGA kartı için sürücüler kuracaksın.

VSCode, PlatformIO Kurulumu: RVfpga sistemini Nexys A7 kartına indirmek, RVfpga'e programlar indirip çalıştırmak için PlatformIO'yu, bir entegre geliştirme ortamı kullanacaksın. PlatformIO Microsoft'un Visual Studio Code'unun (VSCode) bir eklentisi olarak kurgulanmıştır. PlatformIO çapraz-platformdur, kendi-içerisinde bir ayıklayıcı da içerir.

VSCode ile PlatformIO'nun ikisini de kurmak için şu adımları izle:

1. VSCode'u kur:

a. Kurulum dosyasını şu bağlantıdan indir:

<https://code.visualstudio.com/Download>

b. Bir terminal açıp VSCode'u kurup yürüt:

```
cd ~/Downloads
sudo dpkg -i code*.deb
code
```


Windows / macOS: VSCode paketleri burada Windows (.exe dosyası) ile macOS (.zip dosyası) için de vardır <https://code.visualstudio.com/Download>. Bu işletim sistemlerinde normal programlardaki gibi adımları izleyerek kurup yürüt.

2. PlatformIO'yu VSCode üzerine kur:

a. python3 yardımcılarını şunu terminale yazarak kur:

```
sudo apt install -y python3-distutils python3-venv
```

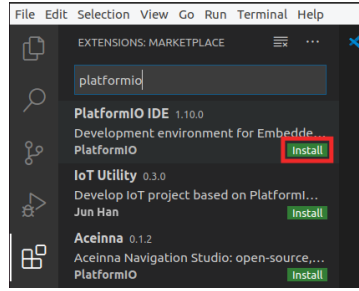
Windows / macOS: bu adım (2.a) Windows'ta gerekli değildir. MacOS'te python3'ü kurmak için *homebrew*'ü kullanabilirsin: `brew install python3`

- b. Daha açık değilse Başlat düğmesini seçip arama menüsünde “VSCode” yazıp, VSCode’u seçip ya da bir Ubuntu terminalinde `code` yazıp VSCode’u başlat.
- c. VSCode’un solundaki Eklentiler ikonuna tıkla  (Figür 1’e göz at).



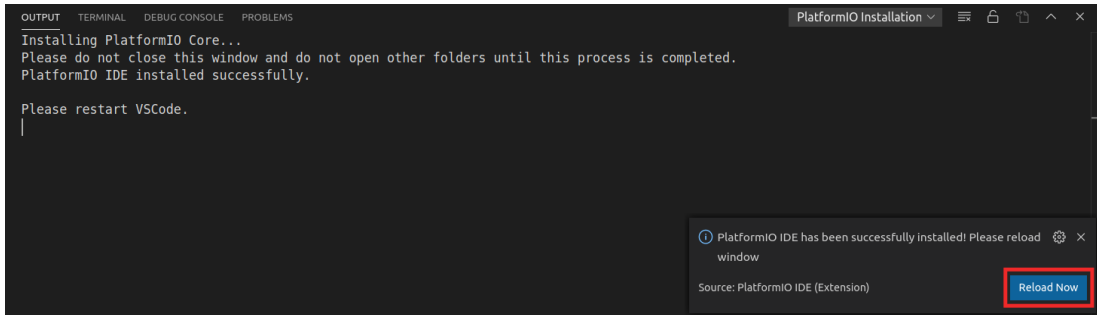
Figür 1. VSCode’un Eklentiler ikonu

- d. Arama kutusuna *PlatformIO* yaz, PlatformIO IDE’sini yanındaki kur butonuna tıklayarak kur (Figür 2’ye göz at).



Figür 2. PlatformIO IDE Eklentisi

- e. Aşağıdaki OUTPUT (ÇIKTI) penceresi seni kurulum süreciyle ilgili bilgilendirecek. Bitince pencerenin sağ aşağısındaki “Reload Now” (Şimdi Yenile) butonuna tıkla, böylelikle PlatformIO’nun VSCode içerisindeki kurulumu bitecek (Figür 3’e göz at).



Figür 3. PlatformIO kurulduktan sonra Reload Now (Şimdi Yenile)

Nexys A7 kablo sürücülerini kurulumu: Nexys A7 kartı için sürücülerini manüel kurman gerekir.

- Bir terminal aç.
- `[RVfpgaPath]/RVfpga/driversLinux_NexysA7` dizinine git. (Kolaylık için bu sürücülerini RVfpga klasöründe sağlıyoruz. Bu kılavuzun Bölüm 5’inde Vivado’yu kurunca bölümde açıklandığı gibi o sürücülerini de indirdiğin pakette bulabilirsin.)

- Kurulum senaryosunu (script) çalıştır:

```
chmod 777 *
sudo ./install_drivers
```
- Değişikliklerin etkimesi için Nexys A7 kartının kablosunu bilgisayarıdan çıkarıp bilgisayarı yeniden başlat.

Windows: sürücülerini Nexys A7 kartına kurmak için Ek B'deki yönergeleri izle.

macOS: ek sürücü kurmak gerekmez.

B. RVfpga'i FPGA'e indirip RVfpga'de programlar çalıştır

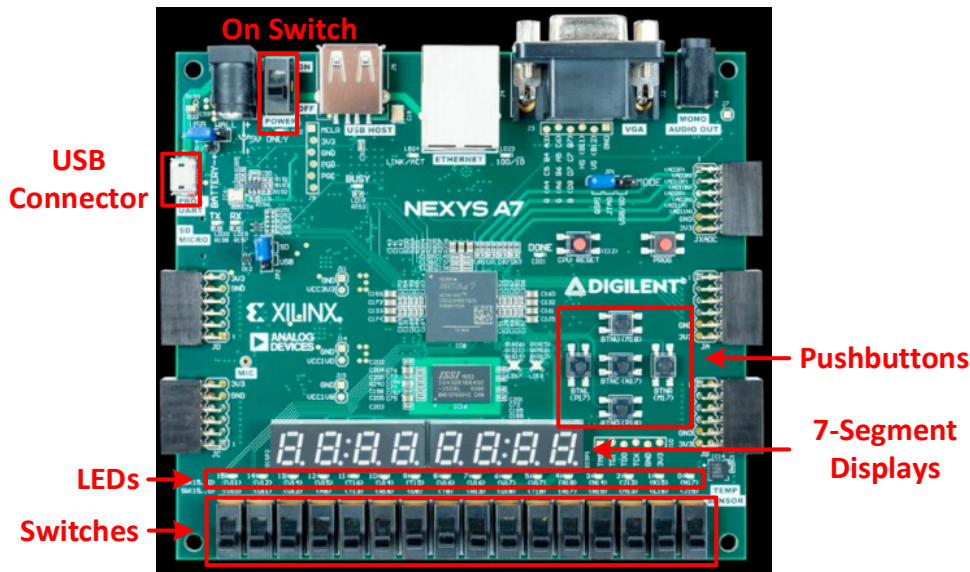
Şimdi RVfpga'i, FPGA'e hedeflenen RISC-V sistemi, Nexys A7 FPGA kartına indireceksin. Bu İlk Kullanım Kılavuzunda değiştirmeyecek olsak da RVfpga'in Verilog'u *[RVfpgaPath]/RVfpga/src* içerisinde erişilebilir. Bu GSG'nin Bölüm 4'ünde RVfpga'in kaynak kodunu tanımlayacağız, RVfpga 6-10 arası Deneylerde daha detaylandıracağız. RVfpga'i o deneylerin kimi alıştırma ve uygulamalarında da değiştireceksin.

RVfpga'i Nexys A7 FPGA kartına şu adımları bitirerek indir:

- Adım 1.** Nexys A7 FPGA kartını bilgisayara bağlayıp kartı aç
- Adım 2.** PlatformIO ile C programını aç
- Adım 3.** RVfpga'i Nexys A7 kartına indir
- Adım 4.** Programı RVfpga'e indirip çalıştır

Adım 1. Nexys A7 FPGA kartını bilgisayara bağlayıp kartı aç

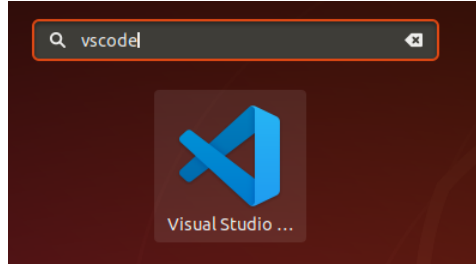
Verilen USB kablosunu kullanarak Nexys A7 kartını bilgisayarına bağla. Figür 4 Nexys A7 FPGA kartındaki LEDlerin, anahtarların, USB bağlayıcısının, açma anahtarının, düğmelerin, 7-kesimli ekranların fiziksel yerini gösterir. USB bağlayıcı portu ile Nexys A7 kartı arasına bir kablo bağlayıp kartı aç.




Figür 4. Digilent's Nexys A7 FPGA board's I/O interfaces
 (figure of board from <https://reference.digilentinc.com/>)

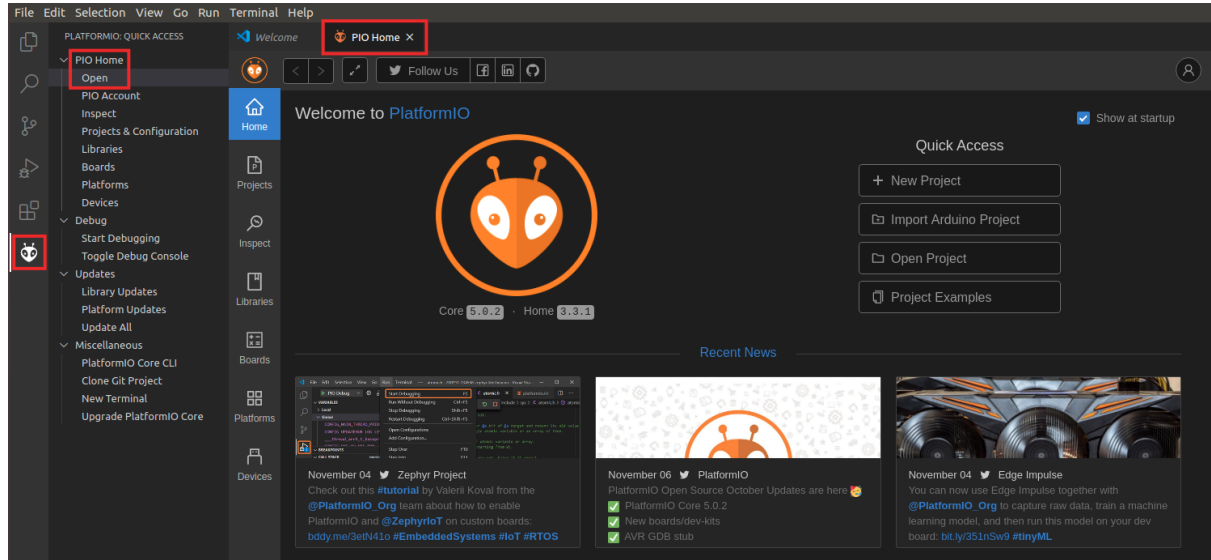
Adım 2. PlatformIO ile C programını aç

Şimdi Başlat Menüsünde VSCode yazarak (Figür 5'e göz at) ya da bir terminalde `code` yazarak Visual Studio Code'u (VSCode) aç.



Figür 5. VSCode'u aç

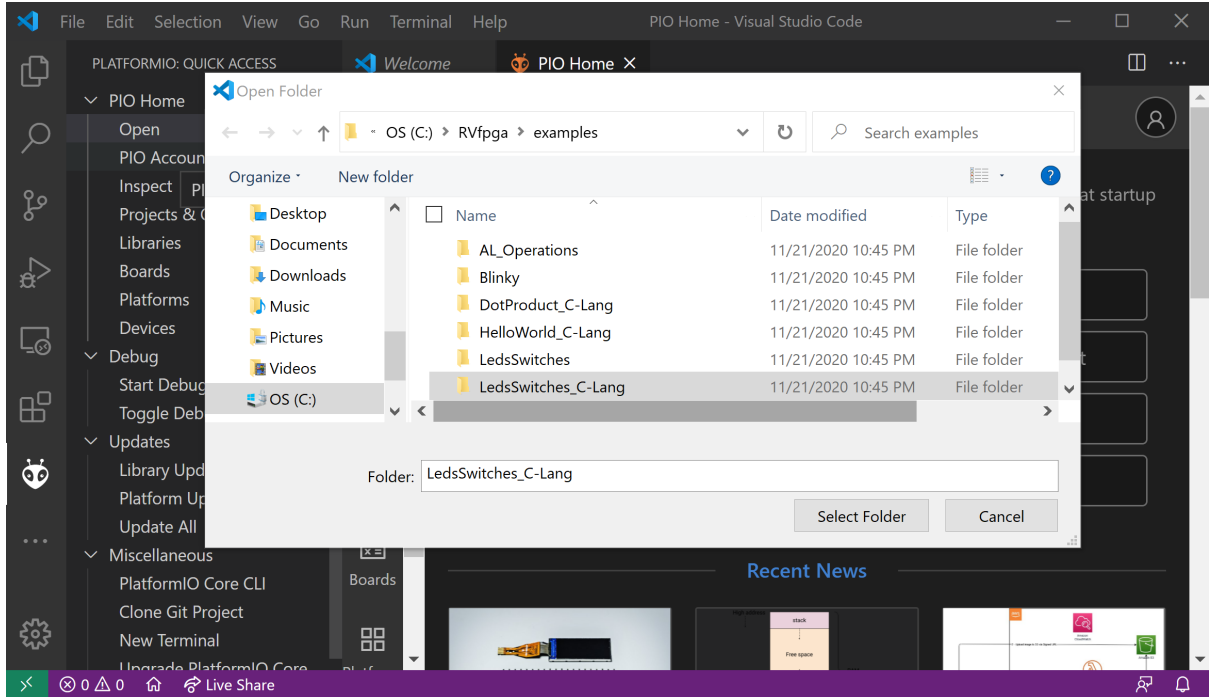
Eğer PlatformIO Home (PIO Home) penceresi kendiliğinden açılmazsa sol şerit menüdeki PlatformIO ikonuna tıkla . Ardından PIO Home'u genişletip Aç üzerine tıkla. Şimdi PIO Home Welcome penceresine açılacaktır (Figür 6'ya göz at).



Figür 6. PIO Home'u aç

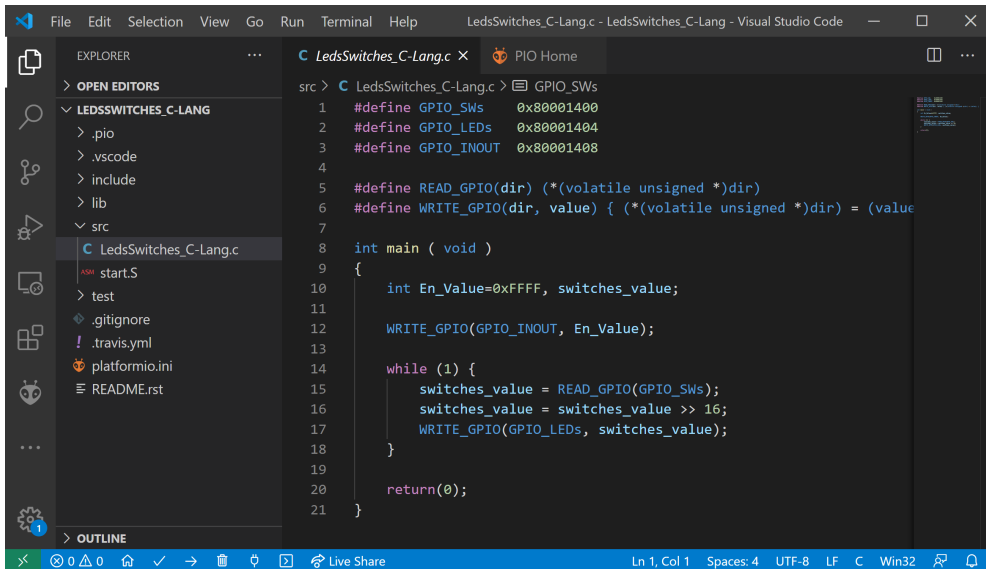
Şimdi üst dosya menüsünden Dosya (File) → Klasörü Aç'a (Open Folder) tıklayıp şunu seç:
[RVfpgaPath]\RVfpga\examples\LedsSwitches_C-Lang

Klasörü seç ancak açma (Figür 7'ye göz at). PlatformIO şimdi Nexys A7 kartındaki anahtarların değerini okuyup karttaki LEDlere yazan LedsSwitches_C-Lang programını açacaktır.



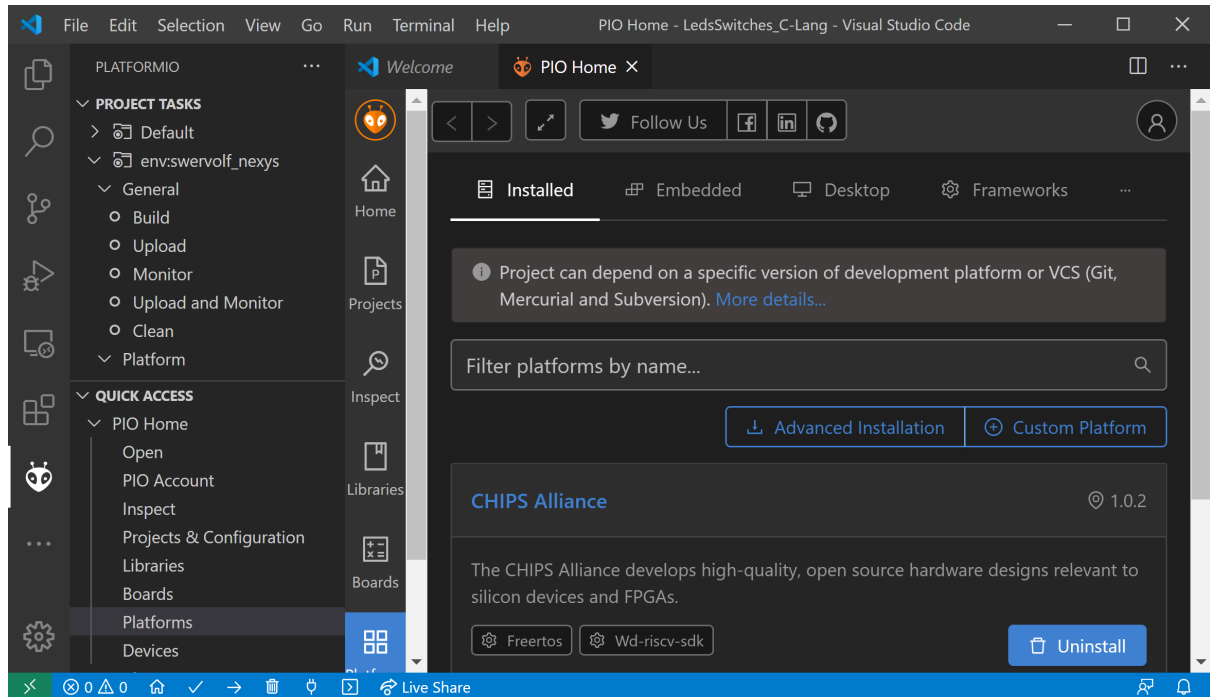
Figür 7. LedsSwitches_C-Lang örneğini aç

LedsSwitches_C-Lang programını *src* klasörünü genişletip LedsSwitches_C-Lang.c'ye çift tıklayarak açabilirsin (Figür 8). Bu programı bu İlk Kullanım Kılavuzunda sonra daha detaylı tartışacağız. Bu Hızlı Başlangıç Kılavuzu için, Nexys A7 kartında çalışacak olan, bu programı RVfpga'ya yalnızca indireceğiz.




Figür 8. LedsSwitches_C-Lang.c programı

Önemli olarak bir RVfpga örneği PlatformIO'da ilk kez açıldığında Chips Alliance platformu kendiliğinden kurulur (PIO Home → Platforms içerisinde görebilirsin, Figür 9'da gösterildiği gibi). Bu platform ileride kullanacağın önceden-kurgulanmış RISC-V araç zinciri (toolchain), RISC-V için OpenOCD, RVfpga veri dosyası (bitfile) ile RVfpgaSIM, JavaScript ile Python senaryoları, örnekler gibi sonra kullanacağın araçlar içerir. Olur da Chips Alliance platformu kendiliğinden kurulmazsa manüel olarak da kurabilirsin, Bölüm 6.A'da açıklandığı gibi.



Figür 9. PlatformIO'da kurulu Chips Alliance platformu

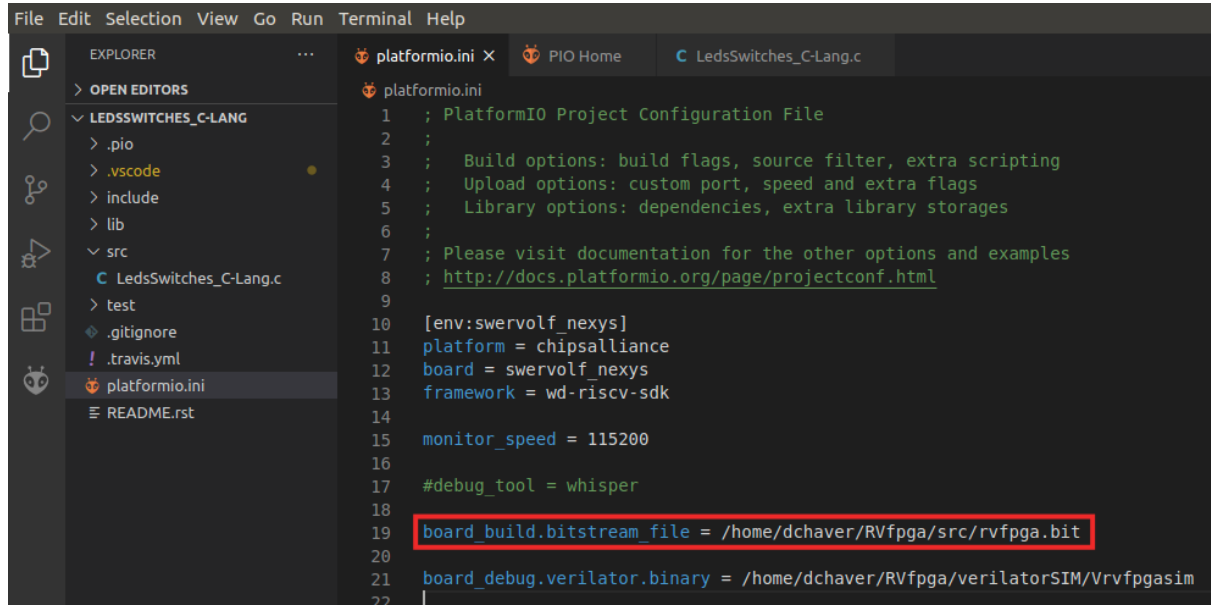
Adım 3. RVfpga'i Nexys A7 kartına indir

Şimdi, bir çevre birimlerine destekli RISC-V işlemcisi içeren RISC-V SoC'si olan, RVfpga'i indirebilirsin. GEZGİN (EXPLORER) penceresinde üzerine çift-tıklayarak platformio.ini (PlatformIO ilk değerlendirme dosyası) dosyasını aç, Figür 10'da gösterildiği gibi. (Eğer Gezgin penceresi açıkta değilse sol şerit menüde  üzerine tıklayarak açabilirsin.) Şimdi RVfpga'i tanımlayan veri dosyasının konumunun yolunu board_build.bitstream_file yolunu kendi yolunla değiştirerek ekle (Figür 10'a göz at):

board_build.bitstream_file = [RVfpgaPath]/RVfpga/src/rvfpga.bit


platformio.ini dosyasını Ctrl-s'e basarak kaydet.

Proje Yapılandırma Dosyası (*platformio.ini*) için çok türlü komut vardır; bu seçenekler üzerine daha çok bilgiye şuradan erişilebilir: <https://docs.platformio.org/en/latest/projectconf/>.




Figür 10. RVfpga veri dosyasına giden yolu ekle

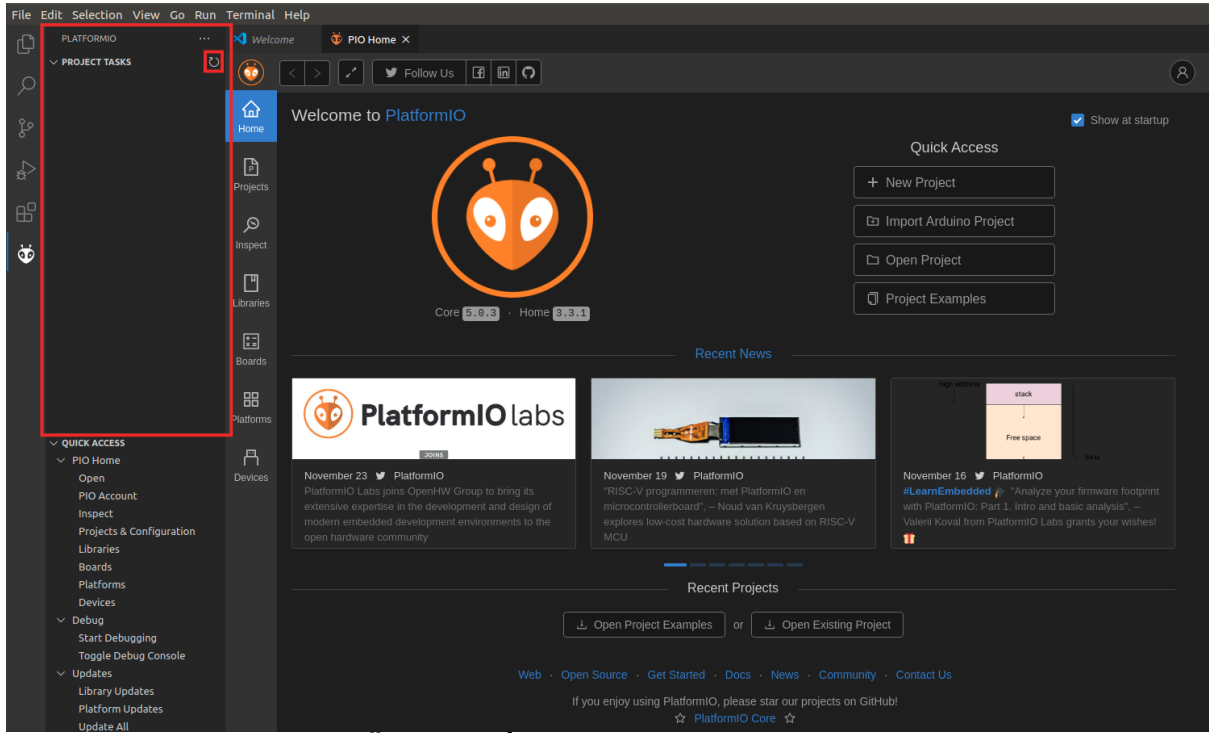
RVfpga'ı (bu veri dosyasında tanımlandığı gibi) Nexys A7 kartına indir:

- Sol menü şeridindeki PlatformIO ikonuna  tıkla (Figür 11'e göz at).



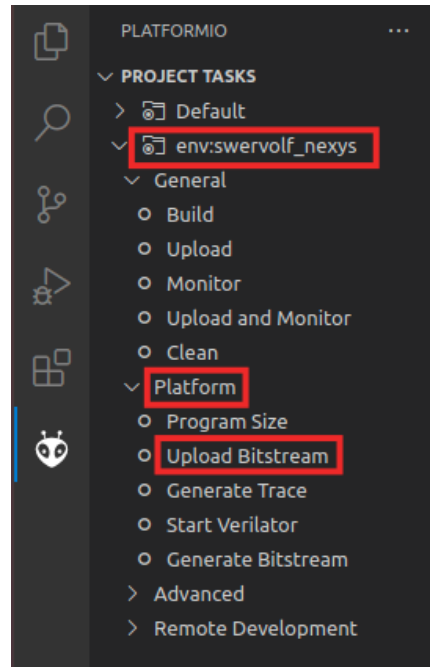
Figür 11. PlatformIO ikonu

- Proje Görevleri (Project Tasks) penceresi boşsa (Figür 12),  üzerine tıklayarak Proje Görevlerini yenilemen gerek. Bu birkaç dakika alabilir.



Figür 12. PROJE GÖREVLERİ (PROJECT TASKS) penceresi boş – Yenile


- Ardından Proje Görevleri (Project Tasks) → env:swervolf_nexys → Platform'u genişletip Veri Akışını Yükle'ye (Upload Bitstream) tıkla, Figür 13'te gösterildiği gibi. **Bir ya da iki saniye sonra FPGA, RVfpga SoC'si ile programlanmış olacak** (karttaki 7-Kesimli Ekranlar 8 tane sıfır göstermeli).

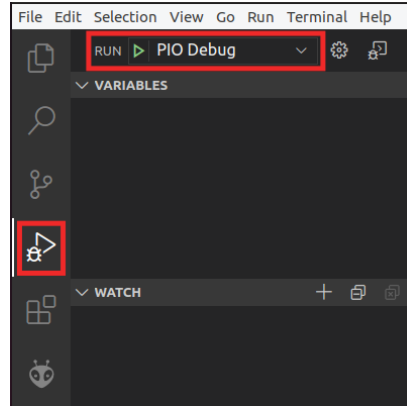


Figür 13. Veri Akışını (Bitstream) Yükle

Adım 4. Programı RVfpga'e indirip çalıştır

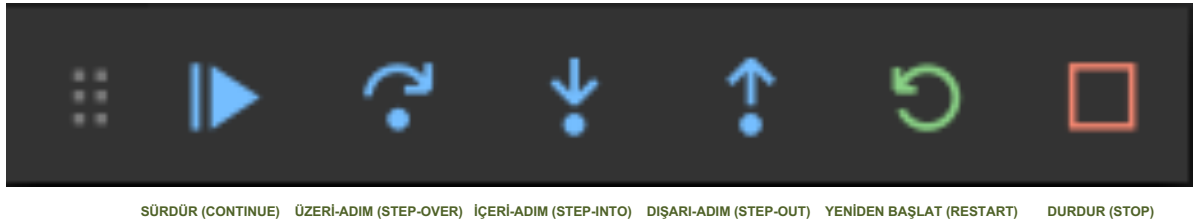
RVfpga Nexys A7 kartına inip çalıştığına göre şimdi programı RVfpga'in belleğine indirip programı çalıştırıp/ayıklayacaksın. Sol yan çubuktaki "Çalıştırıp Ayıkla (Run and Debug)"

butonuna tıkla: . Oynat (Play) butonuna basarak ayıklayıcıyı başlat ("PIO Ayıkla (PIO Debug)" seçeneğinin seçili olduğunun sağlamasını yap). Bu butonu pencerenin üstünün yakınlarında bulabilirsin (Figür 14'e göz at). Program ilk derlenir, ardından ayıklama başlar.




Figür 14. Programı derleyip indir ardından ayıklayıcıyı başlat

Ayıklama oturumunu denetlemek için editörün üstünün yakınında beliren ayıklama araç çubuğunu kullanabilirsin (Figür 15'e göz at). Bu İlk Kullanım Kılavuzunda sonra bütün seçenekleri tanımlayıp deneyeceğiz.



Figür 15. Ayıklama (Debugging) araçları

PlatformIO main işlevinin başlangıcına geçici bir kesme noktası (breakpoint) ayarlar. Yani programı çalıştırmak için Sürdür (Continue) butonuna  bas. Şimdi Nexys A7 FPGA kartındaki anahtarların durumunu değiştirip karşılık gelen LEDlerin ışığının açılmasını gör.

3. RISC-V MİMARİSİNİN TANITIMI

RISC-V 2011'de University of California, Berkeley'deki Par Lab'de oluşturulmuş bir Yönerge Kümesi Mimarisi'dir (Instruction Set Architecture) (ISA). RISC-V için amaç küçük, kısıtlı, düşük-kaynaklı IoT aygıtlarından süperbilgisayarlara bütün uygulamalarda kullanılan işlemciler için bir "Evrensel ISA" olmaktır. RISC-V mimarları bu amaca ulaşmak için mimariye beş prensip belirledi:

- Yazılım paketleriyle programlama dillerinin geniş bir aralığıyla uyumlu olmalı.
- Gerçekleştirilmesi FPGAlerden ASIClere (uygulamaya özel entegre devreler, application specific integrated circuits), yeni gelen teknolojiler de dahil, bütün teknoloji seçeneklerinde uygulanabilir olmalı.
- Mikro kod ya da fiziksel bağlantılı (hardwired) denetim, düzenli (in-order) ya da düzensiz (out-of-order) boruhattı (pipeline), paralelliğin değişik türlerini, gibi gibi gerçekleştiren değişik mikromimari senaryolarında verimli olmalı.
- Belirli görevlere gereken maksimum performansı ISA'nın kendisi ayakbağı olmadan uygunlaştırılabilir.
- Taban yönerge kümesi, geliştiricilere yaygın, sağlam bir çerçeve (framework) sunarak, stabil, dayanıklı olmalı.

RISC-V bir açık standarttır, üzerine standart kamu malı olup 2015'ten beri şimdi adı **RISC-V International**, RISC-V mimarileri için donanım ile yazılım geliştirilmesinin özendirilmesini yapan kâr amacı gütmeyen kuruluş, olan **RISC-V Foundation** eliyle yönetilmiştir. 2018'de RISC-V Foundation, Linux Foundation ile birlikte çalışmaya başlayıp Mart 2020'de İsviçre ana merkezli RISC-V International oldu. Bu geçiş standardın gelecekteki açıklılığıyla ilgili kaygıları yok etti. 2020'den beri RISC-V International, araştırmadan, akademiden, endüstriden Microchip, NXP, Samsung, Qualcomm, Micron, Google, Alibaba, Hitachi, Nvidia, Huawei, Western Digital, ETH Zurich, KU Leuven, UNLV, UCM gibi adları içeren 200'den çok anahtar oyuncunun desteğini almıştır.

RISC-V, açık bir standart olup artımlı yerine modüler oluşundan dolayı geçtiğimiz 10-20 yılda küresel olarak önemli birkaç, yüksek olasılıkla tek, ISA'dır. Modülerliği onu esnek, güzel yapıyor. İşlemciler taban ISA ile yalnızca kullanılan eklentileri gerçekleştirir. Bu modüler yaklaşım x86 ya da ARM gibi artımlı mimari, yani önce ISAların genişletilip bütün işlemcilerin eski yazılım programlarıyla geri uyumluluk için "kullanım dışı" olarak etiketlenmiş yönergeleri bile gerçekleştirmesinin gerektiği, kullanan geleneksel ISAlardan ayrılır. Örneğin 80 yönerge ile başlayan x86'in bugün 1300'den, makine kodundaki bütün opcodesı sayarsan 3600'den, çok yönergesi vardır. Yönergelerin böyle çok olup, geri uyumluluğun zorunlu olması büyük, çok-güç-tüketen, kısa opcodesların, ya da kısa yönergelerin, hepsi kullanımda olduğu için, uzun yönergeleri desteklemesi gereken işlemcilerle sonuçlanıyor.

RISC-V'nin dört taban ISA seçeneği vardır: iki 32-bit sürüm (tam sayı ile gömülü sürümleri, RV32I ile RV32E), 64- ile 128-bit sürümleri (RV64I ile RV128I), Tablo 2'de gösterildiği gibi. Onaylı (Ratified) olarak işaretlenmiş ISA modülleri bugün onaylanmıştır. Donuk (Frozen) olarak işaretlenmiş modüller onaylanmadan önce çok değişmemesi beklenir. Taslak (Draft) olarak işaretlenmiş modüllerin onaylanmadan önce değişmesi beklenir. Küçük işlemciler kurgulayabilmek maliyet-, alan-, enerji- kısıtlı aygıtlar için özellikle önemli bir gerekliliktir. Yönerge eklentileri belirli görevlerin yapılabilmesi için bu taban ISAların üzerine eklenebilir, örneğin kayan noktalı sayı işlemleri, çarpma ile bölme, vektör işlemleri. Bu özelleşmiş donanım eklentileri ayrıca standart içinde olup derleyicilerin bilgisindedir, dolayısıyla istenen seçeneklerin derleyicide etkinleştirilmesi hedeflenmiş ikili kod oluşturulmasını sağlayacaktır. Eklentiler bir harf ile tanınıp gerçekleştirmenin donanım kabiliyetlerini göstermek için çekirdek ISAe eklenmelidir, Tablo 3'teki gibi. Örneğin, RVM çarpma/bölme eklentisidir, RVF kayan-nokta eklentisidir, gibi gibi.

Tablo 2. RISC-V base (taban) ISAları(tablo şuradan <https://riscv.org/technical/specifications/>)

Base	Version	Status
RVWMO	2.0	Ratified
RV32I	2.1	Ratified
RV64I	2.1	Ratified
<i>RV32E</i>	<i>1.9</i>	<i>Draft</i>
<i>RV128I</i>	<i>1.7</i>	<i>Draft</i>

Tablo 3. RISC-V standart ISA eklentileri(tablo şuradan <https://riscv.org/technical/specifications/>)

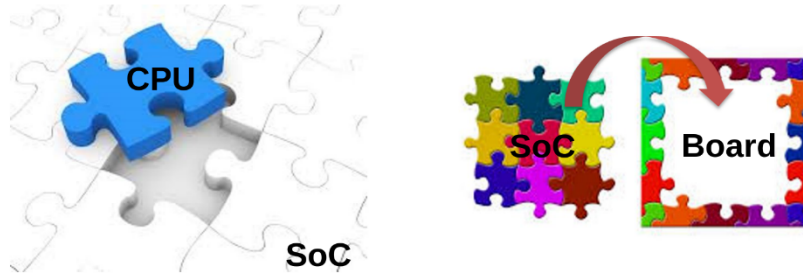
Extension	Version	Status
Zifencei	2.0	Ratified
Zicsr	2.0	Ratified
M	2.0	Ratified
<i>A</i>	<i>2.0</i>	<i>Frozen</i>
F	2.2	Ratified
D	2.2	Ratified
Q	2.2	Ratified
C	2.0	Ratified
<i>Ztso</i>	<i>0.1</i>	<i>Frozen</i>
<i>Counters</i>	<i>2.0</i>	<i>Draft</i>
<i>L</i>	<i>0.0</i>	<i>Draft</i>
<i>B</i>	<i>0.0</i>	<i>Draft</i>
<i>J</i>	<i>0.0</i>	<i>Draft</i>
<i>T</i>	<i>0.0</i>	<i>Draft</i>
<i>P</i>	<i>0.2</i>	<i>Draft</i>
<i>V</i>	<i>0.7</i>	<i>Draft</i>
<i>N</i>	<i>1.1</i>	<i>Draft</i>
<i>Zam</i>	<i>0.1</i>	<i>Draft</i>

G harfi, “genel” anlamına gelir, MAFD eklentilerinin içerildiğini belirtmek için kullanılır. Önemli olarak bir şirket ya da şahıs standart modüllerde kullanılmayacağı garantilenen opcodeları kullanarak kendisine özel eklentiler geliştirebilir. Bu üçüncü-parti gerçekleştirmelerin daha hızlı pazara-sürüm-süresiyle geliştirilmesini sağlar.

Örneğin, bir 64-bit RISC-V gerçekleştirmesi, bütün dört genel ISA eklentilerini içerip üzerine *Bit Manipulation* (Bit Manipülasyonu) ile *User Level Interrupts* (Kullanıcı Düzeyli Kesintiler) eklentilerini içeren, RV64GBN ISA olarak adlandırılır. Bütün bu modüller ayrıcalıksız (unprivileged) ya da kullanıcı spesifikasyonunun kapsamındadır. RISC-V vakfı ayrıca genel amaçlı işletim sistemlerini çalıştırma için gereken ayrıcalıklı (privileged) işlemlerin gereklilikleriyle yönergelerinin bir kümesini de kapsar.

4. RVFPGA TANITIMI

Bu bölümde RVfpga sistemini çekirdekten FPGA kart arayüzüne bütünüyle tanımlıyoruz. Figür 16 normal bir gömülü sistemin hiyerarşik düzenini işlemci çekirdeğinden başlayıp, çekirdek çevresinde kurgulanmış SoC'ye geçip, son olarak sistem ile kart arayüzünü görselleştiriyor. İşlemci çekirdeğini, RISC-V yönergelerini yürütür, tanımlayarak başlıyoruz (**Western Digital'in SweRV EH1 Çekirdeği (Core)**); ardından, Bölüm B'de, sistemin donanım bileşenlerini (çekirdek, bellek, girdi/çıkışı) entegre eden **SweRVolf SoC'si** ile RVfpga'de kullanmak için eklenen eklentileri tanımlıyoruz; Bölüm C'de Nexys A7 FPGA kartında gerçekleştirilmiş olan SweRVolf SoC'sini (**RVfpga**) tanımlayıp (**RVfpga**) simülasyonda kullanılan SweRVolf SoC'sini (**RVfpgaSIM**) tanımlıyoruz. Son olarak Bölüm D'de RVfpga sisteminin bütün dosya yapısını açıklıyoruz.



Figür 16. Gömülü Sistem (Embedded System) düzeni

A. SweRV EH1 Core ile SweRV EH1 Core Complex

Western Digital geçtiğimiz yıllarda üç RISC-V çekirdeği geliştirdi: **SweRV EH1** (RVfpga'de kullanılan çekirdek), **SweRV EH2**, **SweRV EL2** (RVfpga'in gelecek sürümleri bu çekirdekleri içerebilir). Bütün çekirdeklerin Apache 2.0 lisansı var. SweRV EH1 Çekirdeği bir 32-bit, 2-yönlü (2-way) süperskaler, 9-aşamalı (9-stage) boruhattı (pipeline) çekirdeğidir. SweRV Core EH2 çekirdeği üzerine kurgulanıp EH1 Çekirdeğini ek performans için çift iş parçacığı kapasitesiyle genişletir. SweRV Çekirdeği EL2 daha tutumlu performanslı, daha küçük bir çekirdektir. Şu RISC-V sayfası <https://www.westerndigital.com/company/innovations/risc-v> üç erişilebilir çekirdeğin üstünden geçer, Tablo 4'de ana özellikleri verildiği gibi.

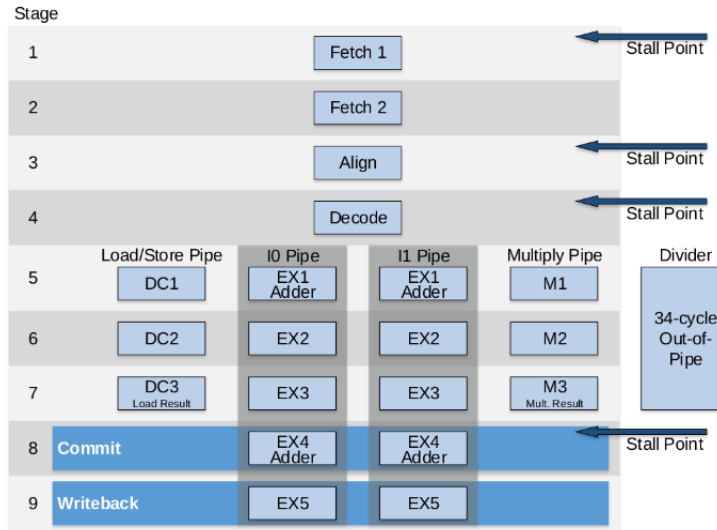
Tablo 4. Üç WD RISC-V Çekirdeğinin ana özellikleri

(tablo şuradan <https://www.westerndigital.com/company/innovations/risc-v>)

Core Name	RISC-V Type	Pipeline Stages	Threads	Size @ TSMC	CoreMarks/Mhz
SweRV Core EH1	RV32IMC	9- dual issue	Single	.11mm @ 28nm	4.9
SweRV Core EH2	RV32IMC	9- dual issue	Dual	.067 @ 16nm	6.3
SweRV Core EL2	RV32IMC	4- single issue	Single	.023 @ 16nm	3.6

Bu üç çekirdek arasından **SweRV EH1 Çekirdeği** (RVfpga paketiyle sağlanmıştır, şuradan da erişilebilir <https://github.com/chipsalliance/Cores-SweRV>) yüksek performansı/MHz ile basit iş parçacığı yapısı için seçilmiştir. Dahası, Chips Alliance, açık-kaynak donanımlar sağlamaya kendini adanmış bir grup, eksiksiz, onaylanmış SweRVolf adında bir SoC (RVfpga paketiyle sağlanmıştır, şuradan da erişilebilir <https://github.com/chipsalliance/Cores-SweRVolf>), sağlar. RVfpga, SweRVolf SoC'nin Western Digital'in **SweRV EH1 Çekirdeği** sürüm **1.6** kullanan bir eklentisini kullanır.

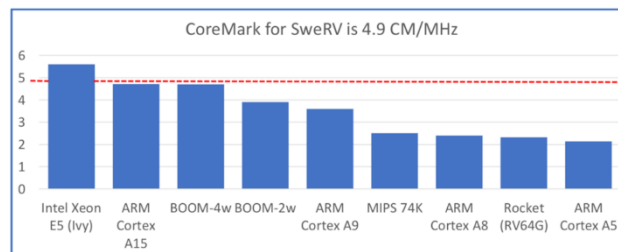
SweRV EH1 Çekirdeği makine-moduna (M-modu) özel, RISC-V'in tam sayı (I), sıkıştırılmış yönerge (C), tam sayı çarpma bölme (M) eklentilerini destekleyen bir 32-bit CPU çekirdeğidir. Programlamanın Referans Manüeli (https://github.com/chipsalliance/Cores-SweRV/blob/master/docs/RISC-V_SweRV_EH1_PRM.pdf) çekirdeğin bütün özelliklerini, yapısından zamanlama bilgisiyle bellek eşlemelerine, detayıyla tanımlar. SweRV EH1 çift-yayınlı 9-aşamalı, dört aritmetik mantık ünitesini (ALUlar), EX1'den EX4'e etiketlenmiş iki, I0, I1, boruhattında destekleyen boruhattı (Figür 17'ye göz at) içeren bir süperskaler çekirdektir. Boruhattının iki yönü de ALU işlemlerini destekler. Boruhattının bir yönü yüklemeyi/depolamayı desteklerken öteki yönde 3-dönüm gecikme çarpanı vardır. İşlemcide ayrıca bir adet boruhattı-dışı (out-of-pipeline) 34-dönüm gecikme bölene vardır. Boruhattında dört bekleme noktası vardır: 'Fetch 1', 'Align', 'Decode', 'Commit'. 'Fetch 1' aşaması bir Gshare dallanma kestiricisi içerir. 'Align' aşamasında yönergeler üç getirme arabelleğinden (fetch buffer) alınır. 'Decode' aşamasında dört yönerge arabelleğinden en çok iki yönerge çözülür. 'Commit' aşamasında dönüm başına en çok iki yönerge uygulanır. Son olarak, 'Writeback' aşamasında mimari yazmaçlar güncellenir.



Figür 17. SweRV EH1 çekirdek mikromimarisi

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

Figür 18 değişik çekirdeklerle işlemcilerin karşılaştırmasını gösterir. SweRV EH1 Çekirdeğinin MHz başına performansı 4.9 CM/MHz (CoreMark per MHz) etkileyici yüksekliğindedir: ARM Cortex A8'den iki kat hızlı olup ARM Cortex A15 performansını bile aşmaktadır.

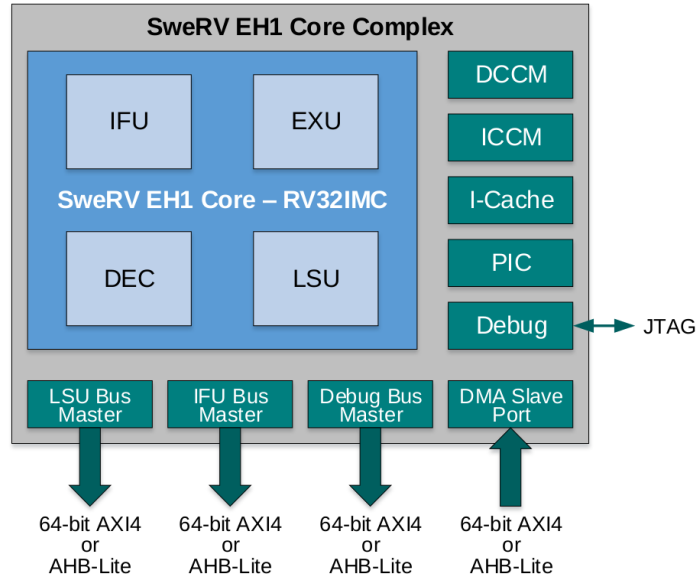


Figür 18. İş parçacığı ile MHz başına değerlendirme karşılaştırması

(figür şuradan https://content.riscv.org/wp-content/uploads/2019/12/12.11-14.20a3-Bandic-WD_SweRV_Cores_Roadmap_v4SCR.pdf)

Western Digital ayrıca SweRV EH1 Çekirdeğine **SweRV EH1 Core Complex** (Figür 19'a göz at), adında yukarıda tanımlanıp figürde mavi renklendirilmiş EH1 Çekirdeğine şunları ekleyen bir eklenti de sağlar:

- Biri yönergeler için (ICCM), diğeri veri için (DCCM), çekirdeğe sıkı bağlı iki adanmış bellek. Bu bellekler düşük-gecikmeli erişim ile SECDED ECC (single-error correction and double-error detection error correcting codes) koruması sağlar. Bellekler 4, 8, 16, 32, 48, 64, 128, 256, ya da 512KB olarak yapılandırılabilir.
- Bir isteğe bağlı 4-yönlü küme-ilişkisel eşlikli ya da ECC korumalı yönerge önbellegi.
- Bir isteğe bağlı 255 dış kesinti destekleyen Programlanabilir Kesinti Denetleyicisi (PIC).
- Yönerge getirme için dört sistem veri yolu arayüzü (IFU Bus Master (IFU Veri Yolu Ustası)), veri erişimleri (LSU Bus Master (LSU Veri Yolu Ustası)), ayıklama erişimleri (Debug Bus Master (Ayıklama Veri Yolu Ustası)), sıkı bağlı belleklere (64-bit AXI4 ya da AHB-Lite veri yolları olarak yapılandırılabilir) dış DMA erişimleri (DMA Slave Port (DMA Köle Portu)).
- RISC-V Ayıklama spesifikasyonu ile uyumlu Çekirdek Ayıklama Ünitesi.



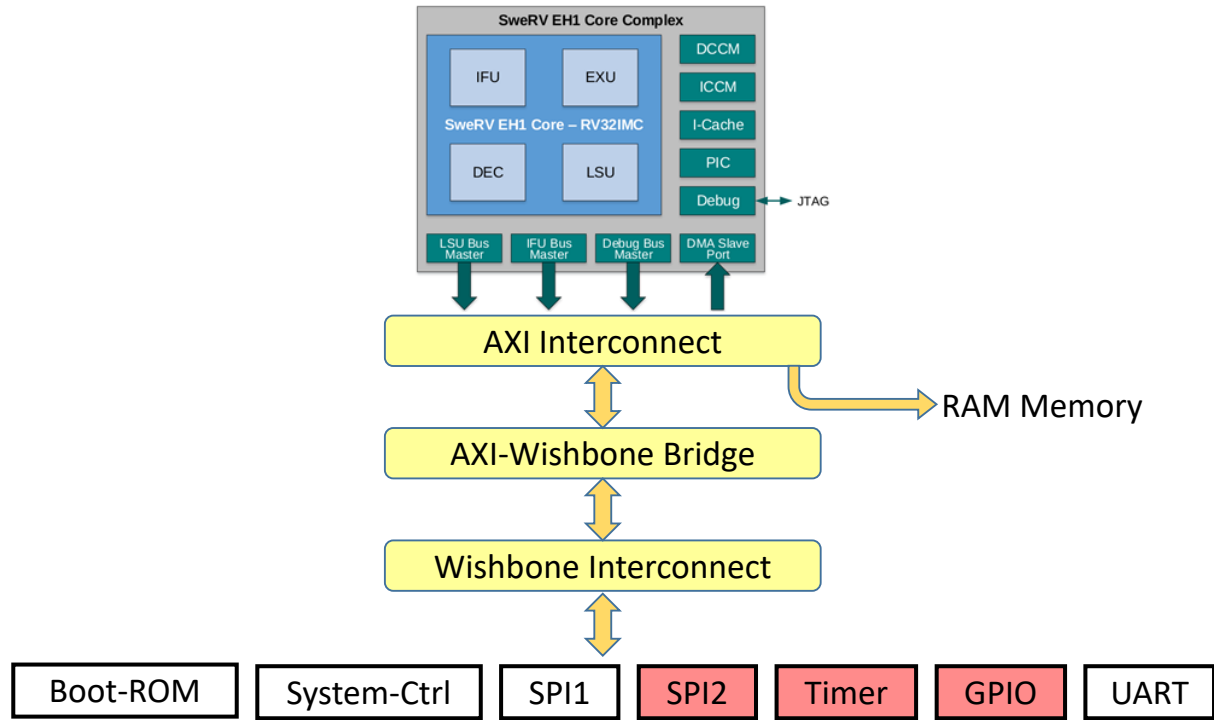
Figür 19. SweRV EH1 Core Complex

(figure from https://github.com/chipsalliance/Cores-SweRV/blob/master/docs/RISC-V_SweRV_EH1_PRM.pdf)

B. Genişletilmiş SweRVolf SoC

Bu RVfpga paketinde SweRV EH1 Core Complex üzerine kurgulanmış **SweRVolf SoC sürüm 0.7** kullanıyoruz. SweRVolf SoC RVfpga'ın indirmesiyle sağlanmış olup şuradan da erişilebilir: <https://github.com/chipsalliance/Cores-SweRVolf/releases/tag/v0.7>.

SweRV EH1 Core Complex (Figür 19'a göz at) yanı sıra, SweRVolf SoC ayrıca bir Boot ROM, bir UART, bir Sistem Denetleyicisi ile bir SPI denetleyicisi (Figür 20 bunları beyazla gösterir) de içerir. Bu kursta SweRVolf SoC'yi daha çok işlevsellikle genişletiyoruz, bir başka SPI denetleyicisi (SPI2), bir GPIO (Genel Amaçlı Girdi/Çıktı), 8-sayı 7-Kesimli Ekranlar ile bir zamanlayıcı (Figür 20 Sistem Denetleyicisi içinde bulunan 7-Kesimli Ekran dışındaki bu çevre birimlerini kırmızıyla gösterir) gibi. SweRV EH1 Çekirdeği bir AXI veri yolu kullandığı için, çevre birimleri de bir Wishbone veri yolu kullandığı için SoC ayrıca bir AXI-Wishbone köprüsü de içerir.



Figür 20. Genişletilmiş SweRVolf SoC

Tablo 5 çekirdeğe Wishbone ara bağlantısıyla bağlanmış çevre birimlerinin bellek-eşlenmiş adreslerini gösterir.

Tablo 5. Genişletilmiş SweRVolf SoC çevre birimlerinin bellek-eşlenmiş adresleri

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

i. Girdi/Çıktı

The SweRVolf SoC'si çevre birimleriyle iletişim kurmak için iki tür donanım kullanır: Verilog'da yazılmış özel yapım denetleyiciler ile OpenCores'dan [<https://opencores.org/>], ücretsiz açık kaynak iş birliği duygusuyla geçit IP (Fikri Mülkiyet) çekirdekleri geliştirilmesi için bir topluluk, açık-kaynak denetleyiciler. SweRVolf SoC'sinin bu kursta kullandığımız genişletilmiş sürümü aşağıda listelenmiş olan, RVfpga 6-10 arası Deneylerde detayıyla kullanıp, açıklayıp, üstüne üstlük genişleteceğimiz, Girdi/Çıktı (I/O) arayüzlerini içerir.

- **Sistem Denetleyicisi:** sistem denetleyicisi SoC sürüm bilgisi, RAM ilk değerlendirme durumlu, RISC-V makine zamanlayıcılı (<https://github.com/chipsalliance/Cores-SweRVolf>) içerisinde eksiksiz bellek eşlemesini bulabilirsin) yazmaç tutma gibi yaygın sistem işlevlerini barındırır. Dahası, biz Sistem Denetleyicisini, Nexys A7 kartındaki 8-sayı 7-Kesimli Ekranlarda göstermek üzere değeri depolamak için iki yazmaçla genişlettik, bunlar 0x80001038 ile 0x8000103C belleklerinde eşlenmiştir.

- **SPI:** iki açık-kaynak SPI denetleyicileri (şuradan alınmıştır https://opencores.org/projects/simple_spi_and_named_SPI1_and_SPI2) SweRVolf'un genişletilmiş sürümünde gerçekleştirilmiştir. Açıktaki yazmaçlar (SPI_SPCR, SPI_SPSR, SPI_SPDR, SPI_SPER, SPI_SPSS) 0x80001040 - 0x8000107F (SPI1 için) adresleri ile 0x80001100 - 0x8000113F (SPI2 için) adresleri aralığında eşlenmiştir.
- **Zamanlayıcı:** Şuranın zamanlayıcı modülünü kullanıyoruz <https://opencores.org/projects/ptc>. Yazmaçları 0x80001200'dan 0x800012FF'e olan adres aralığında eşlenmiştir.
- **GPIO:** Şuranın GPIO denetleyicisini kullanıyoruz <https://opencores.org/projects/gpio>. 0x80001400'dan 0x800014FF'e olan adres aralığında eşlenmiş 32 I/O portu içerir. Girdi ya da çıktı olarak yapılandırılabilmesi için bütün pinler birer üçdurumlu arabellek modülüne bağlanmıştır.
- **UART:** SweRVolf'ta bir açık-kaynak UART denetleyicisi (şuradan erişilebilir <https://opencores.org/projects/uart16550>) vardır. Açıktaki yazmaçları 0x80002000 ile 0x80002FFF arasındaki adreslere eşlenmiştir.

ii. Bellek

SweRVolf SoC'si, bir Boot ROM'un yanı sıra kullanıcının RAM ile SPI Flash bellekleri içerebilmesi için gerekli donanımı içerir.

- **Boot ROM:** bir Boot (Önyükleme) ROM'u bir ilk-aşama bootloader (ön yükleyici) içerir. Sistem sıfırlandıktan sonra SweRVolf SoC'si 0x80000000'dan 0x80000FFF'e adresleri kaplayan bu alandan ilk yönergeleri getirmeye başlayacaktır.
- **RAM:** SweRVolf SoC'si bir bellek denetleyicisi içermez ancak bellek eşlemesinin ilk 128MiB'ini ayırır (0x00000000-0x07FFFFFF), AXI veri yolunu açığa koyar ki kullanıcı RAM belleğine bir bellek denetleyicisiyle erişebilsin.
- **SPI Flash:** SPI Flash belleği de önceki bölümde tanımlanan SPI1 denetleyicisi kullanılarak içerilebilir (adres aralığı: 0x80001040-0x8000107F).

iii. Ara Bağlantı

SweRV EH1 Çekirdeği çekirdek ile belleği bağlamak için bir AXI4 veri yolu kullanır. Veri yolu bir AHB-Lite veri yolu olarak da yapılandırılabilir ancak o seçeneği bu içeriklerde kullanmayacağız. Çevre birimlerinin (I/O aygıtları) hepsi bir Wishbone veri yoluna, OpenCore CPU ile çevre birimlerinde çok kullanılan bir açık kaynak veri yolu, bağlıdır. Sistem çekirdeği çevre birimlerine bağlamak için bir AXI'dan Wishbone'a giden köprü de içerir (Figür 20'de gösterildiği gibi).

Bu bölümde AXI4 veri yolu ile Wishbone veri yolunun işlemini kısaca tanımlıyoruz. Eğer bu veri yollarının spesifikasyonu konusunda bilgini genişletmek istiyorsan aşağıda sağlanan referansları kullanabilirsiniz.

AXI4 Veri Yolu

SweRV EH1 Core Complex dış dünyayla iletişim kurmak için bir AXI4 Ara Bağlantı kullanır (Figür 19'a göz at). Advanced eXtensible Interface (Gelişmiş Genişletilebilir Arayüz) (AXI) çokça işlemcinin kullandığı yaygın bir veri yolu olup ARM Advanced Microcontroller Bus Architecture on-chip interconnect spesifikasyonunun bir parçasıdır.

İzleyen alt bölümlerde AXI4 ara bağlantısının ana yönlerinden birkaçını kısaca açıklayacağız. AXI spesifikasyonunun hepsini şu dokümanda bulabilirsin:
https://static.docs.arm.com/ihi0022/e/IHI0022E_amba_axi_and_ace_protocol_spec.pdf

• AXI Veri Yolu Ana Özellikleri

AXI veri yolu teknolojisinin ana özellikleri şunlardır:

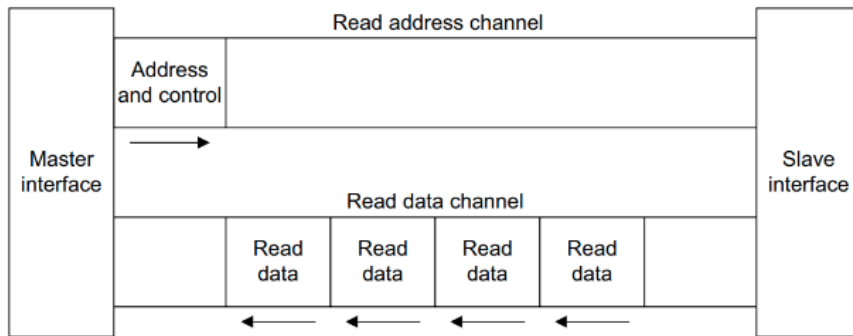
- Yüksek-bant genişliği tasarımlarına da düşük-gecikme tasarımlarına da uygundur
- Karmaşık köprüler kullanmadan yüksek-sıklıklı işlem sağlar
- Bileşenlerin geniş bir aralığının arayüz gerekliliklerini karşılar
- İlk erişimde yüksek gecikmeli bellek denetleyicilerine uygundur
- Ara bağlantı mimarilerinin gerçekleştirmesinde esneklik sağlar
- Eldeki AHB ile APB arayüzlerine geri uyumludur
- Adres/denetim ile veriye ayrı fazlar sağlar
- Hizalanmamış veri aktarımlarını destekler (bayt seçici vurumlarını kullanarak)
- Yalnızca başlangıç adresi yayınlı patlama-tabanlı hareketlere olanak sağlar
- Okumayla yazma için ayrı veri kanalları sağlar, ki bu düşük-maliyetli DMA yapılmasına olanak sağlayabilir
- Adres bilgisinin veri aktarımından önce yayınlanmasına olanak sağlar
- Birden çok göze çarpan adres ile düzensiz hareket bitirme yayınına destek sağlar
- Zamanlama kapatması sağlamak için yazmaç aşamalarının kolayca eklenmesine olanak sağlar

• AXI Mimarisi

AXI protokolü şu bağımsız hareket kanallarını tanımlar:

- Adres oku
- Veri oku
- Adres yaz
- Veri yaz
- Tepki yaz

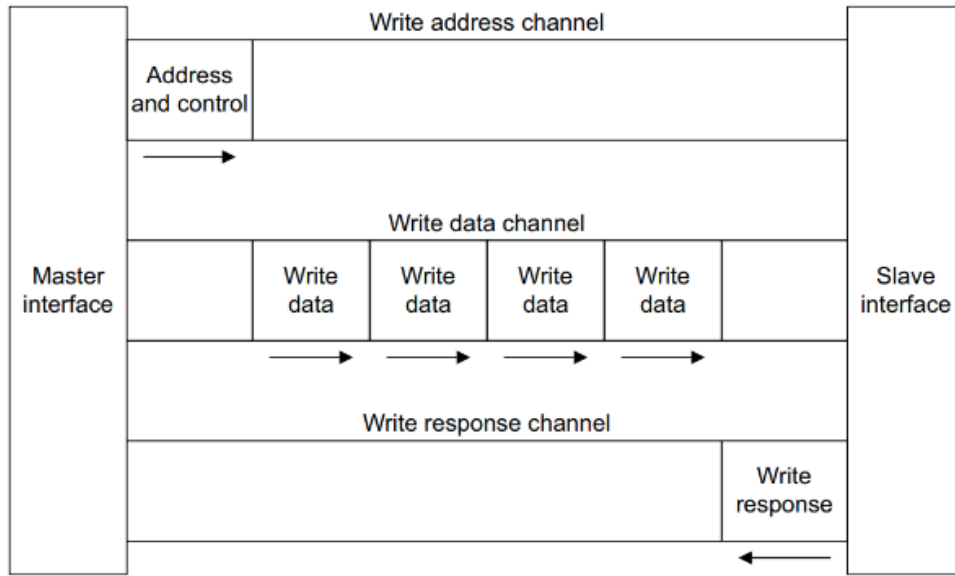
Figür 21 bir okuma hareketinin nasıl adres oku, veri oku kanallarını kullandığını gösterir. İlk olarak adres ile denetim bitleri usta aygıttan yollanır, ardından köle aygıt veriye veri oku kanalında yanıtlar



Figür 21. Okumaların kanal mimarisi

(figür şuradan https://static.docs.arm.com/ihi0022/e/IHI0022E_amba_axi_and_ace_protocol_spec.pdf)

Figür 22 bir yazma hareketinin nasıl adres yaz, veri yaz, veri yanıt kanallarını kullandığını gösterir. Bir okuma gibi, usta aygıt adres ile denetim bitlerini yollar. Ardından usta aygıt veriyi veri yaz kanalından yollar, sonra köle aygıt yanıt yollar.



Figür 22. Yazmaların kanal mimarisi

(figür şuradan https://static.docs.arm.com/ihi0022/e/IHI0022E_amba_axi_and_ace_protocol_spec.pdf)

AXI adres kanalı aktarılabacak verinin doğasını tanımlayan adreslerle denetim bilgilerini taşır. Veri, ustayla köle arasında şunların biri kullanılarak aktarılır:

- Veriyi köleden ustaya aktarmak için bir veri oku kanalı (Figür 21).
- Veriyi ustadan köleye aktarmak için bir veri yaz kanalı (Figür 22). Bir yazma hareketinde köle, ustaya aktarımın bittiğinin sinyalini vermek için tepki yaz kanalını kullanır (Figür 22).

- **AXI Sinyalleri**

Tablo 6. AXI veri yolunda kullanılan ana sinyallerle kısa tanımlarını gösterir. Sinyaller önceki bölümdeki beş kanala karşılık gelen beş grupta düzenlenmiştir:

- **Adres yaz kanalı** sinyalleri, adları **AW** ile başlayan
- **Veri yaz kanalı** sinyalleri, adları **W** ile başlayan
- **Tepki yaz kanalı** sinyalleri, adları **B** ile başlayan
- **Adres oku kanalı** sinyalleri, adları **AR** ile başlayan
- **Veri oku kanalı** sinyalleri, adları **R** ile başlayan

Tablo 6. AXI Sinyalleri(tablo şuradan https://static.docs.arm.com/ihi0022/e/IHI0022E_amba_axi_and_ace_protocol_spec.pdf)

Signal	Source: master/ slave	Input/ Output	Description
Aclk	Global	Input	Global clock signal.
AResetn	Global	Input	Global reset signal
AWID[3:0]	Master	Input	Write address ID.
AWADDR[31:0]	Master	Input	Write address.
AWLEN[3:0]	Master	Input	Write burst length.
AWSIZE[2:0]	Master	Input	Write burst size.
AWBURST[1:0]	Master	Input	Write burst type.
AWLOCK[1:0]	Master	Input	Write lock type.
AWCACHE[3:0]	Master	Input	Write cache type.
AWPROT[2:0]	Master	Input	Write protection type.
WDATA[31:0]	Master	Input	Write data.
ARID[3:0]	Master	Input	Read address ID.
ARADDR[31:0]	Master	Input	Read address.
ARLEN[3:0]	Master	Input	Read Burst length.
ARSIZE[2:0]	Master	Input	Read Burst size.
ARLOCK[1:0]	Master	Input	Read Lock type.
ARCACHE[3:0]	Master	Input	Read Cache type.
ARPROT[2:0]	Master	Input	Read Protection type.
RDATA[31:0]	Master	Input	Read data.
WLAST	Master	Input	Write last.
RLAST	Slave	Output	Read last.
AWVALID	Master	Output	Write address valid.
AWREADY	Slave	Output	Write address ready.
WVALID	Master	Output	Write valid.
RAVLID	Slave	Output	Read valid.
WREADY	Slave	Output	Write ready.
BID[3:0]	Slave	Output	Write Response ID.
RID[3:0]	Slave	Output	Read response ID.
BRESP[1:0]	Slave	Output	Write response.
RRESP[1:0]	Slave	Output	Read response.
BVALID	Slave	Output	Write response valid.

Wishbone veri yolu

SweRVolf çevre birimleri Taşınır IP Çekirdekleri için Wishbone Yongadaki-Sistem (System-on-Chip) (SoC) Ara Bağlantı Mimarisini kullanır (<https://opencores.org/howto/wishbone>). Bu veri yolunun en büyük amacı Yongadaki-Sistem entegrasyon problemlerini çözerek tasarım yeniden kullanımını arttırmaktır. Eskiden IP çekirdekleri entegrasyonu zorlaştıran, standart olmayan ara bağlantı şemaları kullanırdı. Bu standart olmayan ara bağlantılar çekirdekleri birbirine bağlamak için özel tutkal mantığı oluşturulmasını gerektirirdi. Wishbone veri yolu gibi standart bir ara bağlantı şeması benimsenirse son kullanıcı çekirdekleri daha hızlı, daha kolay entegre edebilir.

- **Wishbone ana özellikleri**

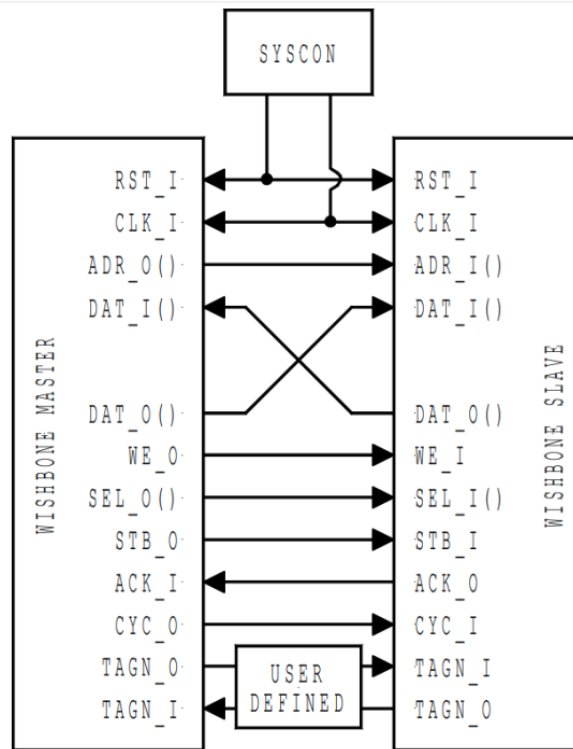
Wishbone veri yolu teknolojinin ana özellikleri şunlardır:

- Büyük proje ekiplerinin kullandığı tasarım metodolojilerini destekler.
- Şunların da kapsandığı dolgun bir popüler veri aktarım veri yolu protokolleri kümesi içerir:
 - READ/WRITE dönümleri
 - BLOCK aktarım dönümleri
 - READ/MODIFY/WRITE dönümleri
- 64-bite kadar modüler veri yolu genişlikleriyle işlenen boyutları sağlar.

- YÜKSEK SON HANELİ de DÜŞÜK SON HANELİ de veri sıralamayı destekler.
- Uçtan uca, paylaşımlı veri yolu, çaprazlayıcı anahtar, anahtarlama kumaş ara bağlantıları gibi değişik çekirdek ara bağlantılarını destekler.
- IP çekirdeklerinin veri aktarım hızlarını darboğazlamalarına olanak sağlayacak tokalaşma protokolleri içerir.
- Bir saatlik (single clock) veri aktarımlarını destekler.
- Normal dönüm sonlandırmasını, yeniden deneme sonlandırmasını, hata kaynaklı sonlandırmayı destekler.
- Modüler adres genişliklerini destekler.
- Köleler için bölümsel adres çözme şeması sağlar. Bu yüksek hızlı adres çözümünü kolaylaştırır, gereksiz mantığı azaltır, değişik adres boyutlandırma, ara bağlantı yöntemlerini destekler.
- Kullanıcı-tanımlı etiketler sağlar. Bunlar bilgiyi bir adrese ya da veri yoluna ya da veri yolu dönümüne uygulamada yararlıdır. Kullanıcı-tanımlı etiketler veri yolu dönümlerini şunun gibi bilgileri tanımak için değiştirirken özellikle yararlıdır:
 - i. Veri aktarımları
 - ii. Eşlik ya da hata düzeltme bitleri
 - iii. Kesinti vektörleri
 - iv. Önbellek denetleme işlemleri
- Esnek sistem tasarımları için bir Usta/Köle mimarisi içerir.
- Çoklu işlem (çoklu-USTA) kabiliyetleri vardır. Bu değişik SoC yapılandırmalarına olanak sağlar
- Son kullanıcının bir tahkim metodolojisi içerir (öncelik tahkimcisi, round-robin tahkimcisi gibi)

• **Wishbone Mimarisi, Sinyalleri**

Figür 23 bir usta (burada SweRV EH1 çekirdeği) ile bir köle (burada GPIO, SPI, ... gibi bir çevre birimi) arasındaki Wishbone üzerinden standart ilişkiyi görselleştirir. Wishbone veri yolu AXI4 veri yolundan çok daha yalındır, daha az sinyal kullanır, Tablo 7’de gösterildiği gibi.

**Figür 23. Wishbone Mimarisi**(figür şuradan <https://opencores.org/howto/wishbone>)**Tablo 7. Wishbone Sinyalleri**(tablo şuradan <https://opencores.org/howto/wishbone>)

Signal name	description	Signal name	Description
CLK_O	It coordinates all activities for the internal logic within the WISHBONE interconnect. The INTERCON module connects the [CLK_O] output to the [CLK_I] input on MASTER and SLAVE	CLK_I	All WISHBONE output signals are registered at the rising edge of [CLK_I]. All WISHBONE input signals are stable before the rising edge of [CLK_I].
RST_O	It forces all WISHBONE interfaces to restart. All internal self-starting state machines are forced into an initial state. The INTERCON connects the [RST_O] output to the [RST_I] input on MASTER and SLAVE	DAT_I()	The data input array [DAT_I()] is used to pass binary data. The array boundaries are determined by the port size, with a maximum port size of 64-bits (e.g. [DAT_I(63..0)]).
		DAT_O()	The data output array [DAT_O()] is used to pass binary data. The array boundaries are determined by the port size, with a maximum port size of 64-bits (e.g. [DAT_O(63..0)]).
		RST_I()	The reset input [RST_I] forces the WISHBONE interface to restart
		TGD_I()	Data tag type [TGD_I()] is used on MASTER and SLAVE interfaces. It contains information that is associated with the data input array [DAT_I()], and is qualified by signal [STB_I].
		TGD_O()	Data tag type [TGD_O()] is used on MASTER and SLAVE interfaces. It contains information that is associated with the data output array [DAT_O()], and is qualified by signal [STB_O]

Signal name	Description	Signal name	Description
ACK_I	The acknowledge input [ACK_I], when asserted, indicates the normal termination of a bus cycle	ACK_O	The acknowledge output [ACK_O], when asserted, indicates the termination of a normal bus cycle
CYC_O	The cycle output [CYC_O], when asserted, indicates that a valid bus cycle is in progress	CYC_I	The cycle input [CYC_I], when asserted, indicates that a valid bus cycle is in progress
STALL_I	The pipeline stall input [STALL_I] indicates that current slave is not able to accept the transfer in the transaction queue	STALL_O	The pipeline stall signal [STALL_O] indicates that the slave can not accept additional transactions in its queue
ERR_I	The error input [ERR_I] indicates an abnormal cycle termination	ERR_O	The error output [ERR_O] indicates an abnormal cycle termination
RTY_I	The retry input [RTY_I] indicates that the interface is not ready to accept or send data, and that the cycle should be retried	RTY_O	The retry output [RTY_O] indicates that the interface is not ready to accept or send data, and that the cycle should be retried
STB_O	The strobe output [STB_O] indicates a valid data transfer cycle	STB_I	The strobe input [STB_I], when asserted, indicates that the SLAVE is selected. A SLAVE shall respond to other WISHBONE signals only when this [STB_I] is asserted
WE_O	The write enable output [WE_O] indicates whether the current local bus cycle is a READ or WRITE cycle	WE_I	The write enable input [WE_I] indicates whether the current local bus cycle is a READ or WRITE cycle

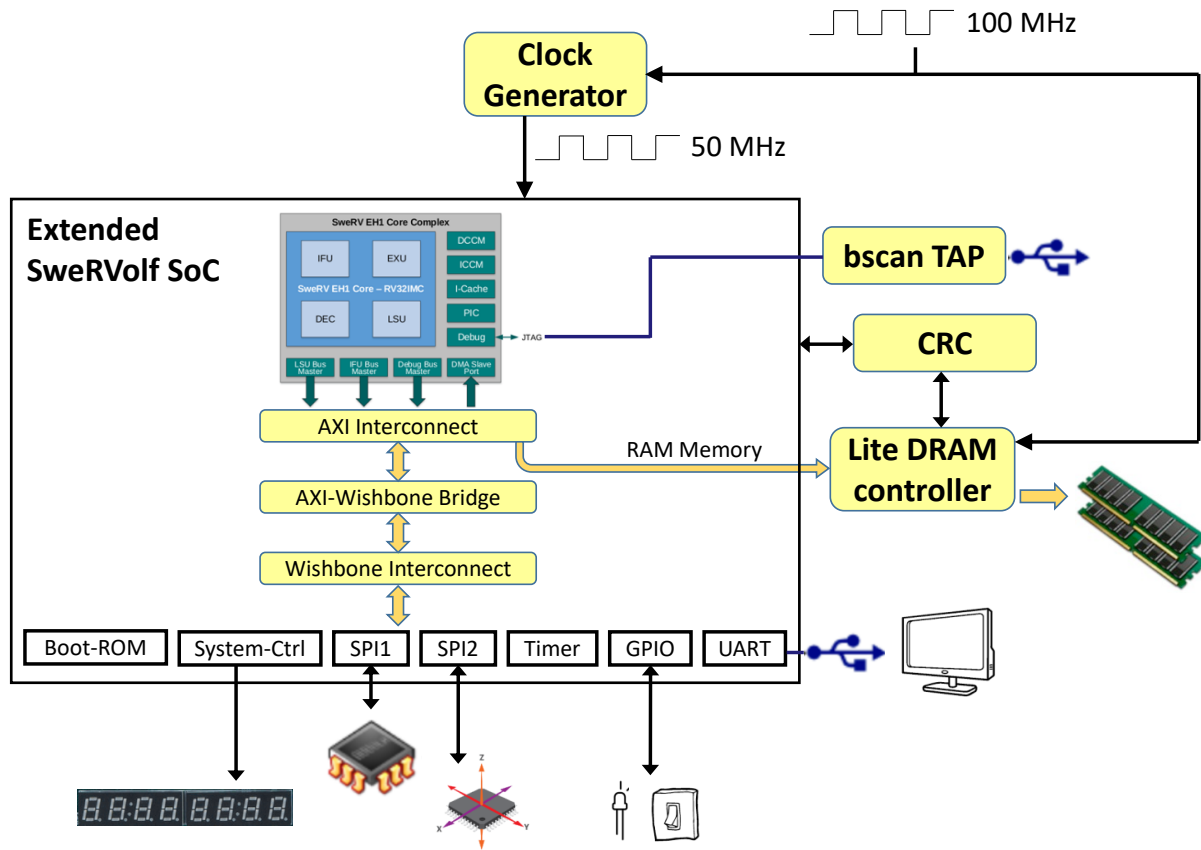
C. Nexys A7 Kartı ile Simülasyonda Genişletilmiş SweRVolf SoC'si

Genişletilmiş SweRVolf SoC'si (Figür 20) (1) Nexys A7 (ya da Nexys4 DDR) FPGA kartı üzerinde, ki bu yapılandırmaya **RVfpga** denir, ya da (2) simülasyonda, ki buna **RVfpgaSIM** denir, çalıştırılabilir.

i. RVfpga

RVfpga, Nexys A7 FPGA kartı, kartın çevre birimlerine hedeflenmiş Genişletilmiş SweRVolf SoC'dir (Figür 20). (Nexys 4 DDR FPGA kartı da kullanılabilir). **RVfpga**'in kullandığı ana ögeler Figür 24'te görselleştirilmiştir:

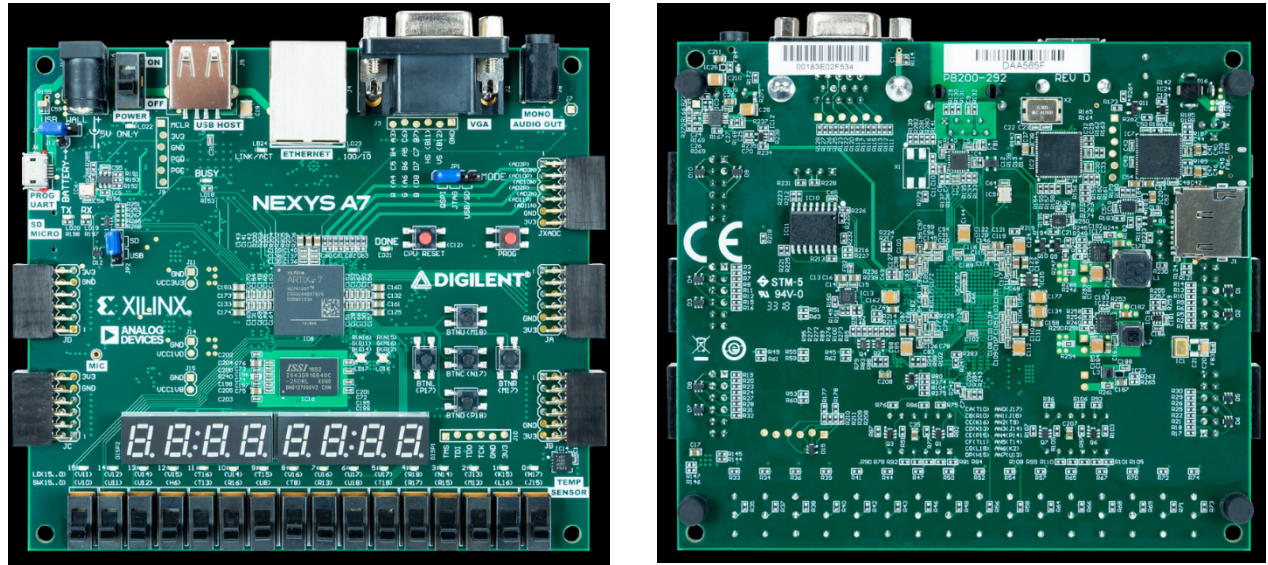
- FPGA'ye programlanan donanım:
 - **Genişletilmiş SweRVolf SoC'si**
 - **Lite DRAM denetleyicisi**
 - **Saat Oluşturucu:** the Nexys A7 kartı, **Lite DRAM denetleyicinin** kullandığı bir **100 MHz** kristal salıngaç içerir. Bu saatin sıklığı **SweRVolf SoC** kullanımı için **50 MHz'ye** geri ölçeklenmiştir.
 - **Saat Alanı Çaprazlama modülü:** iki alanın bağlantısı: SweRVolf SoC ile Lite DRAM.
 - **JTAG portu için BSCAN mantığı**
- Nexys A7 (ya da Nexys4 DDR) FPGA kartından RVfpga'de kullanılan Bellek/Çevre Birimleri:
 - **DDR2 bellek** (yukarıda sözü edilen Lite DRAM üzerinden erişilir)
 - **USB bağlantısı**
 - **SPI Flash bellek**
 - **SPI İvmeölçer**
 - **16 LED, 16 Anahtar**
 - **8-sayı 7-Kesimli Ekranlar**



Figür 24. RVfpga

Nexys A7 kartı (Figür 25) elektrik ile bilgisayar mühendisliği müfredatı için önerilen bir eğitimci kartıdır. Bu kart \$265 (ya da indirimli akademik ücretlendirmeye \$198.75 – bir Digilent üyeliğine .edu email adresiyle kaydol) tutar. Digilent, Nexys A7 kartının kapsamlı bir referans manüelini şurada sağlar:

https://reference.digilentinc.com/media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf. Bu kart 5V duvar girişinden (kartla sağlanmaz) ya da PC'den karttaki microUSB bağlayıcısıyla güç alabilir. Bir Microchip PIC24 mikrodenetleyici FPGA'e yükleme sürecini yönetir, bu da kartı kullanıcı-dostu bir seçenek yapar. Kart Xilinx'in Vivado Design Suite'i ya da OpenOCD ile programlanabilir. İstenen yapılandırma FPGA'e dört farklı kaynaktan biriyle indirilebilir: bir FAT32 formatlı MicroSD kart, bir FAT32 formatlı USB kalem sürücü, iç flash bellek, ya da bir JTAG arayüzü.



Figür 25. Digilent'in Nexys A7 FPGA kartı
 (figure from <https://reference.digilentinc.com/>)

Nexys A7-100T FPGA kartı şu arayüzlerle aygıtları içerir:

- 128 MiB DDR RAM
- 128 Mibit SPI Flash Bellek
- 8-sayı 7-Kesimli Ekran
- 16 Anahtar
- 16 LED
- Bir mikrofon, ses jakı, VGA 25 portu, USB konakçı portu, RGB-LEDler, I2C sıcaklık sensörü, SPI ivmeölçeri, gibisini içeren sensörler, bağlayıcılar
- Xilinx Artix-7 FPGA, şu özelliklerle:
 - 15.850 Mantık dilimlik dört 6-girdi LUT, 8 flip-flop.
 - 4.860 Kibitlik toplam blok RAM
 - 6 saat yönetim tuğlası (clock management tile) (CMTs)
 - 170 I/O ucu
 - 450 MHz iç saat sıklığı

ii. RVfpgaSIM

SweRVolf SoC simülasyon yapmayı etkinleştirmek için bir Verilog örtüsü de içerebilir. Buna **RVfpgaSIM** denir, bu HDL simülatörlerde kullanılmak üzere Genişletilmiş SweRVolf SoC'yi bir testbenchte örtüleyen bir simülasyon hedefidir. İki biçimde kullanılabilir:

- Verilator'de bütün-sistem simülasyonları için (ya da diğer HDL simülatörlerinde)
- OpenOCD ile JTAG VPI üzerinden bir ayıklayıcıya bağlanmak için

Çok açık-kaynak HDL simülatör bulunuyor olsa da biz Verilator (<https://www.veripool.org/wiki/verilator>) kullanıyoruz. Bu açık, ücretsiz HDL simülatörü sentezlenebilir Verilog ile SystemVerilog kabul ederken en hızlı Verilog/SystemVerilog simülatörü olduğunu söyler. Endüstri ile akademide çok kullanılır; ARM ile RISC-V sağlayıcılarının IPLerinde kutudan destek sağlar; Chips Alliance ile Linux Foundation kılavuzluğu altındadır.

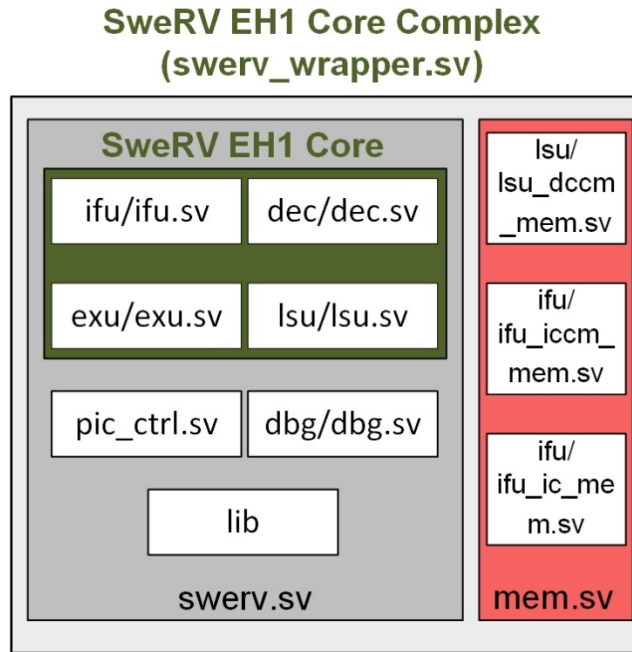
D. Dosya Yapısı

Önceki bölümlerde bu içeriklerde kullandığımız sistemin yüksek-düzye düzenini gösterdik, **SweRV EH1 Core Complex**'ten (Figür 19), **Geniştirilmiş SweRVolf SoC**'ye (Figür 20), son olarak da **RVfpga** (Figür 24) ile **RVfpgaSIM** gerçekleştirmelerine.

Bu bölümde bütün sistemin dosya yapısını tanımlıyoruz. Bu açıklamaları okurken dosyaları açıp onlara kendi bilgisayarında bak. Dosyalara şuradan erişilebilir **[RVfpgaPath]/RVfpga/src**.

i. SweRV EH1 Core Complex

Figür 26, **SweRV EH1 Core Complex** (Figür 19) dosya yapısını gösterir. Çekirdek üç ana blokta düzenlenmiştir: bir SweRV EH1 Core (yeşil) ile diğer öğeleri (Kesinti Denetleyicisi ya da Ayıklama Ünitesi gibi) içeren SweRV örtüsü (gri) ile Veri/Yönerge bellekleri ile Yönerge Ön Belleği (kırmızı).



Figür 26. SweRV EH1 Core Complex

SweRV EH1 Core Complex için Verilog dosyalarına şu klasörden erişilebilir:
[RVfpgaPath]/RVfpga/src/SweRVolfSoC/SweRVEh1CoreComplex

Dosyalara biz bu bölümde değindikçe kendin görmek için o dizini bilgisayarında bul.

SweRV EH1 Core Complex için üst dosya şu dosyadadır: *swerv_wrapper.sv*; üst modülün adı **swerv_wrapper**'dir, Figür 26'da gri ile kırmızıyla parlatılmış iki bloğa denk gelen iki modülü somutlaştırır:

- **mem** (*mem.sv* içerisinde gerçekleştirilmiştir): bu modül DCCM (*lsu_dccm_mem*, *lsu/lsu_dccm_mem.sv*), ICCM (*ifu_iccm_mem*, *ifu/ifu_iccm_mem.sv*), Instruction Cache (Yönerge Ön Belleği) (*ifu_ic_mem*, *ifu/ifu_ic_mem.sv*) modüllerinin gerçekleştirmelerinin somutlaştırmasını yapar.
- **swerv** (*swerv.sv* içerisinde gerçekleştirilmiştir): bu modül çekirdeği oluşturan üniteleri somutlaştırır.

SweRV EH1 Core (Figür 26'da yeşille parlatılmıştır) şu dört üniteden oluşur:

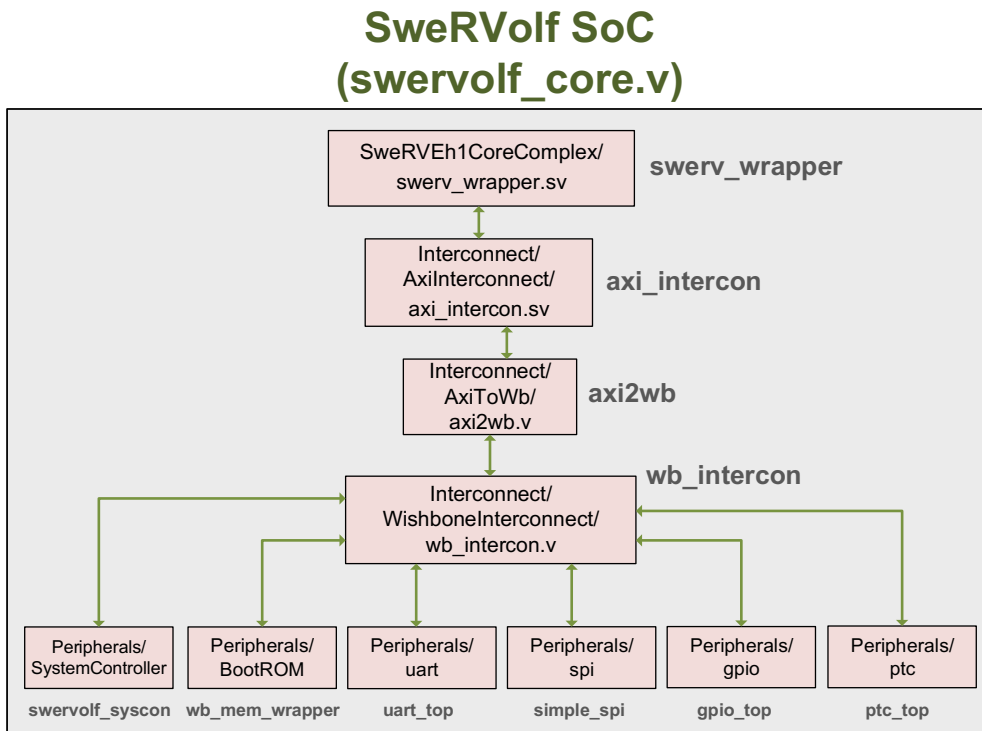
- Klasör **ifu** (Instruction Fetch Unit (Yönerge Getirme Ünitesi)): bu klasör lcache (yönerge önbelleği), Fetch (Getir), Branch Predictor (Dallandırma Kestiricisi), Aligner (Hizalayıcı) için Verilog dosyalarını (üst modüle *ifu.sv* içerisinde erişilebilir) içerir.
- Klasör **dec** (Decode Unit (Çözme Ünitesi)): bu klasör Instruction Decoding (Yönerge Çözme), Dependency Scoreboard (Bağımlılık Skor Tahtası), Register File (Kayıt Dosyası) için Verilog dosyalarını (üst modüle *dec.sv* içerisinde erişilebilir) içerir.
- Klasör **exu** (Execution Unit (Yürütme Ünitesi)): bu klasör çekirdekteki aritmetik/mantık üniteleri için Verilog dosyalarını (üst modüle *exu.sv* içerisinde erişilebilir) içerir: iki boruhatlı ALU, bir boruhatlı Multiplier (Çarpan), bir boruhatlı-dışı Divider (Bölen).
- Klasör **lsu** (Load Store Unit (Yükleme Depolama Ünitesi)): bu klasör boruhatlı Load/Store Unit (Yükleme/Depolama Ünitesi) için Verilog dosyalarını (üst modüle *lsu.sv* içerisinde erişilebilir) içerir.

Bu modülde içerilen diğer üniteler:

- Klasör **dbg** (Debug Unit (Ayıklama Ünitesi)): bu klasör çekirdeğin kalanını dinlenme moduna koyup, komutları/adresleri yollayıp, ardından çekirdeği normal moda döndürmekle sorumlu Debug Unit (Ayıklama Ünitesi) için Verilog dosyalarını (üst modüle *dbg.sv* içerisinde erişilebilir) içerir.
- Klasör **lib**: bu klasör AXI ile AHB-Lite Veri Yolları için Verilog dosyalarını içerir.
- Modül **pic_ctrl** (*pic_ctrl.sv* içerisinde gerçekleştirilmiştir): bu modül Programmable Interrupt Controller (Programlanabilir Kesinti Denetleyicisi) gerçekleştirir.

ii. Genişletilmiş SweRVolf SoC

Figür 27 Genişletilmiş SweRVolf SoC (Figür 20) dosya yapısını gösterir. SoC, Figür 20



Figür 27. Genişletilmiş SweRVolf SoC

Genişletilmiş SweRVolf SoC dosyaları şuranın içerisinde:

[RVfpgaPath]/RVfpga/src/SweRVolfSoC

Dosyalara biz bu bölümde değindikçe kendin görmek için o dizini bilgisayarında bul.

Genişletilmiş SweRVolf SoC için üst modüle şuradan erişilebilir:

[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v. Dosyayı açtığında Genişletilmiş SweRVolf SoC (Figür 20) modüllerini içerdiğini göreceksin:

- **axi_intercon** (*Interconnect/AxiInterconnect/axi_intercon.v* içerisinde erişilebilir): bu modül başka bir dosya üzerinden satır 105'te içerilir (``include "axi_intercon.vh"`). SweRV EH1 Core Complex'i AXI-to-Wishbone köprüsüyle bağlar.
- **axi2wb** (*Interconnect/AxiToWb/axi2wb.v* içerisinde erişilebilir): *swervolf_core.v* satır 174'te somutlaştırılan bu modül AXI tabanlı EH1 Çekirdeği ile Wishbone-tabanlı çevre birimleri arasındaki iletişimi sağlayan AXI-to-Wishbone köprüsüdür.
- **wb_intercon** (*Interconnect/WishboneInterconnect/wb_intercon.v* içerisinde erişilebilir): bu modül başka bir dosya üzerinden satır 168'de içerilir (``include "wb_intercon.vh"`). Sonra çözümleyip değiştireceğimiz bir çoklayıcıyla AXI-to-Wishbone köprüsünü değişik çevre birimleriyle bağlar.
- **wb_mem_wrapper** (*Peripherals/BootROM/wb_mem_wrapper.v* içerisinde erişilebilir): *swervolf_core.v* satır 210'da yukarıda tanımlanan Boot Memory örtüsü somutlaştırılır. Basit RAM modülü olan **dpram64** modülünü somutlaştırır (*Peripherals/BootROM/dpram64.v* içerisinde erişilebilir).
- **swervolf_syscon** (*swervolf_0.7/rtl/swervolf_syscon.v* içerisinde erişilebilir): *swervolf_core.v* satır 225'te somutlaştırılan bu modül System Controller (Sistem Denetleyicisi) tanımlar.
- **simple_spi**: OpenCores'dan alınmış SPI denetleyicisi, *Peripherals/spi/simple_spi_top.v* içerisinde erişilebilir. *swervolf_core.v* 251 (SPI1) ile 387 (SPI2) satırlarında somutlaştırılır.
- **uart_top**: OpenCores'dan alınmış UART denetleyicisi, *Peripherals/uart/uart_top.v* içerisinde erişilebilir. *swervolf_core.v* satır 272'de somutlaştırılır.
- **gpio_top**: OpenCores'dan alınmış GPIO denetleyicisi, *Peripherals/gpio/gpio_top.v* içerisinde erişilebilir. *swervolf_core.v* satır 338'de somutlaştırılır.
- **swerv_wrapper** (*SweRVEh1CoreComplex/swerv_wrapper.v* içerisinde erişilebilir): Western Digital'in SweRV EH1 Core Complex, önceki bölümde tanımlanan, somutlaştırması (*swervolf_core.v* satır 406) (Figür 26).

iii. Kartta yürütme ile simülasyon için örtüler

SİMÜLASYON:

RVfpgaSIM, Genişletilmiş SweRVolf SoC'i HDL simülatörlerinde kullanılan bir testbench'te örtüleyen simülasyon hedefidir. Şuradan erişilebilir **[RVfpgaPath]/RVfpga/src/rvfpgasim.v**.

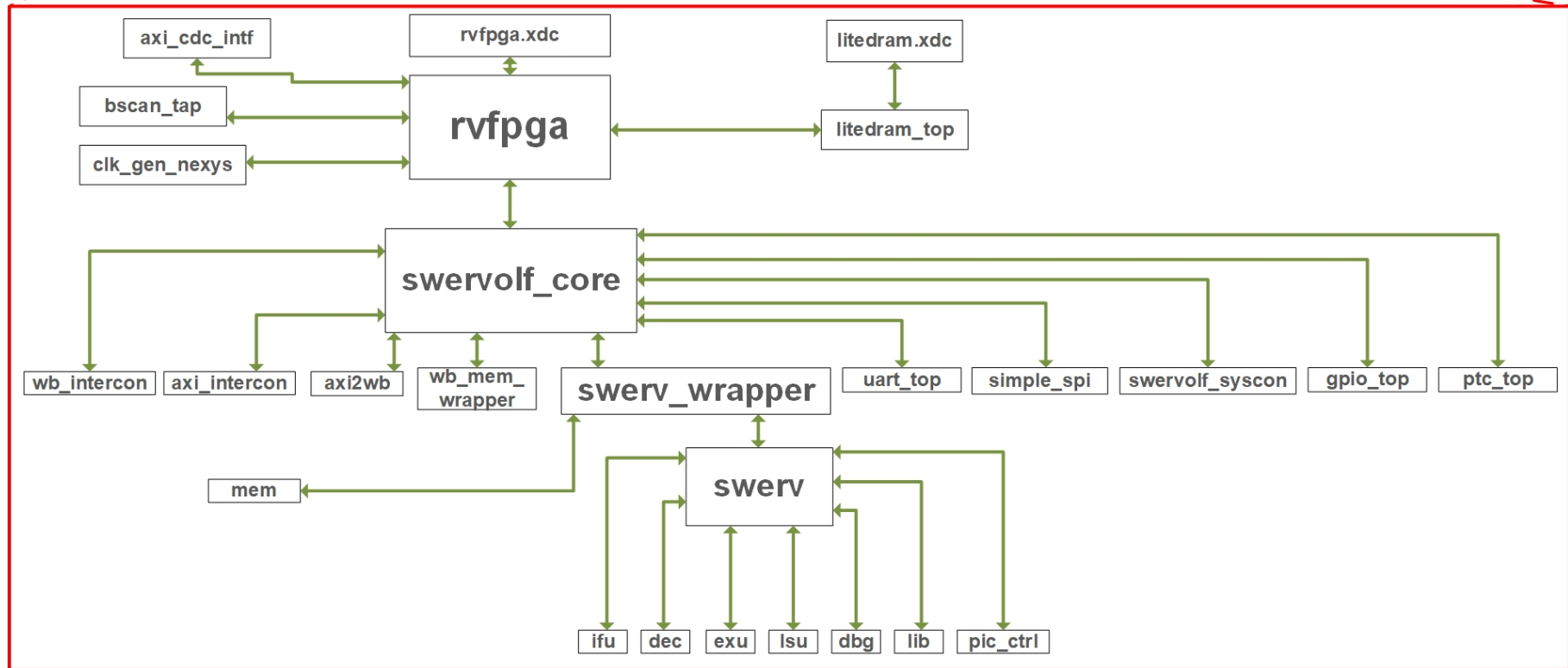
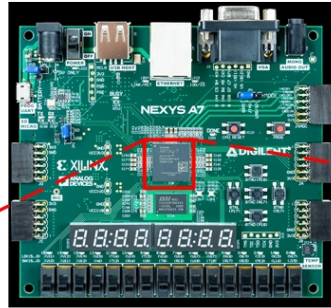
KARTTA YÜRÜTME:

RVfpga (şuradan erişilebilir: **[RVfpgaPath]/RVfpga/src/rvfpga.v**), **Extended SweRVolf SoC**'i Nexys A7 FPGA kartıyla çevre birimlerine hedefleyen bir örtüyle örter (Figür 24). Bu modül, birkaç diğer modülle birlikte (*clock generator module*, **clk_gen_nexys**, *clock*

domain crossing module, **axi_cdc_intf**, BSCAN module for the JTAG port, **bscan_tap**, gibi), iki ana SoC yapısını somutlaştırır:

- **swervolf_core**: önceki alt bölümde tanımlanan **Extended SweRVolf SoC** somutlaştırması (Figür 27). Bu ayrıca SoC ile kart arasındaki bağlantıları tanımlayan *rvfpga.xdc* adında bir kısıtlar dosyasını da gerektirir ([RVfpgaPath]/RVfpga/src/ içerisinde erişilebilir).
- **litedram_top**: [RVfpgaPath]/src/LiteDRAM/litedram_top.v dosyasında somutlaştırılan, SweRVolf SoC'sini DDR2 bellekle bağlayan LiteDRAM DDR2 Denetleyicisi için örtü. Bu da Bellek Denetleyicisi ile karttaki DDR2 Bellek arasındaki bağlantıları tanımlayan *litedram.xdc* adında bir kısıtlar dosyasını gerektirir ([RVfpgaPath]/RVfpga/src/LiteDRAM içerisinde erişilebilir).

Figür 28 bütün sistemin Nexys A7 FPGA kartında gerçekleştirmesinin hiyerarşisini özet olarak gösterir.



Figür 28. Nexys A7 FPGA kart gerçekleştirmesinin Modüller Hiyerarşisi

5. YAZILIM ARAÇLARINI KURMA

Aşağıdaki yönergeler Ubuntu 18.04 içindir, ancak diğer Linux işletim sistemleri, Windows ya da macOS de, yakın (neredeyse birebir) adımlar izler. Kimi durumlarda diğer OSler için belirli yönergeli kutular yerleştiriyoruz. Ubuntu kullanıyorsan, bu kutuları görmezden gelebilirsin.

Yönergeler şu araçları nasıl kuracağını gösterir

- A. Vivado:** Yongadaki Sistemi yeniden sentezleme için gereklidir. Bu genelde 6-10 arası, değişik niteliklerin taban SoC'ye eklenerek içerildiği deneylerde yapacağın bir iştir.
- B. VSCode (Visual Studio Code) ile PlatformIO:** bunlar GSG ile Deneylerde kullanılan ana araçlardır. FPGA'i programlayıp, üzerinde programları çalıştırma/ayıklama için kullanılırlar.
- C. Verilator ile GTKWave:** SoC'nin simülasyonunu yapıp değişik sinyalleri çözümleme için gereklidir. Bu araçları da genelde 6-10 arası deneylerde kullanacaksın.

Önemli olarak GSG ile Deneylerde yapacağın çoğu iş için VSCode ile PlatformIO'yu kurmak yetecektir. Ancak ileride daha çok kurulum yapmamak için diğer araçları (Vivado, Verilator, GTKWave) da şimdi kurmanı öneriyoruz.

Bu işlem (indirme hızına bağlı olarak) birkaç saat alabilir, ancak sürecin yoğunluğu programlar indirilip kurulurken beklemeye geçer.

A. Vivado'yu Kurma

Vivado, RISC-V FPGA için Verilog koduna bakma, değişiklik yapma, sentezleme için kullanılan bir Xilinx aracıdır. Sonraki deneylerde çok kullanacaksın. Kurulum yönergeleri burada bulunurken <https://reference.digilentinc.com/vivado/installing-vivado/start> aşağıda özetlenmiştir.

Windows: yukarıdaki web sayfası (<https://reference.digilentinc.com/vivado/installing-vivado/start>) Windows'ta Vivado kurulumu için de detaylı yönergeler içerir. Aşağıda Windows için belirli yönergeler gerektiğinde kutular yerleştiriyoruz.

macOS: Vivado macOS'te desteklenmez; bundan dolayı bu OS'te Vivado'yu kullanmak için Linux/Windows Sanal Makine gerekir.

1. Şuraya git <https://reference.digilentinc.com/vivado/installing-vivado/start>

2. Xilinx indirme sayfasına yönlendirileceksin:
<https://www.xilinx.com/support/download.html>

3. "Self Extracting Web Installer" (Kendini Çıkartan Web Kurucusu) olanı kurman önerilir. Bu doküman yazılırken indirme sayfasındaki şu linkteydi: [Xilinx Unified Installer 2019.2: Linux Self Extracting Web Installer](#)

WINDOWS: Bu doküman yazılırken Windows için "Self Extracting Web Installer" indirme sayfasında şu linkteydi: [Xilinx Unified Installer 2019.2: Windows Self Extracting Web Installer](#)

4. Kurucuyu indirmeden önce Xilinx üyeliğinde giriş yapman istenilecek. Eğer üyeliğin yoksa, oluşturman gerekecek.

5. Veri (binary) dosyasını yürüt. Bir terminal açıp yetkili (root) yap ("sudo su" yaz). Ardından veri (binary) dosyasını (Xilinx_Unified_2019.2_1106_2127_Lin64.bin) terminale sürükley. Eğer dosyayı yürütülebilir yapıp çalıştırmayı sorarsa, OK'i seç.

- **Sorun Giderme:** Eğer terminal yetki vermezse (permission denied), terminalde şunu yaz (veri (binary) dosyasının dizininde):
> sudo chmod +x ./Xilinx_Unified_2019.2_1106_2127_Lin64.bin
> sudo ./Xilinx_Unified_2019.2_1106_2127_Lin64.bin

WINDOWS: Windows'ta adım 3 ile 4'te indirdiğin .exe dosyasını üzerine çift-tıklayarak yürütmen yeterli.

6. Vivado kurucusu seni kurma sürecinde yönlendirecek. Önemli olarak:

- İndirilecek ürün olarak **Vivado'yu** (Vitis'i *değil*) seç.
- Vivado HL **Webpack'i** (Vivado HL System Edition'ı *değil*) seç; Webpack ücretsizdir.
- Ötekilerinde varsayılan seçili olmalı.

İpucu: Eğer Vivado'nun kurulum dizinini değiştirdiysen sonraki adımlarda yolu uygun olarak değiştirmen gerekecek.

WINDOWS: Adımlar 7 ile 8, Windows'ta gerekli değildir. Bu iki adımı görmezden gelip adım 9'a geçebilirsin.

7. Vivado kurulduktan sonra ortamı ayarlaman gerekecek. Bir terminal açıp şunu yaz:

```
source /tools/Xilinx/Vivado2019.2/settings64.sh
```

O satırı (source /tools/Xilinx/Vivado2019.2/settings64.sh) ~/.bashrc dosyasına ekle ki terminali açtığında hep çalışsın.

8. Şu satırı bir terminale yazarak Vivado'yu dene:

```
vivado
```

Sorun Giderme:

- Eğer sistemin o yürütülürü bulamazsa şunları yoluna (path değişkeni) eklemen gerekecek:

```
/tools/Xilinx/DocNav  
/tools/Xilinx/Vivado/2019.2/bin
```

- Şöyle bir hata alırsan "application-specific initialization failed...", bir terminale şunu yaz:

```
sudo ln -s /lib/x86_64-linux-gnu/libtinfo.so.6 /lib/x86_64-  
linux-gnu/libtinfo.so.5
```

9. Nexys A7 FPGA kartı için kablo sürücülerini manuel kurman gerekecek. Şunları bir terminal penceresinde yaz:

```
cd
/tools/Xilinx/Vivado/2019.2/data/xicom/cable_drivers/lin64/install_script/install_drivers/

sudo ./install_drivers
```

WINDOWS: Windows'ta Vivado kurulumu kendiliğinden Nexys A7 kartı için PlatformIO ile uyumsuz sürücülerini kurar. Dolayısıyla, Windows kullanıyorsan, **sürücülerini Ek B'de açıklandığı gibi güncellemelisin. Bunu Vivado kurulumu sürücülerin üzerine yazdığı için Hızlı Başlangıç Kılavuzunda yaptıysan bile yapmalısın.**

10. Diligent Kart Dosyalarını (Board Files) da manüel kurman gerekecek.

- vivado-boards [arşivini](#) GitHub deposundan indirip çıkar.
- Arşivden çıkarılmış klasörü açıp *new/board_files* dizinine git. Bu dizinin içindeki bütün klasörleri seçip kopyala.
- Vivado'nun kurulduğu klasörü aç (varsayılanda */tools/Xilinx/Vivado*). Bu klasörün içerisinde *<version>/data/boards/board_files* dizinine git, kart dosyalarını bu dizine yapıştır.
- *new/board_files* dizinine gidip şunu yazarak terminali de kullanabilirsin:

```
sudo cp -r *
/tools/Xilinx/Vivado/2019.2/data/boards/board_files
```

WINDOWS: Adım 10'da açıklandığı gibi indirilmiş klasörleri kopyala/yapıştır. Windows'ta Vivado'nun *board_files* klasörünü şurada bulabilirsin:
C:\Xilinx\Vivado\2019.2\data\boards\board_files

B. VSCode ile PlatformIO'yu Kur

Şimdi VSCode ile PlatformIO'yu kuracaksın. **Eğer bunu Hızlı Başlangıç Kılavuzu – Bölüm 1 – içerisinde yaptıysan burada yinelemene gerek yoktur, direkt Bölüm C'ye geçebilirsiniz.**

PlatformIO gömülü sistemler için Microsoft'un Visual Studio (VS) Code'unun üzerine kurgulanmış bir entegre geliştirme ortamıdır (IDE). RISC-V işlemcisini (FPGA üzerine konumlanmış olan) C ya da çevirici kullanarak programlamayı sağlar. PlatformIO çarpaz-platform olup kendi-içerisinde ayıklayıcı içerir.

VSCode ile PlatformIO'nun ikisini de kurmak için şu adımları izle:

LINUX komut-satırı: VSCode+PlatformIO kullanmak önerilen yöntem olsa da Ek A ilgilenenler için yerli RISC-V araç zinciri ile OpenOCD'yi Linux'ta indirip kurmayı, PlatformIO yerine kullanmayı açıklar. PlatformIO kullanacaksan Ek A'yı görmezden gel.

1. VSCode Kur:

VSCode kurmak için şu adımları izle:

- a. .deb dosyasını şu linkten indir:
<https://code.visualstudio.com/Download>
- b. Bir terminal aç, şunu terminalde yazarak VSCode'u kurup yürüt:

```
cd ~/Downloads
sudo dpkg -i code*.deb
code
```

Windows / macOS: VSCode paketleri Windows (.exe file) ile macOS (.zip file) için de şuradan erişilebilirdir <https://code.visualstudio.com/Download>. Bu işletim sistemlerinde bir uygulamayı kurup yürütmek için kullanılan yaygın adımları izle.

2. PlatformIO'yu VSCode üzerine Kurma:


PlatformIO'yu kurmak için şu adımları izle:

- python3 yardımcılarını şunları terminale yazarak kur:

```
sudo apt install -y python3-distutils python3-venv
```

Windows / macOS: bu adım (2.a) Windows'ta gerekmez. macOS'te python3'ü kurmak için homebrew kullanabilirsin:

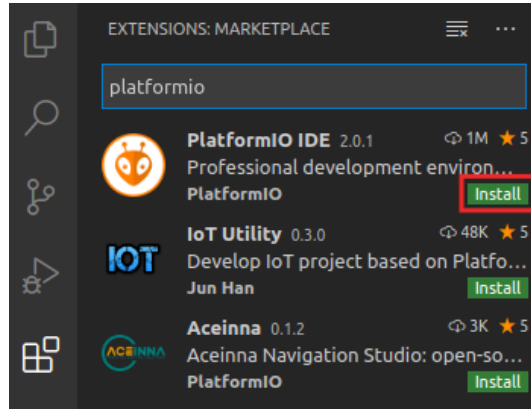
```
brew install python3
```

- Açık değilse Başlat butonunu seçip arama menüsünde "VSCode" yazıp, ardından VSCode'u seçerek ya da bir terminalde `code` yazarak VSCode'u başlat.
- VSCode'da, VSCode'un sol yan çubuğunda konumlanmış Eklentiler ikonuna  tıkla (Figür 29'a göz at).



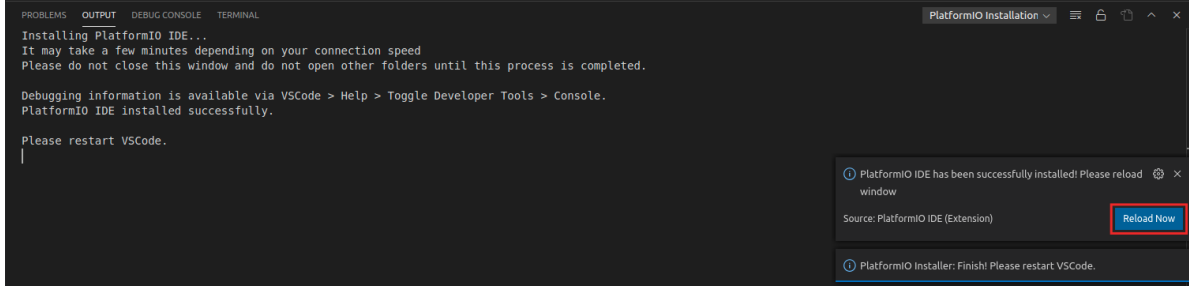
Figür 29. VSCode'un Eklentiler ikonu

- Arama kutusunda *PlatformIO* yazıp PlatformIO IDE'sini yanındaki butona tıklayarak kur (Figür 30'a göz at).



Figür 30. PlatformIO IDE Eklentisi

- e. Aşağıdaki ÇIKTI (OUTPUT) penceresi seni kurulum süreciyle ilgili bilgilendirecek. Bittiğinde pencerenin sağ altındaki “Şimdi Yenile (Reload Now)” üzerine tıkla, PlatformIO, VSCode içerisinde kurulmuş olacak (Figür 31’e göz at).



Figür 31. Şimdi Yenile, PlatformIO kurulduktan sonra

C. Verilator ile GTKWave’i Ubuntu 18.04’da Kur

Bu bölümdeki yönergeler yalnızca Linux sistemler için geçerlidir.

Windows: bu bölümdeki yönergeler yerine Ek C’yi kullan.

macOS: bu bölümdeki yönergeler yerine Ek D’yi kullan.

Verilator (yönergeler şuradadır: <https://www.veripool.org/projects/verilator/wiki/Installing> ancak aşağıda da özetlidir) ile GTKWave’i Ubuntu 18.04 Linux sistemine kurmak için şu adımları izle. Bu işlem uzun süre alır.

- `sudo apt-get install git make autoconf g++ flex bison libfl2 libfl-dev`
- `sudo apt-get install -y gtkwave`
- `git clone https://git.veripool.org/git/verilator`
- `cd verilator`
- `git pull`
- `git checkout v4.020`
- `autoconf`
- `./configure`
- `make` (hızlanması için `make -j$(nproc)` da kullanabilirsin)
- `sudo make install`

➤ `export PATH=$PATH:/usr/local/bin` (sistemdeki yolu değiştir)

/usr/local/bin'ı yoluna kalıcı olarak eklemek için son satırı `~/.bashrc` dosyana ekle.

6. RVFPGA'İ ÇALIŞTIRMA, PROGRAMLAMA

Bu bölümde RVfpga'de, Diligent Nexys A7 FPGA Kartına (Figür 24'e göz at) hedeflenip indirilmiş **Genişletilmiş SweRVolf SoC'si** (Figür 20), yedi programın nasıl çalıştırılacağını gösteriyoruz.

LINUX / Windows / macOS: Gerekli araçlarla yazılımların Bölüm 5'te açıklandığı gibi düzgün kurulduğunu varsayarsak bu bölümdeki yönergelerin hepsi üç işletim sisteminde de çalışmalıdır. Kimi yerlerde, Linux'ta ileriye eğik ya da Windows'ta geriye eğik çizgi gibi, küçük detayları değiştirmen gerekebilir.

Tablo 8'deki yedi programın nasıl çalıştırılacağını göstererek RVfpga'in nasıl kullanılacağını gösteriyoruz. İlk üç program RISC-V çevirici dilinde, son dört program C'de yazılmıştır. Program başına programı RVfpga'de çalıştırma yönlendirmeleri aşağıda tanımlanmıştır.

Tablo 8. RVfpga Örnek Programları

Program Adı	Tanım	Dil
AL_Operations	aritmetik, mantıksal işlemler yapar	RISC-V çeviricisi
Blinky	Nexys A7 kartındaki bir LED'i yakıp söndürür	RISC-V çeviricisi
LedsSwitches	Nexys A7 kartındaki anahtarların değerlerini okuyup değeri LEDlere yazar	RISC-V çeviricisi
LedsSwitches_C-Lang	Nexys A7 kartındaki anahtarların değerlerini okuyup değeri LEDlere yazar	C
HelloWorld_C-Lang	dizisel port üzerinden istemciye kısa bir mesaj yazdırır	C
VectorSorting_C-Lang	bir vektörü en büyükten en küçüğe sıralar	C
DotProduct_C-Lang	iki vektörün iç çarpımını hesaplar	C

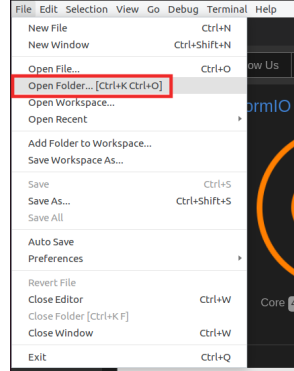
Önemli olarak, bu yedi örneği yürütebilmeden önce, **FPGA'i RVfpga SoC'si ile programlaman gerekir**, izleyen bölümde açıklandığı gibi.

A. FPGA'i RVfpga ile Programla

Bu bölümde FPGA'i RVfpga ile programlama için önerilen, PlatformIO'yu kullanan, yöntemi açıklıyoruz. FPGA'i RVfpga SoC'si ile programlama için şu adımları izle:

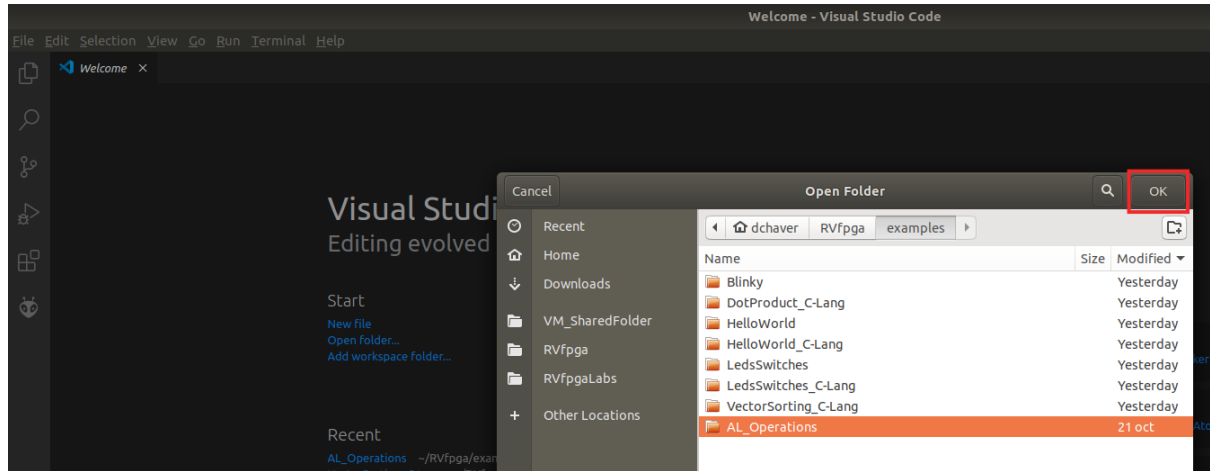
(Eğer FPGA'i programlamada Vivado'yu kullanmaya ilgilysen buradaki yönergeler yerine Ek E'deki yönergeleri izleyebilirsin. Ancak orada tanımlanan yöntem yalnızca Linux ile Windows için (macOS *değil*) çalışır – genel olarak da RVfpga'i FPGA'e indirmede Vivado'yu kullanma yöntemi önerilmez. Onun yerine bu yönergeleri izleyip Ek E'yi görmezden gelmen önerilir.)

- Nexys A7 kartını bilgisayarına bağla.
- Sol üstteki anahtarı kullanarak Nexys A7 kartını çalıştır.
- Eğer açıkta değilse VSCode ile PlatformIO'yu aç.
- Üst menü çubuğunda Dosya (File) → Klasörü Aç'a (Open Folder) (Figür 32'ye göz at) tıklayıp şu dizine git `[RVfpgaPath]/RVfpga/examples/`



Figür 32. Klasörü Aç

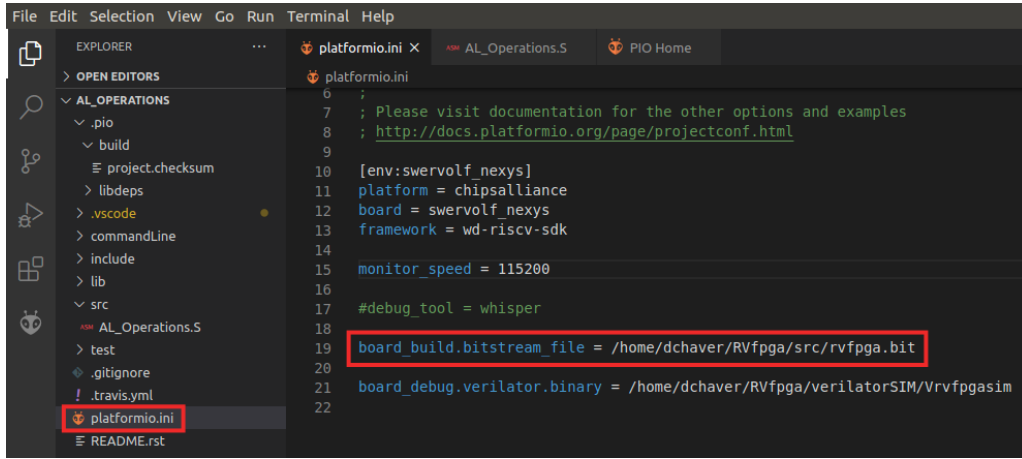
- Kullanacağın PlatformIO projesini seç. Örneğin bu bölümde Tablo 8'deki ilk örnek olan `AL_Operations`'u kullanıyoruz, ki sonraki bölümde bunu ayıklayacaksın, ancak adımlar bütün örneklerde değişmeden çalışır. Dolayısıyla `AL_Operations` dizinini seç (ancak açma, yalnızca seç – Figür 33'e göz at), pencerenin üstünde OK'a tıkla. PlatformIO şimdi örneği açacaktır.



Figür 33. `AL_Operations` klasörünü aç


- Sol yan çubukta `platformio.ini`'ye tıklayarak `platformio.ini` dosyasını aç (Figür 34'e göz at). Şu satırı değiştirerek sisteminde RVfpga veri akışına (bitstream) giden yolu belirle (Figür 34'e göz at). Önemli olarak RVfpga SoC'sinni önden-sentezlenmiş bir veri akışı (bitstream) RVfpga klasöründe şurada sağlanmıştır: `[RVfpgaPath]/RVfpga/src/rvfpga.bit`.

```
board_build.bitstream_file = [RVfpgaPath]/RVfpga/src/rvfpga.bit
```




Figür 34. Platformio ilk değerlendirme dosyası: platformio.ini

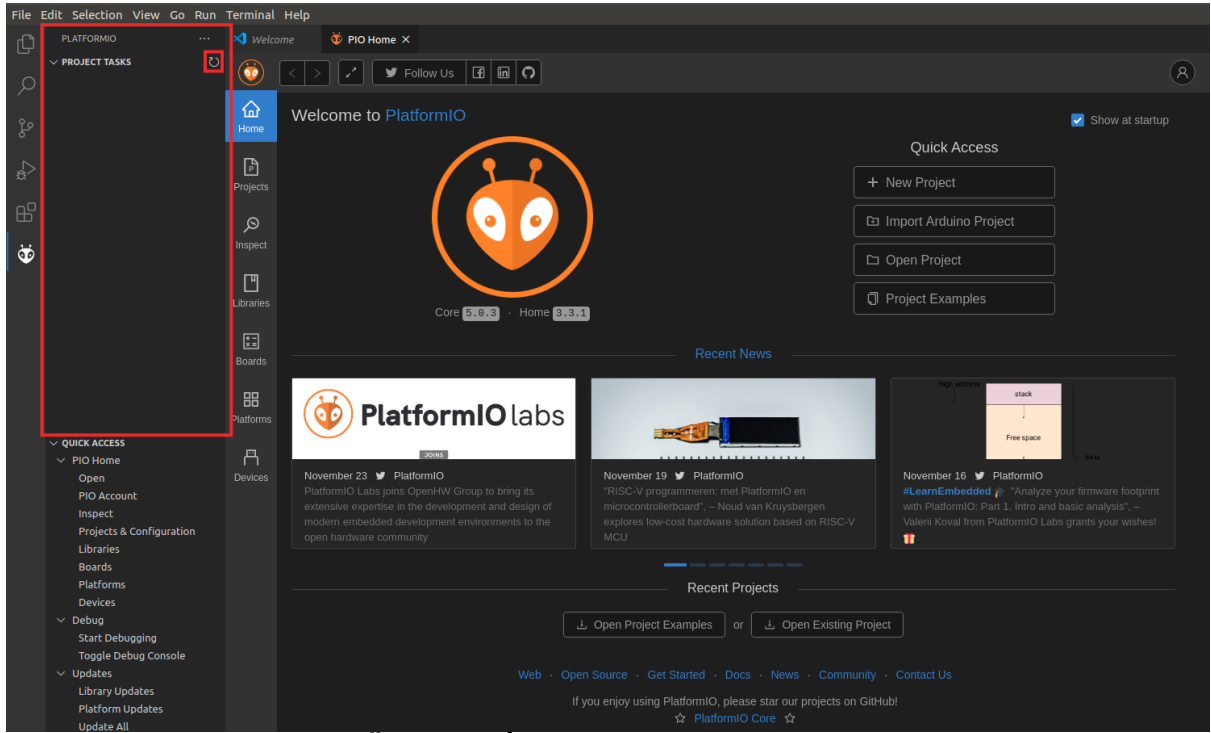
Proje Yapılandırma Dosyasında (*platformio.ini*) kullanabileceğin türlü komut bulunmakta, şuradan bilgi edinebilirsin: <https://docs.platformio.org/en/latest/projectconf/>.

- g. Sol menü şeridindeki PlatformIO ikonuna  tıkla (Figür 35'e göz at).



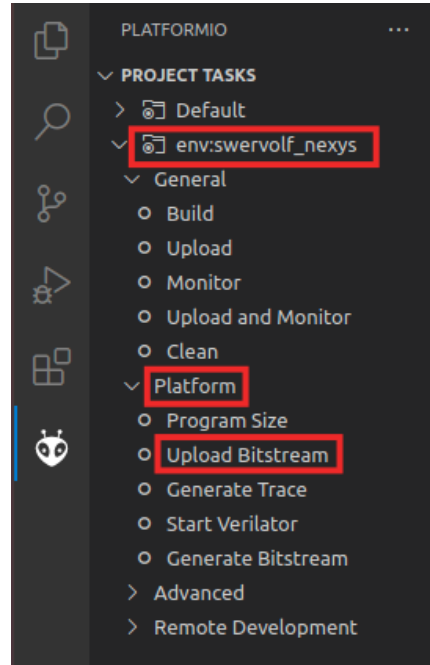
Figür 35. PlatformIO ikonu

Eğer Proje Görevleri (Project Tasks) penceresi boşsa (Figür 36),  üzerine tıklayarak Proje Görevlerini (Project Tasks) yenilemen gerek. Bu birkaç dakika alabilir.



Figür 36. PROJE GÖREVLERİ (PROJECT TASKS) penceresi boş – Yenile

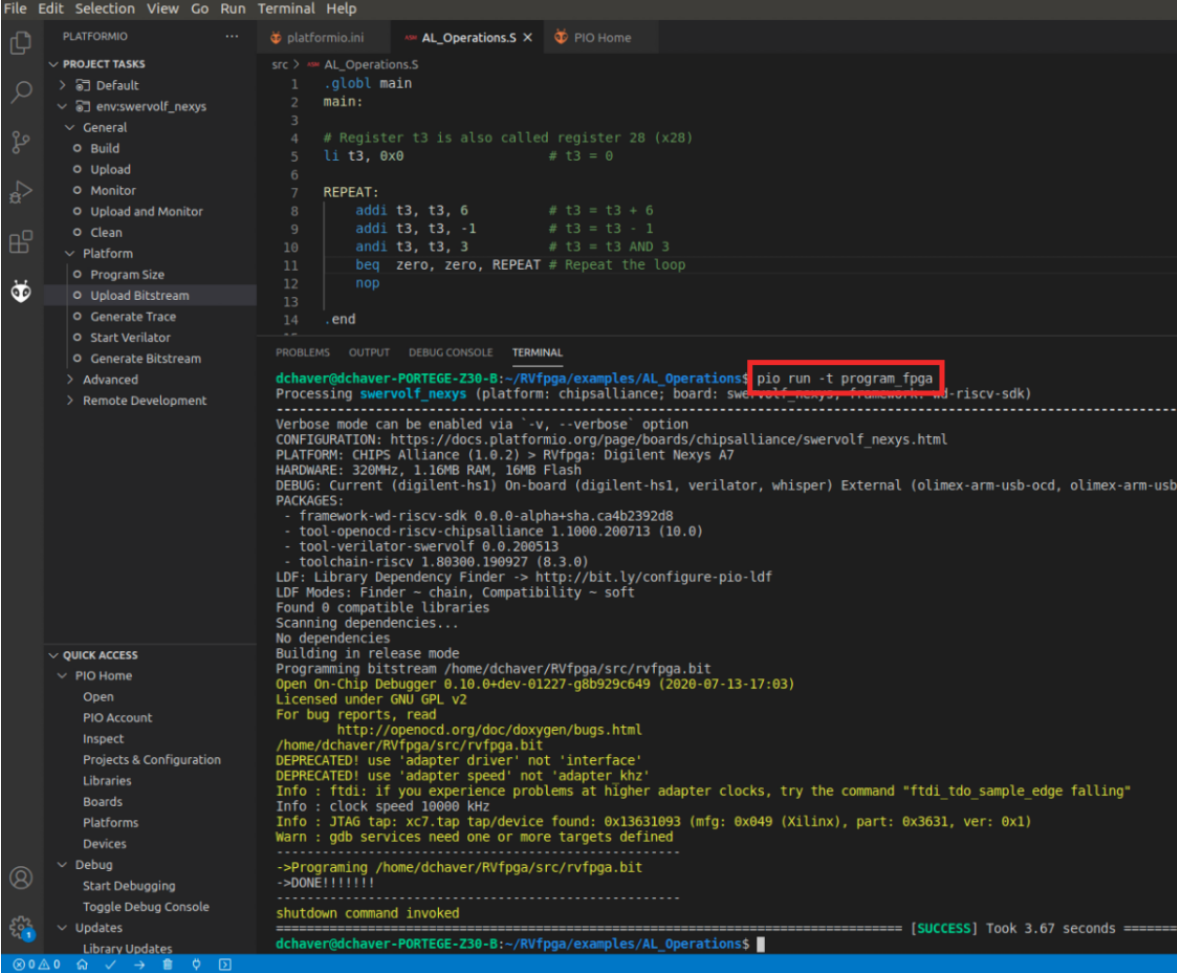
Proje Görevleri (Project Tasks) → env:swervolf_nexys → Platform genişletip Veri Akışını Yükle'ye (Upload Bitstream) tıkla, Figür 37'de gösterildiği gibi. **Bir ya da iki saniye sonra, FPGA RVfpga SoC'si ile programlanmış olacak** (karttaki 7-Kesimli Ekranlar 8 sıfır gösteriyor olmalı).



Figür 37. Veri Akışını Yükle

- h. Önceki adıma alternatif olarak (adım g), RVfpga'i PlatformIO'dan Figür 38'de gösterildiği gibi bir terminal penceresinden indirebilirsin. Yeni bir terminal penceresi açmak için

PlatformIO penceresinin aşağısındaki  butonuna (PlatformIO: Yeni Terminal (New Terminal) butonu) tıkla, ardından şu komutu PlatformIO terminaline yaz (ya da kopyala):
`pio run -t program_fpga`



```

File Edit Selection View Go Run Terminal Help
PLATFORMIO ... platformio.ini AL_Operations.S X PIO Home
PROJECT TASKS
  > Default
  > env:swervolf_nexys
  > General
    > Build
    > Upload
    > Monitor
    > Upload and Monitor
    > Clean
    > Platform
      > Program Size
      > Upload Bitstream
      > Generate Trace
      > Start Verilator
      > Generate Bitstream
      > Advanced
      > Remote Development
  > QUICK ACCESS
    > PIO Home
    > Open
    > PIO Account
    > Inspect
    > Projects & Configuration
    > Libraries
    > Boards
    > Platforms
    > Devices
  > Debug
    > Start Debugging
    > Toggle Debug Console
  > Updates
    > Library Updates

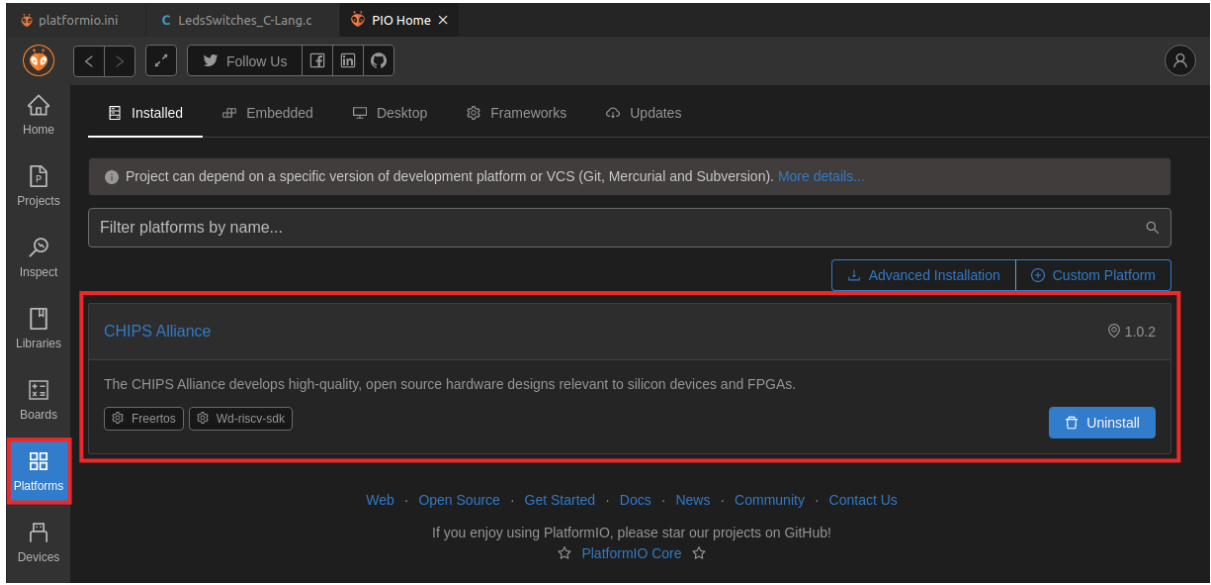
src > AL_Operations.S
1 .globl main
2 main:
3
4 # Register t3 is also called register 28 (x28)
5 li t3, 0x0 # t3 = 0
6
7 REPEAT:
8   addi t3, t3, 6 # t3 = t3 + 6
9   addi t3, t3, -1 # t3 = t3 - 1
10  andi t3, t3, 3 # t3 = t3 AND 3
11  beq zero, zero, REPEAT # Repeat the loop
12  nop
13
14 .end

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
dchaver@dchaver-ORTEGE-Z30-B:~/RVfpga/examples/AL_Operations$ pio run -t program_fpga
Processing swervolf_nexys (platform: chipsalliance; board: swervolf_nexys; framework: riscv-sdk)
Verbose mode can be enabled via '-v, --verbose' option
CONFIGURATION: https://docs.platformio.org/page/boards/chipsalliance/swervolf_nexys.html
PLATFORM: CHIPS Alliance (1.0.2) > RVfpga: Diligent Nexys A7
HARDWARE: 320MHz, 1.16MB RAM, 16MB Flash
DEBUG: Current (diligent-hs1) On-board (diligent-hs1, verilator, whisper) External (olimex-arm-usb-ocd, olimex-arm-usb-ocd)
PACKAGES:
- framework-wd-riscv-sdk 0.0.0-alpha+sha.ca4b2392d8
- tool-openocd-riscv-chipsalliance 1.1000.200713 (10.0)
- tool-verilator-swervolf 0.0.200513
- toolchain-riscv 1.80300.190927 (8.3.0)
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 0 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Programming bitstream /home/dchaver/RVfpga/src/rvfpga.bit
Open On-Chip Debugger 0.10.0+dev-01227-g8b929c649 (2020-07-13-17:03)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.org/doc/doxygen/bugs.html
/home/dchaver/RVfpga/src/rvfpga.bit
DEPRECATED! use 'adapter driver' not 'interface'
DEPRECATED! use 'adapter speed' not 'adapter khz'
Info : ftdi: if you experience problems at higher adapter clocks, try the command "ftdi_tdo_sample_edge falling"
Info : clock speed 10000 khz
Info : JTAG tap: xc7.tap tap/device found: 0x13631093 (mfg: 0x049 (Xilinx), part: 0x3631, ver: 0x1)
Warn : gdb services need one or more targets defined
->Programming /home/dchaver/RVfpga/src/rvfpga.bit
->DONE!!!!!!
shutdown command invoked
===== [SUCCESS] Took 3.67 seconds =====
dchaver@dchaver-ORTEGE-Z30-B:~/RVfpga/examples/AL_Operations$

```




Figür 38. PlatformIO'yu kullanarak RVfpga'ı Nexys A7 FPGA kartına yükle

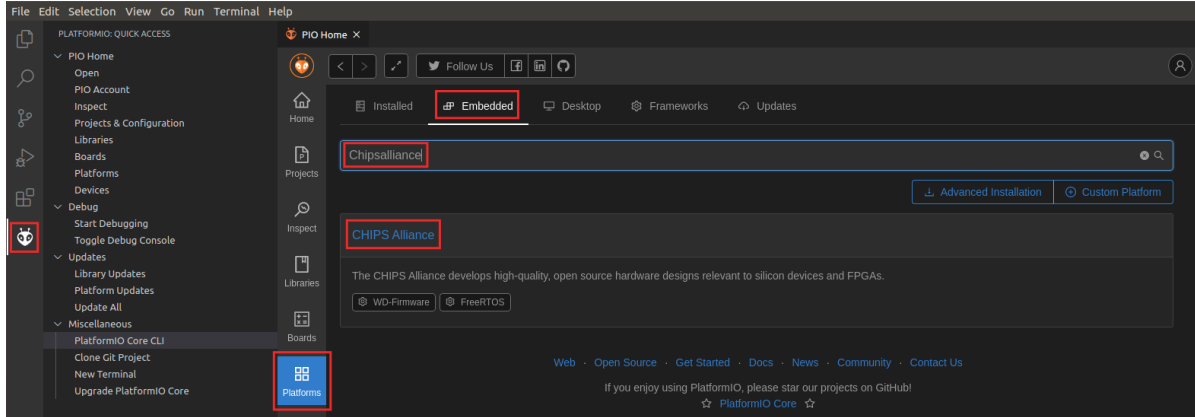
Önemli olarak bir RVfpga örneği PlatformIO'da ilk kez açıldığında Chips Alliance platformu kendiliğinden kurulur (PIO Home → Platforms içerisinde görebilirsin, Figür 39'da gösterildiği gibi). Bu platform ileride kullanacağın önceden-kurgulanmış RISC-V araç zinciri (toolchain), RISC-V için OpenOCD, RVfpga veri dosyası (bitfile) ile RVfpgaSIM, JavaScript ile Python senaryoları, örnekler gibi sonra kullanacağın araçlar içerir.




Figür 39. PlatformIO'da kurulu Chips Alliance platformu

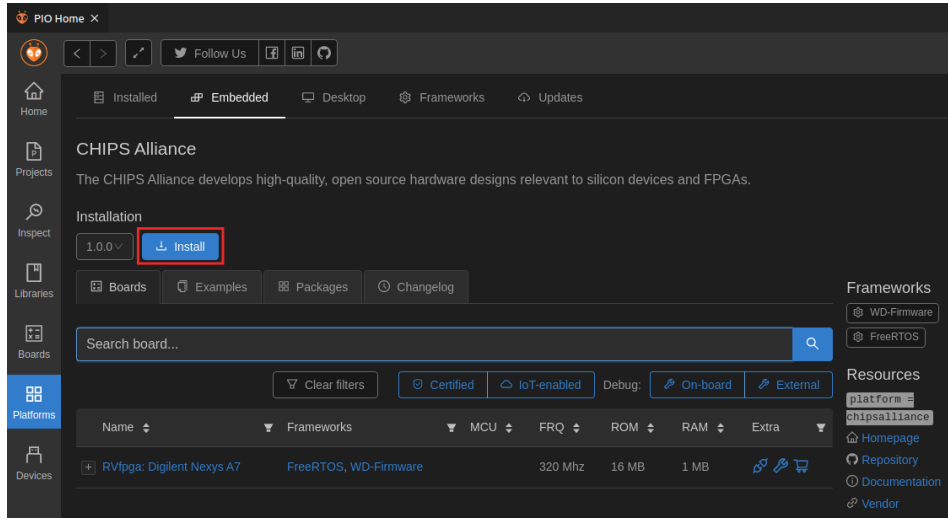
Eğer olur da Chips Alliance platformu kendiliğinden kurulmazsa şu adımları izleyerek manüel kurabilirsin (normalde bu işlemi atlayıp Bölüm B'den ilerleyebilirsin):

- Sol yan çubukta konumlanmış  butonuna tıklayarak Hızlı Erişim (Quick Access) menüsünü gör (Figür 40'a göz at). Ardından, PIO Home'da,  butonuna tıklayıp ardından  sekmesine tıkla (Figür 40). **Chipsalliance**'ı (RVfpga'de kullandığımız platform) arayıp **CHIPS Alliance** butonuna tıklayarak aç (Figür 40).




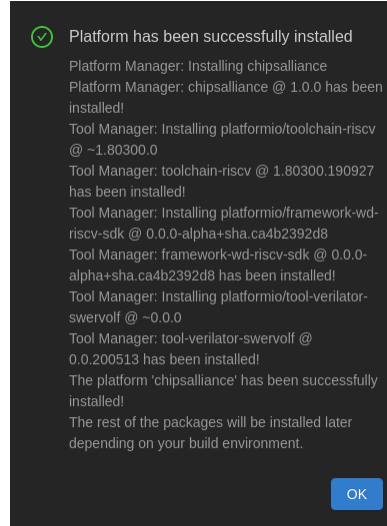
Figür 40. CHIPS Alliance Platformunu seçme

- **CHIPS Alliance** butonuna tıkladıktan sonra Chips Alliance platformunun detaylarını göreceksin (Figür 41'deki gibi).  butonuna tıklayarak kur (Figür 41).



Figür 41. CHIPS Alliance Platformunu kurma

- Kurulum bittikten sonra Figür 42'deki gibi indirilen araçların bir özeti gösterilir. O pencereyi kapatmak için  butonuna tıkla.



Figür 42. CHIPS Alliance Platformunun başarılı kurulumu

B. AL_Operations programı

İlk örnek program, AL_Operations.s (Figür 43'e göz at), sonsuz bir döngü içerisinde aynı yazmaç üzerinde, t3 (x28 de denir), üç aritmetik-mantık yönergesi (ekleme, çıkarma, mantıksal ve (and)) gerçekleştiren bir çevirici programıdır.

```

1  .globl main
2  main:
3
4  # Register t3 is also called register 28 (x28)
5  li t3, 0x0                # t3 = 0
6
7  REPEAT:
8      addi t3, t3, 6          # t3 = t3 + 6
9      addi t3, t3, -1         # t3 = t3 - 1
10     andi t3, t3, 3           # t3 = t3 AND 3
11     beq zero, zero, REPEAT  # Repeat the loop


```

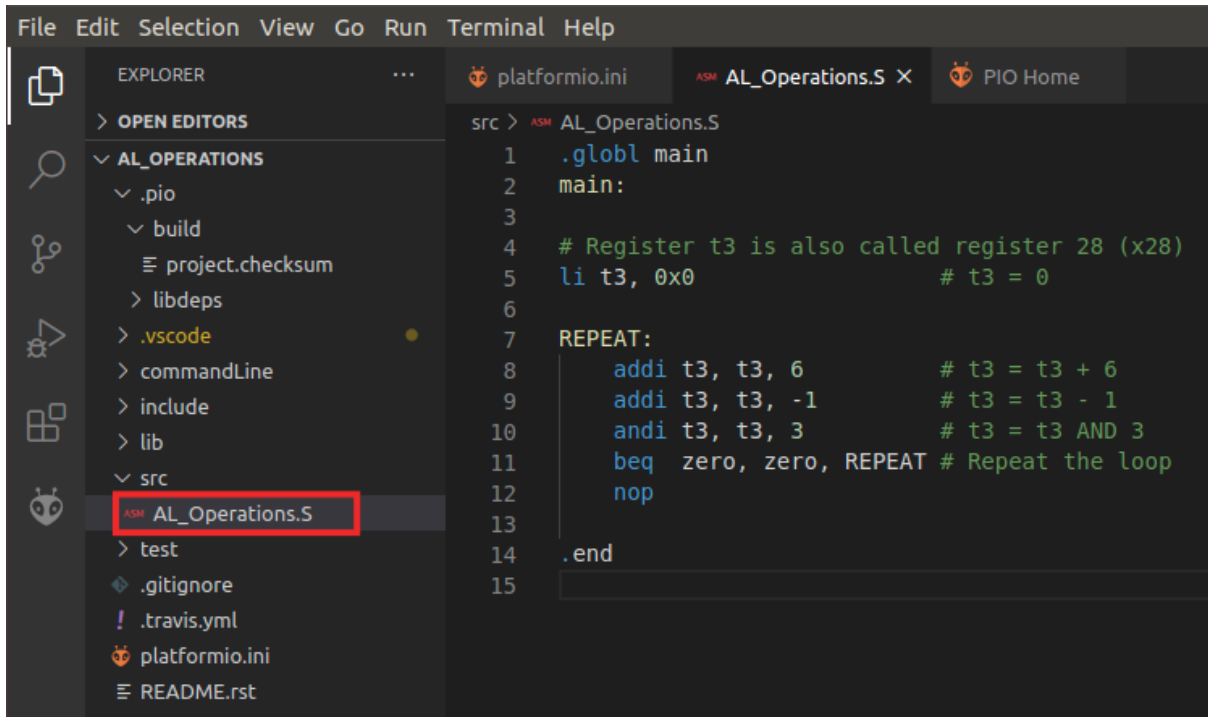
```
12    nop
13
14    .end
```

Figür 43. AL_Operations programı: AL_Operations.S

PlatformIO'yu kullanarak bu kodu çalıştırıp ayıklamak için şu adımları izle:

1. FPGA'i önceki bölümde açıklandığı gibi programla. Önemli olarak *AL_Operations* projesi PlatformIO'da açıkta olmalı.
2. Çevirici programını, *AL_Operations.S*, sol şerit menüdeki Explorer (Gezgin) ikonuna



üzerine tıklayıp , sol yan çubukta *AL_OPERATIONS* altındaki *src*'yi genişletip, *AL_Operations.S* üzerine tıklayarak (Figür 44'e göz at) aç.



Figür 44. AL_Operations.S çevirici dosyasına bak


3. VSCode ile PlatformIO, programı derleme, temizleme, ayıklama için değişik yollar sağlar. VSCode'un aşağısında kullanışlı işlevler sağlayan butonlar bulabilirsiniz:

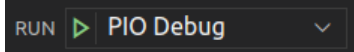


Örneğin,  projeyi kurgulamak için kullanılabilir, ya da  temizlemek için kullanılabilir. Sol yan çubukta (Figür 29'a göz at), "Run (Çalıştır)"



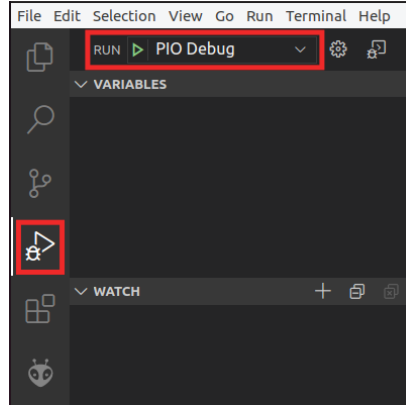
butonu programı derleyip ardından ayıklayıcıyı açmak için kullanılabilir.

4. "Run (Çalıştır)" butonuna  tıkla. Oynat butonuna tıklayarak ayıklayıcıyı başlat



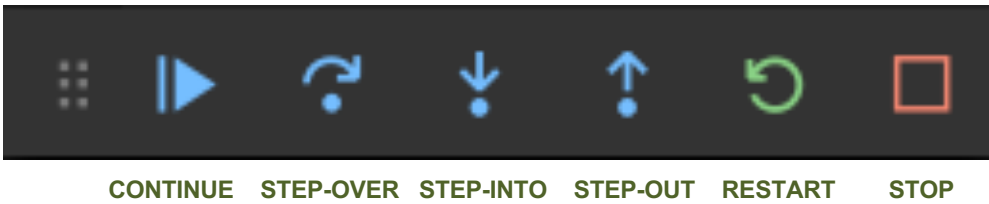
("PIO Debug" seçeneğinin seçilmiş olduğunun sağlanmasını yap). Bu butonu pencerenin üstünde bulabilirsin (Figür 45'e göz at). Program ilk

derlenecek, ardından ayıklama başlayacak. PlatformIO main (ana) işlevinin başlangıcında bir geçici kesme noktası ayarlar, yani yürütme orada duracaktır.



Figür 45. Ayıklayıcıyı başlat

5. Ayıklama oturumunu denetlemek için editörün üstünün yakınlarında görünen ayıklama araç çubuğunu kullanabilirsin (Figür 46'ya göz at). Seçenekler şunlardır:
 - **Continue** (Sürdür) programı sonraki kesme noktasına dek yürütür.
 - **Breakpoints** (Kesme Noktaları) editörde satır numarasının soluna tıklayarak eklenebilir.
 - **Step Over** (Üzeri Adım) şimdiki satırı yürütüp ardından durur.
 - **Step Into** (İçeri Adım) şimdiki satırı yürütür, eğer şimdiki satırda bir işlev çağırısı varsa o işleve sıçrayıp ardından durur.
 - **Step Out** (Dışarı Adım) içinde bulunduğun işlevdeki bütün kodları yürütüp ardından işlevi dönüş yaptığında durur.
 - **Restart** (Yeniden Başlat) ayıklama oturumunu programın başından yeniden başlatır.
 - **Stop** (Durdur) ayıklama oturumunu durdurup normal düzenleme moduna döner. Önemli olarak Stop (Durdur) butonuna bastığında **program RVfpga'de çalışmayı sürdürür** ancak ayıklama oturumu sona erer.
 - **Pause** (Beklet) yürütmeyi bekletir. Program çalışırken Continue (Sürdür) butonu Pause (Beklet) butonuyla yer değiştirir.




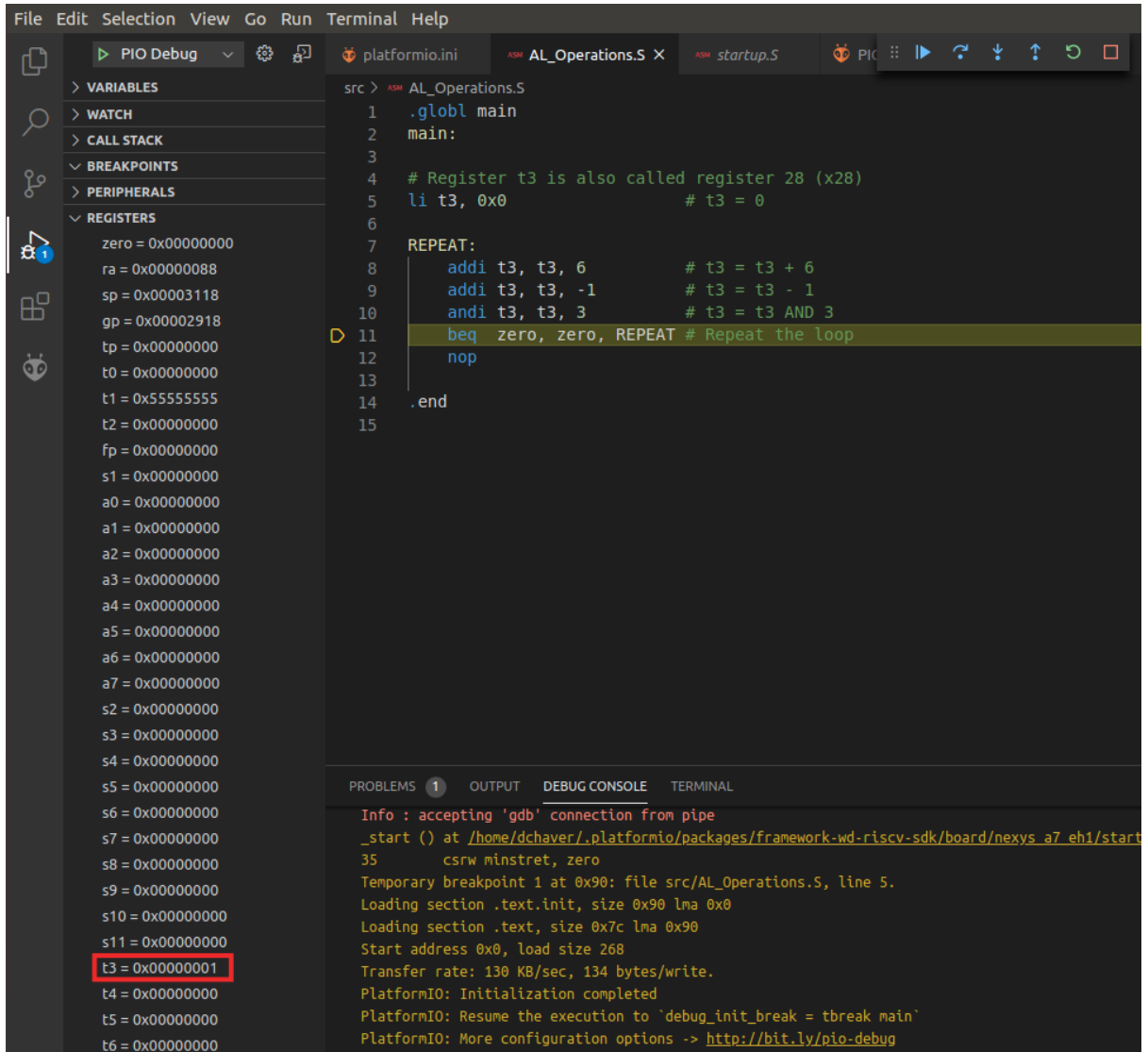
Figür 46. Ayıklama araçları

6. Sol yan çubukta, Debugger (Ayıklayıcı) seçeneklerini görebilirsin. Şu seçenekler vardır:
 - **Variables** (Değişkenler): programında bulunan yerel, genel, durgun değişkenleri değerleriyle listeler.
 - **Call Stack** (Çağrı Yığını): sana şimdiki çalışan işlevi, (varsa) çağırılan işlevi, şimdiki yönergenin bellekteki yerini gösterir.
 - **Breakpoints** (Kesme Noktaları): ayarlanmış kesme noktalarını gösterip satır numaralarını parlatır. Kesme noktaları bu bölümde yönetilebilir. Ayrıca kesme

- noktaların işaret kutusu değiştirilerek geçici olarak etkinliği kaldırılabilir.
- **Peripherals** (Çevre Birimleri): aygıtın bellek-eşlenmiş çevre birimlerinin yazmaçlarının durumunu gösterir.
- **Registers** (Yazmaçlar): işlemcinin yazmaçlarının hepsinin şimdiki değerlerini listeler.
- **Memory** (Bellek): bellekteki belirli bir adresin içeriğini gösterir.
- **Disassembly** (Tersine Çeviri): belirli bir işlevin çevirici kodunu gösterir – bu C gibi yüksek-düzeyle dillerde yönergeleri bir bir ayıklayabilmek için çeviriciyi görmeni sağlar.

7. Ayıklayıcı Yan Çubuğundaki (Debugger Side Bar) Registers (Yazmaçlar) seçeneğini

genişletip yürütmeyi adım-adım sürdür . x28 yazmacının (t3 de denir, REGISTERS (YAZMAÇLAR) bölümünde gösterildiği gibi) üç aritmetik-mantık işleminin sonuçlarını depoladığını gözlemleyeceksin: *ekleme*, *çıkarma*, *mantıksal AND*. Figür 47'ye göz at.



Figür 47. Yazmaç içeriklerine bakma

8. *main* işlevini çağırmadan önce, ~/.platformio/packages/framework-wd-riscv-


`sdk/board/nexys_a7_eh1/startup.S` içerisinde Western Digital'in sağladığı, bir başlangıç dosyası yürütülür. Bu dosya çekirdeği yapılandırır: Yönerge Ön belleği (Instruction Cache) ayarlanır, yazmaçların ilk değerlendirilmesi (*sp* ile *gp* gibi), gibi. Ayıklama başlatıldığında bu dosya ana pencerede açılır (Figür 47'ye göz at), oradan inceleyebilirsin.

Windows: `.platformio` klasörü kullanıcı klasörünün içinde konumlanmıştır (`C:\Users\<USER>`). Önemli olarak gizli (saklı) dosyaları/klasörleri görmek için sistemde ayar yapman gerekebilir.

macOS: Linux gibi, `.platformio` klasörü ev klasörüne konumlanmıştır (`~/platformio`).

9. Bu dizinde (`~/platformio/packages/framework-wd-riscv-sdk/board`) bütün projelerimizde kullandığımız bağlayıcı senaryosunu (linker script) oluşturan `link.lds` dosyası sağlanmıştır. Bu dosya bellekteki çevirici bölümlerinin yerleşimini belirler (*text (metin)*, *data (veri)*, *bss (blok başlangıç sembolü)*...).

10. Son olarak, ayıklamayı durdurup  en soldaki yan menünün

üstünde bulabileceğin  üzerine tıklayarak Gezgin (Explorer) penceresine dön. Üst menü çubuğunda *Dosya (File)* → *Klasörü Kapat (Close Folder)* üzerine tıkla.

C. Blinky programı

İkinci örnek program, `blinky.S`, Nexys A7 kartının en sağdaki LEDlerini yakıp söndüren bir çevirici programıdır (Figür 48'e göz at). Bu program tersine döndürmeler arasında gecikmeyle en sağdaki LEDe bağlı değeri tekrar tekrar tersine döndürür.

```

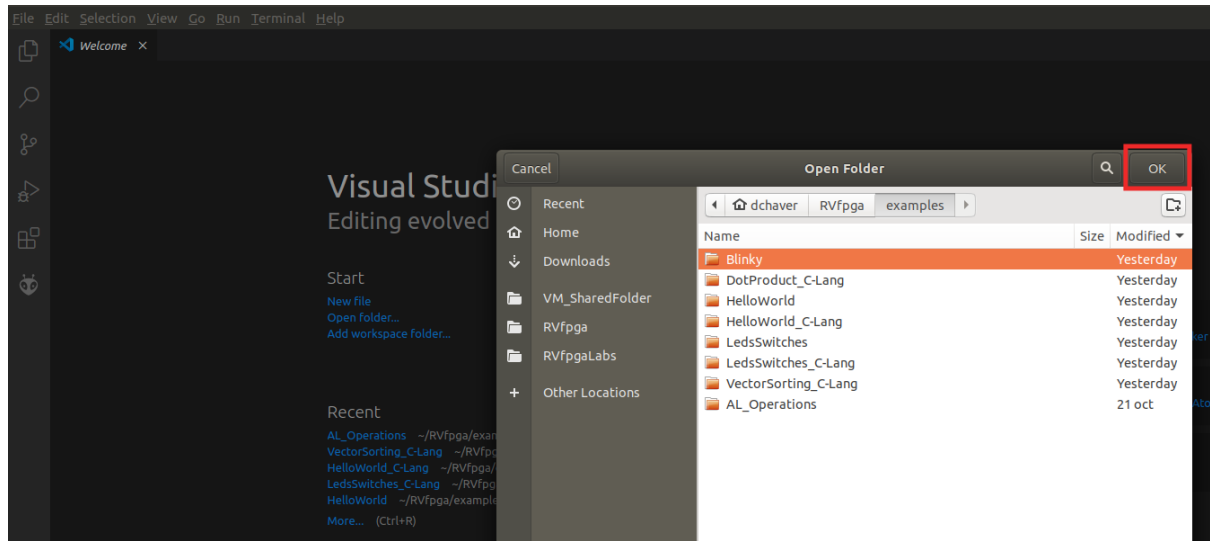
1  #define GPIO_LEDS    0x80001404
2  #define GPIO_INOUT   0x80001408
3
4  #define DELAY 0x100000          /* Define the DELAY */
5
6  .globl main
7  main:
8
9      li x28, 0xFFFF
10     li a0, GPIO_INOUT
11     sw x28, 0(a0)              # Write the Enable Register
12
13     li  t1, DELAY              # Set timer value to control blink speed
14
15     li  t0, 0
16
17 b11:
18     li  a0, GPIO_LEDS
19     sb  t0, 0(a0)              # Write to LEDs
20     xori t0, t0, 1             # invert LED
21     and t2, zero, zero        # Reset timer
22
23 time1:                          # Delay loop
24     addi t2, t2, 1
25     bne t1, t2, time1
26     j   b11

```

Figür 48. blinky.S

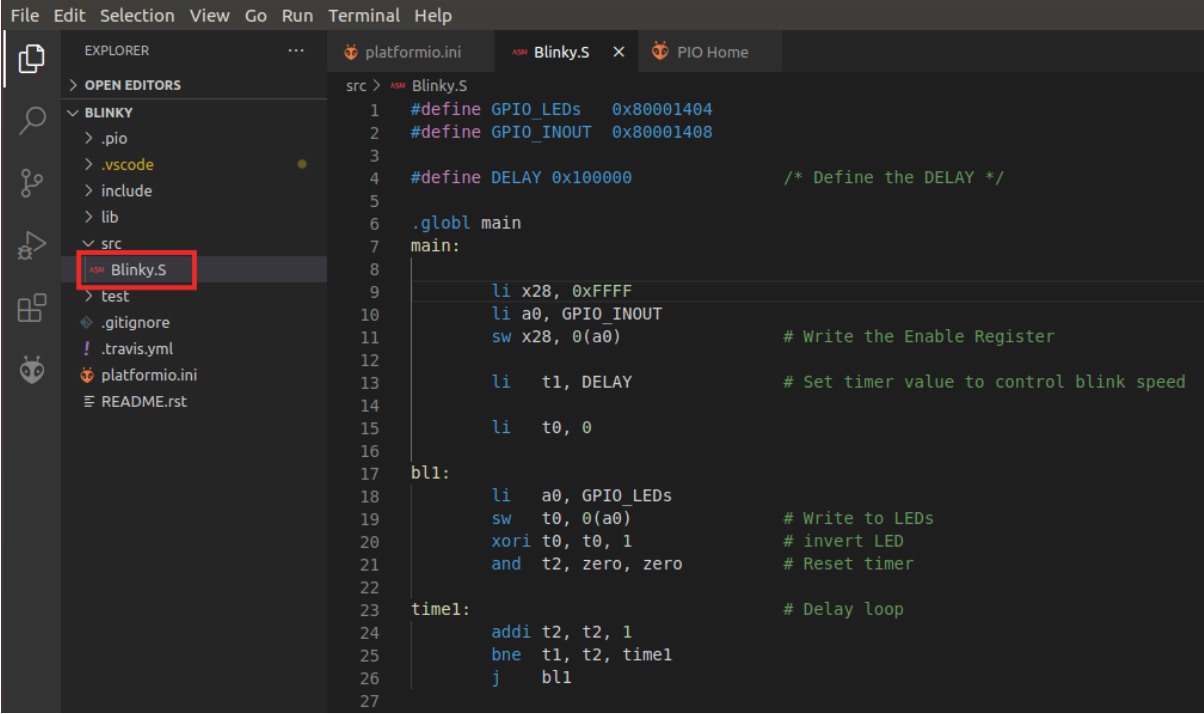
RVfpga’de, FPGA kartına yüklenmiş RISC-V SoC’si, bu kodu çalıştırıp ayıklamak için şu adımları izle:

1. İlk örneği (*AL_Operations*) yürüttüysen RVfpga FPGA kartında programlanmıştır, yani yeniden programlaman gerekmemeli. Ancak RVfpga’i karta yeniden programlaman gerekiyorsa Bölüm A’da açıklandığı gibi *AL_Operations* örneği yerine *Blinky* örneğini kullanarak yap.
2. Üst çubukta, *File (Dosya) → Open Folder (Klasörü Aç)*, üzerine tıklayıp şu dizine git *[RVfpgaPath]/RVfpga/examples/*



Figür 49. Blinky program klasörü

3. *Blinky* dizinini seçip ardından OK’a tıkla.
4. Örneğin çevirici kodunu, dosya *blinky.S*, editörde üzerine tıklayarak aç (Figür 50).


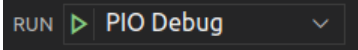




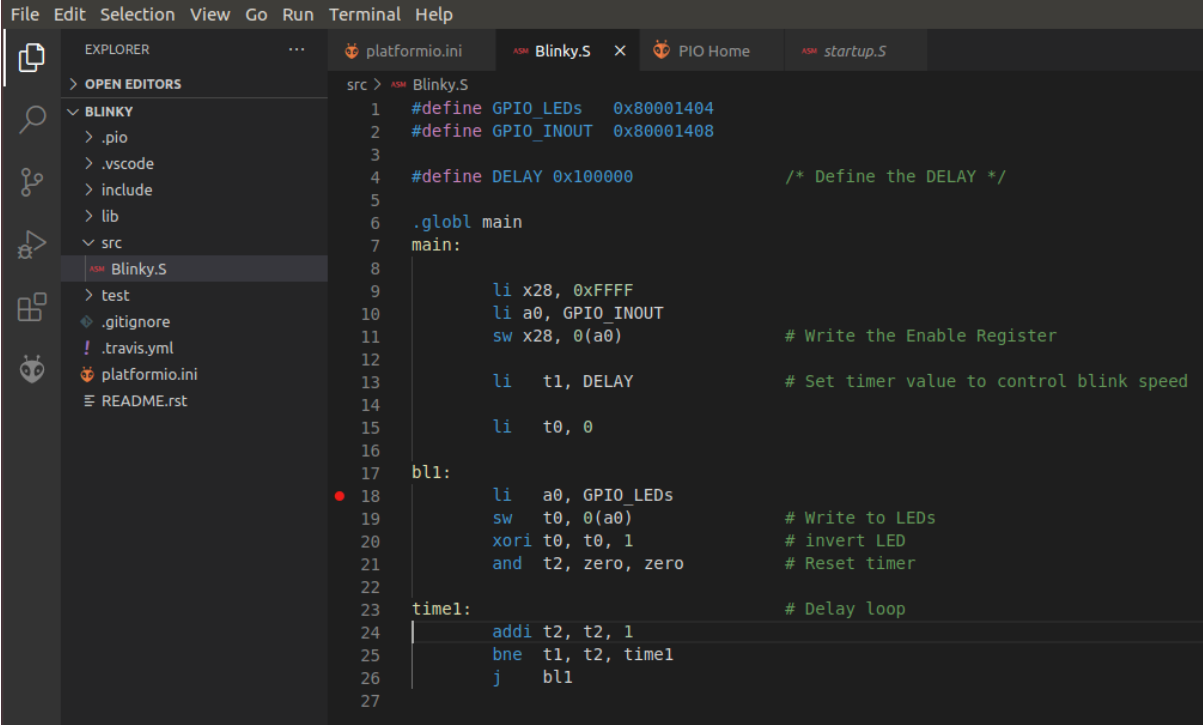
```

src > Blinky.S
1  #define GPIO_LEDS    0x80001404
2  #define GPIO_INOUT   0x80001408
3
4  #define DELAY 0x100000 /* Define the DELAY */
5
6  .globl main
7  main:
8
9      li x28, 0xFFFF
10     li a0, GPIO_INOUT
11     sw x28, 0(a0)      # Write the Enable Register
12
13     li t1, DELAY       # Set timer value to control blink speed
14
15     li t0, 0
16
17 bl1:
18     li a0, GPIO_LEDS
19     sw t0, 0(a0)       # Write to LEDs
20     xori t0, t0, 1     # invert LED
21     and t2, zero, zero # Reset timer
22
23 timel:
24     addi t2, t2, 1     # Delay loop
25     bne t1, t2, timel
26     j bl1
27

```

Figür 50. PlatformIO'da blinky.S

5. Programı çalıştırıp ayıklamak için  üzerine tıkla; ardından oynat butonuna  tıklayarak ayıklamayı başlat. PlatformIO main işlevinin başında bir kesme noktası ayarlar. Yani programı çalıştırmak için Sürdür butonuna  tıkla.
6. Kartın üstünde en sağdaki LED'in yanıp sönmeye başladığını göreceksin.
7. Beklet butonuna tıklayarak yürütmeyi bekle . Yürütme sonsuz döngü içerisinde bir yerde duracaktır (yüksek olasılıkla, `timel` geciktirme döngüsünün içerisinde).
8. Satır numarası 18'in soluna tıklayarak bir kesme noktası belirle. Bir kırmızı nokta belirip kesme noktası BREAKPOINTS (KESME NOKTALARI) sekmesine eklenecektir (Figür 51'e göz at).




```

1  #define GPIO_LEDs    0x80001404
2  #define GPIO_INOUT   0x80001408
3
4  #define DELAY 0x100000    /* Define the DELAY */
5
6  .globl main
7  main:
8
9      li x28, 0xFFFF
10     li a0, GPIO_INOUT
11     sw x28, 0(a0)        # Write the Enable Register
12
13     li t1, DELAY         # Set timer value to control blink speed
14
15     li t0, 0
16
17 bl1:
18     li a0, GPIO_LEDs
19     sw t0, 0(a0)         # Write to LEDs
20     xori t0, t0, 1       # invert LED
21     and t2, zero, zero   # Reset timer
22
23 timel:                   # Delay loop
24     addi t2, t2, 1
25     bne t1, t2, timel
26     j bl1
27

```

Figür 51. blinky.S’de bir kesme noktası ayarlama

9. Ardından yürütmeyi Sürdür butonuna tıklayarak sürdür . Yürütme sürdürülüp bayt depola (sb) yönergesinden, en sağdaki LED’e 1 (ya da 0) yazan, sonra duracaktır.

10. Yürütmeyi birkaç kez sürdür; en sağdaki LED’e yazılan değer hep değiştiğini göreceksin.

11. Ayıklamayı durdur , ardından  üzerine tıklayarak Gezgin penceresine dön. *File (Dosya) → Close Folder (Klasörü Kapat)* seçerek programı kapat.

D. LedsSwitches programı

Üçüncü çevirici örneği karttaki LEDlerle, anahtarlarla iletişim kurar (Figür 52’ye göz at).

```

1  #define GPIO_SWs     0x80001400
2  #define GPIO_LEDs    0x80001404
3  #define GPIO_INOUT   0x80001408
4
5  .globl main
6  main:
7
8  li x28, 0xFFFF
9  li x29, GPIO_INOUT
10 sw x28, 0(x29)        # Write the Enable Register
11
12 next:
13     li a1, GPIO_SWs    # Read the Switches
14     lw t0, 0(a1)
15
16     li a0, GPIO_LEDs

```

```

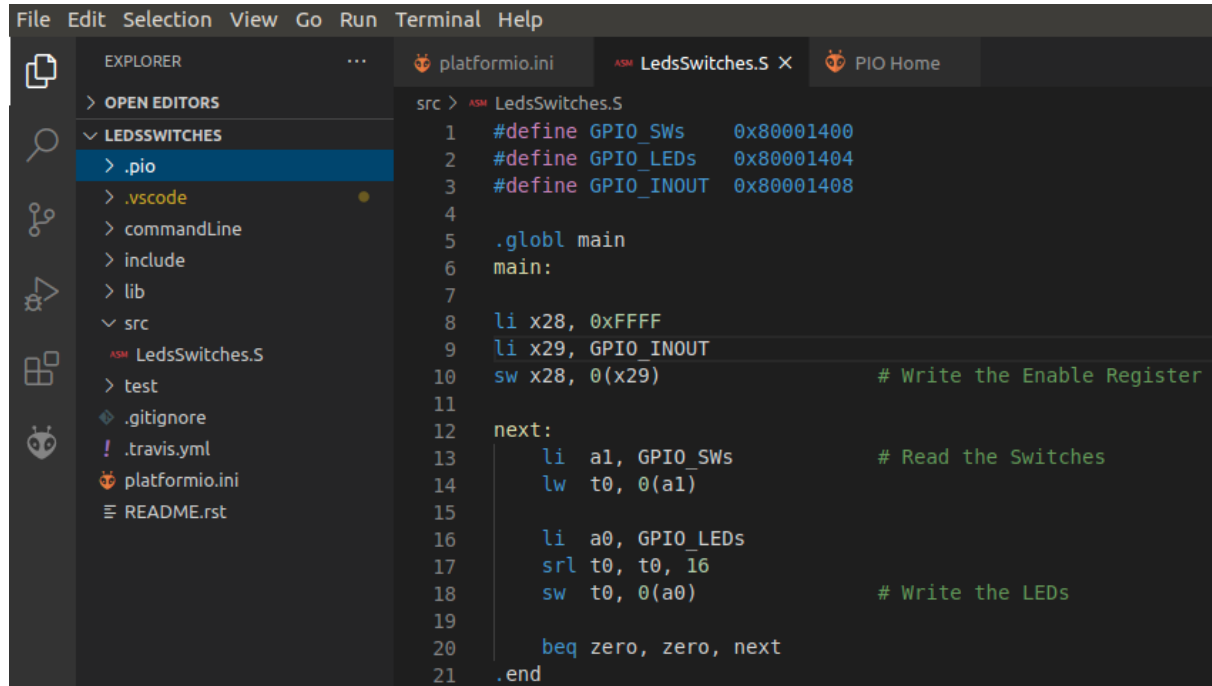
17    srl t0, t0, 16
18    sw t0, 0(a0)           # Write the LEDs
19
20    beq zero, zero, next
21 .end

```


Figür 52. LedsSwitches.S

Bu kodu FPGA kartında çalıştırıp ayıklamak için şu adımları izle:

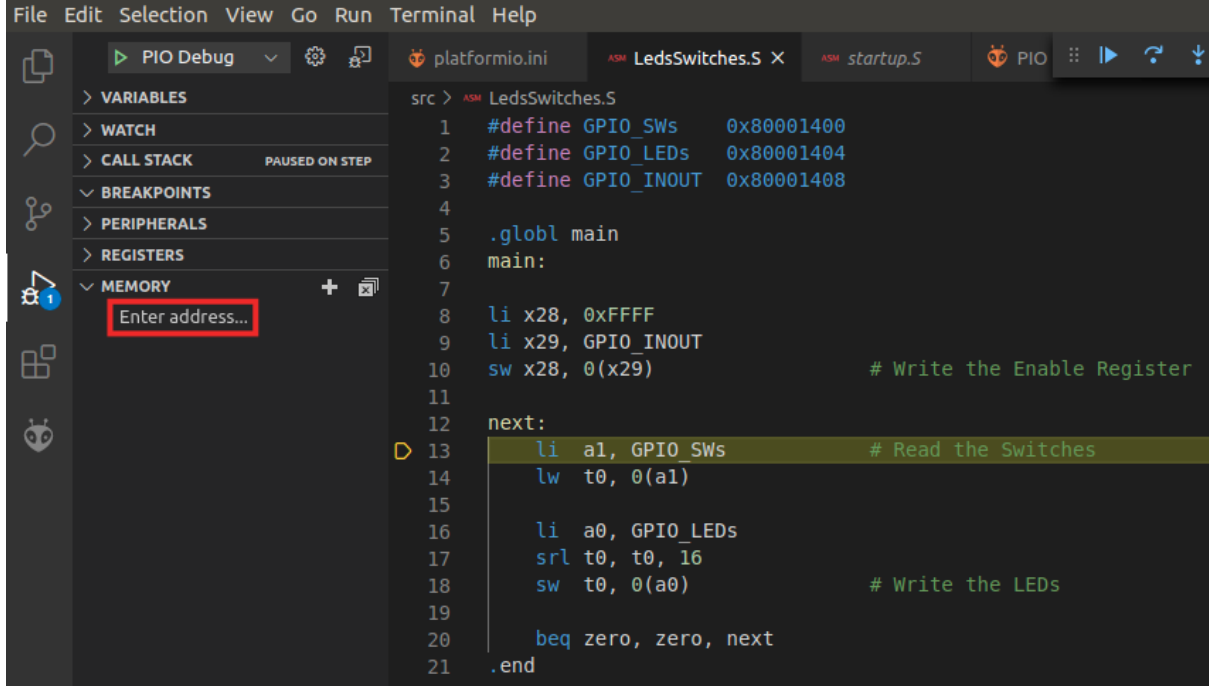
1. Önceki örnekleri yürüttüysen RVfpga FPGA kartında programlanmıştır, yani yeniden programlaman gerekmemeli. Ancak RVfpga'i karta yeniden programlaman gerekiyorsa Bölüm A'da açıklandığı gibi AL_Operations örneği yerine LedsSwitches örneğini kullanarak yap.
2. Üst çubukta *File (Dosya) → Open Folder (Klasörü Aç)* üzerine tıklayıp şu dizine git *[RVfpgaPath]/RVfpga/examples/*. *LedsSwitches* dizinini seçip ardından OK'a tıkla.
3. Program *LedsSwitches.S*'de anahtarların okunup durumlarının LEDlerde gösterildiği bir sonsuz döngü vardır.



Figür 53. PlatformIO'da LedsSwitches.S

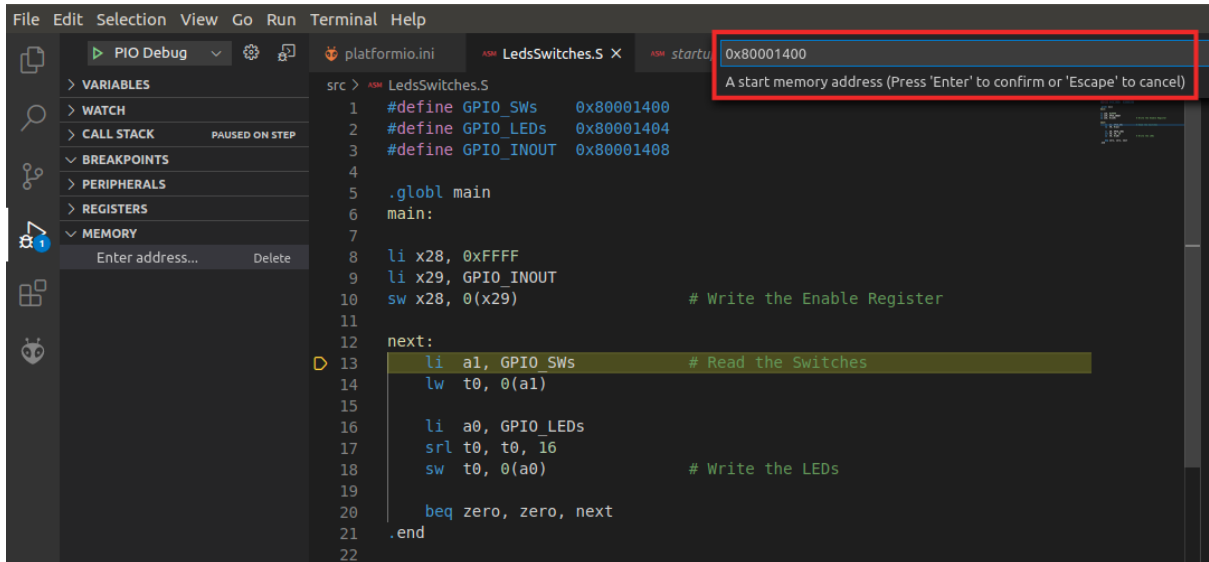
4. Ayıklayıcıyı önceki programlarda açıklandığı gibi başlattıktan sonra program çalışmaya başlar. PlatformIO main işlevinin sonrasına geçici bir kesme noktası ayarlar. Yani programı çalıştırmak için **Sürdür** butonuna  tıkla.
5. Nexys A7 kartının aşağısındaki anahtarlara bas. Anında karttaki LEDlerin anahtarların yeni değerlerini gösterdiğini göreceksin. Yukarıda açıklandığı gibi yürütmeyi bekletebilirsin, adım-adım çalıştırabilirsin, yazmaçları inceleyebilirsin. Bitirdiğinde projeyi *File (Dosya) → Close Folder (Klasörü Kapat)* tıklayarak kapat.
6. Ara sıra bellekte depolanan değerleri incelemek yararlı olabilir. Bunun için PlatformIO bir Bellek Ekranı sağlar.

- a. Yürütmeyi bekletip *sonraki* döngünün başına değin adımla. Pencerenin solundaki Memory Display (Bellek Ekranı) genişlet (Figür 54'e göz at), *Enter address...* (*Adres gir...*) tıkla



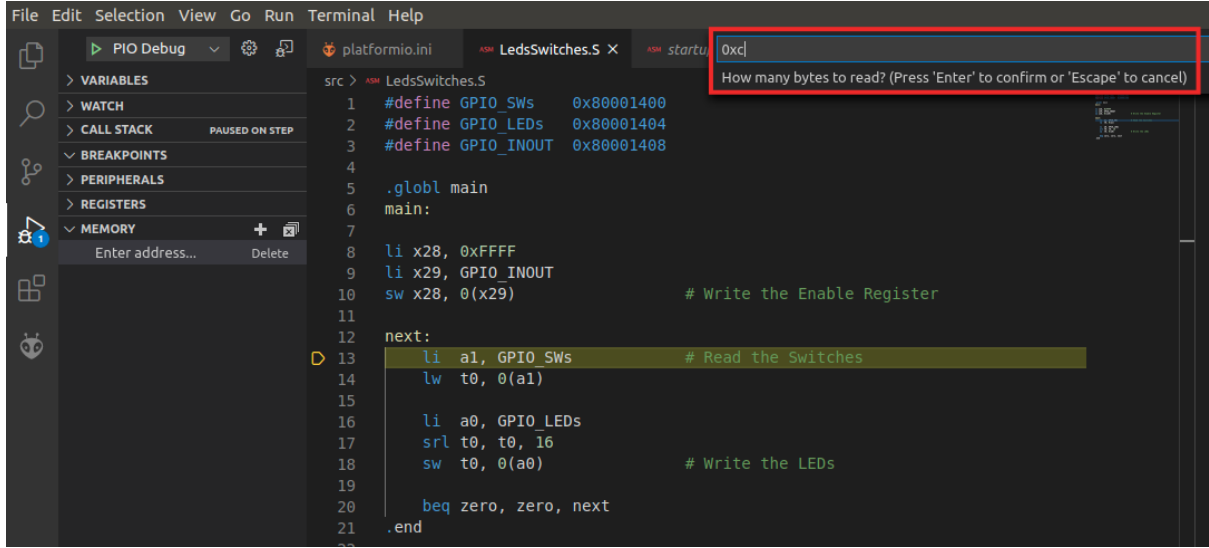
Figür 54. Bellek Ekranı

- b. İlk bellek adresi istenecektir (Figür 55'e göz at). Anahtarların eşlendiği, burada 0x80001400, ilk adresi yerleştir.



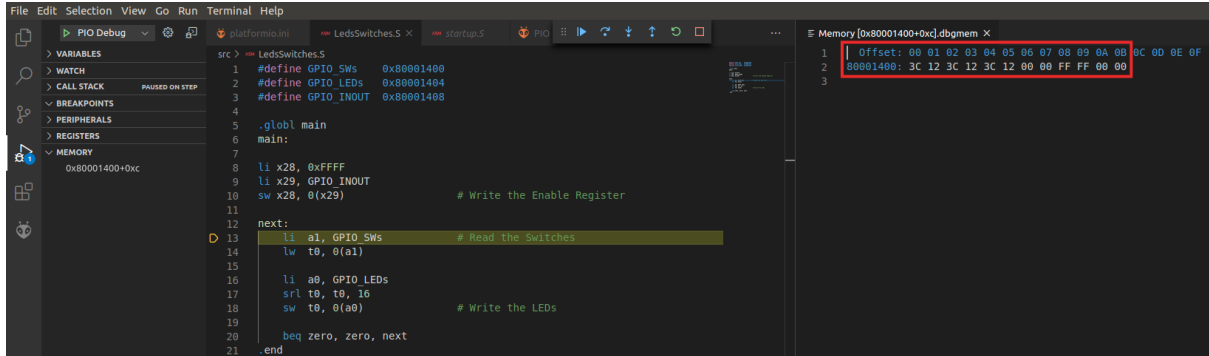
Figür 55. Gösterilecek ilk bellek adresi

- c. Ardından incelemek istediği bayt sayısı istenir (Figür 56'ya göz at), 0xc değerini yerleştir (üç 4-bayt I/O yazmacını incelemek istiyoruz, dolayısıyla 12 bayta gerek duyuyoruz).



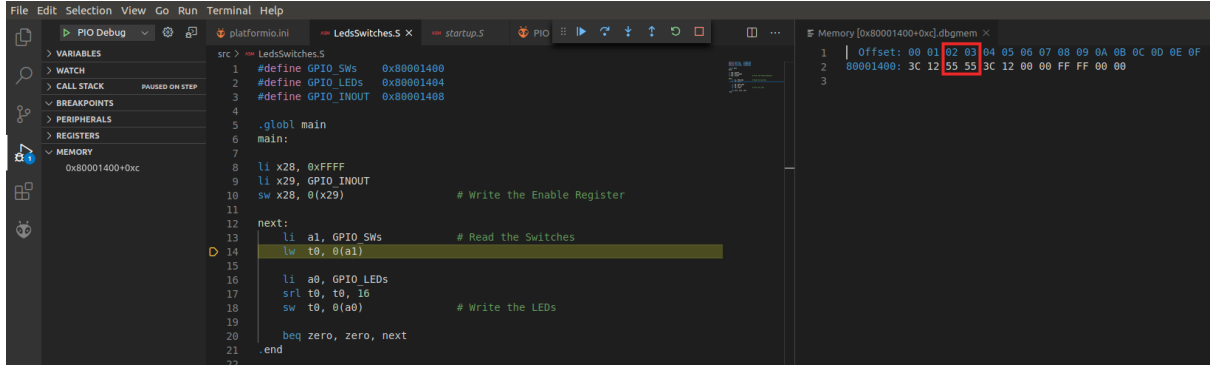
Figür 56. Gösterilecek bayt sayısı

- d. Bellek Ekranı (Memory Display) istenilen 12 baytı göstererek (Figür 57'ye göz at) sağa açılacaktır. 16 anahtardaki değerimiz 0x123C (0x80001402 ile 0x80001403 adreslerindeki baytlara bak). RISC-V mimarisinin düşük son haneli olduğunu göz önünde bulundurursak figürde gösterilen adres bununla uyumludur. 16 LEDin (0x80001404 ile 0x80001405 adreslerinde depolanır) değerleri bunlarla aynı olmalı.



Figür 57. 0x80001400-0x8000140B bellek adresleri

- e. Karttaki anahtarların değerini değiştir, örneğin 0x5555'e, ardından döngüyü bir yineleme daha adım-adım yürüt. Anahtarların değerleri bellekte ilk yönergeyi yürüttükten sonra anında değişirken (Figür 58, üst), LEDlerin değeri de uygun olarak sw yönergesini yürüttükten sonra değişmeli (Figür 58, aşağı).



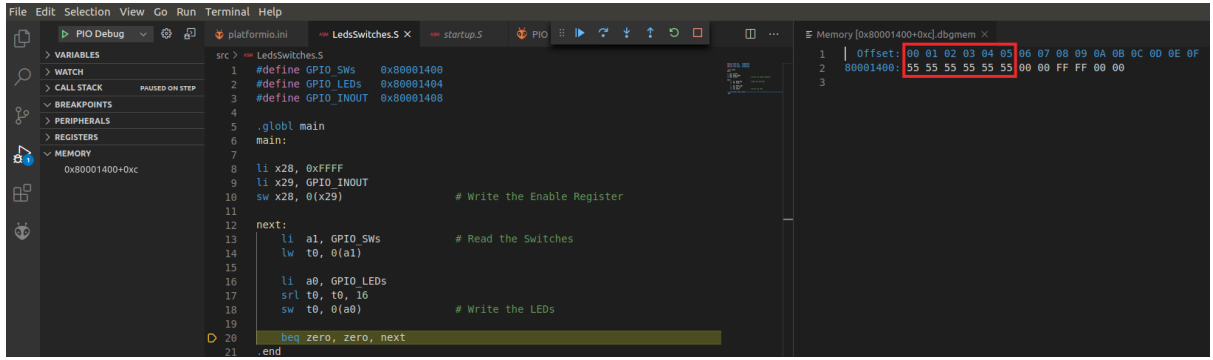
```

src > LedsSwitches.S
1 #define GPIO_SWS 0x80001400
2 #define GPIO_LEDS 0x80001404
3 #define GPIO_INOUT 0x80001408
4
5 .globl main
6 main:
7
8 li x28, 0xFFFF
9 li x29, GPIO_INOUT
10 sw x28, 0(x29)          # Write the Enable Register
11
12 next:
13 li a1, GPIO_SWS        # Read the Switches
14 lw t0, 0(a1)
15
16 li a0, GPIO_LEDS
17 srl t0, t0, 16
18 sw t0, 0(a0)           # Write the LEDs
19
20 beq zero, zero, next
21 .end

```

Memory [0x80001400+0xc].dbgmem X

1	Offset: 00 01 02 03	04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
2	80001400: 3C 12 55 55	3C 12 00 00 FF FF 00 00
3		



```

src > LedsSwitches.S
1 #define GPIO_SWS 0x80001400
2 #define GPIO_LEDS 0x80001404
3 #define GPIO_INOUT 0x80001408
4
5 .globl main
6 main:
7
8 li x28, 0xFFFF
9 li x29, GPIO_INOUT
10 sw x28, 0(x29)          # Write the Enable Register
11
12 next:
13 li a1, GPIO_SWS        # Read the Switches
14 lw t0, 0(a1)
15
16 li a0, GPIO_LEDS
17 srl t0, t0, 16
18 sw t0, 0(a0)           # Write the LEDs
19
20 beq zero, zero, next
21 .end

```

Memory [0x80001400+0xc].dbgmem X

1	Offset: 00 01 02 03 04 05	06 07 08 09 0A 0B 0C 0D 0E 0F
2	80001400: 55 55 55 55 55 55	00 00 FF FF 00 00
3		

Figür 58. Anahtarlarla LEDlerin değişimi

- f. Programın makine yönergelerini depolayan RAM adresleri gibi diğer bellek yerlerini de görebilirsin. 0x0'da (RAM belleğine atanan ilk adres) başlayıp 0x100 bayt kaplayan başka bir bellek aralığı aç (Figür 59). LedsSwitches programının yönergelerinin 0x90-0xC4 adres aralığında, başlangıç programının sonrasında (Startup.S), depolandığını göreceksin.

Memory [0x0+0x100].dbgmem X	
1	Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
2	00000000: 73 10 20 B8 73 10 20 B8 81 40 01 41 81 41 01 42
3	00000010: 81 42 01 43 81 43 01 44 81 44 01 45 81 45 01 46
4	00000020: 81 46 01 47 81 47 01 48 81 48 01 49 81 49 01 4A
5	00000030: 81 4A 01 4B 81 4B 01 4C 81 4C 01 4D 81 4D 01 4E
6	00000040: 81 4E 01 4F 81 4F 37 53 55 55 13 03 53 55 73 10
7	00000050: 03 7C 97 31 00 00 93 81 E1 8E 17 31 00 00 13 01
8	00000060: 61 0E 17 25 00 00 13 05 E5 0D 97 25 00 00 93 85
9	00000070: 65 0D 63 77 B5 00 23 20 05 00 11 05 E3 6D B5 FE
10	00000080: 91 20 01 45 81 45 29 20 01 A0 00 00 00 00 00 00
11	00000090: 37 0E 01 00 13 0E FE FF B7 1E 00 80 93 8E 8E 40
12	000000a0: 23 A0 CE 01 B7 15 00 80 93 85 05 40 83 A2 05 00
13	000000b0: 37 15 00 80 13 05 45 40 93 D2 02 01 23 20 55 00
14	000000c0: E3 02 00 FE 41 11 22 C4 4A C0 17 04 00 00 13 04
15	000000d0: 64 F3 17 09 00 00 13 09 E9 F2 33 09 89 40 06 C6
16	000000e0: 26 C2 13 59 29 40 63 09 09 00 81 44 1C 40 85 04
17	000000f0: 11 04 82 97 E3 1C 99 FE 17 04 00 00 13 04 84 F0
18	

Figür 59. 0x0'dan 0x100'a bellek adresleri

- g. Programın yönergeleri için makine kodunu programın şuradan erişilebilen tersine çevirisini açarak görebilirsin:
[RVfpgaPath]/RVfpga/examples/LedsSwitches/.pio/build/swervolf_nexys/firmware.dis (see Figür 60). İki figürü kıyaslayıp programın yönergelerini tanımaya çalış.

```

65 Disassembly of section .text:
66
67 00000090 <main>:
68   90: 00010e37      lui t3,0x10
69   94: fffe0e13      addi t3,t3,-1 # ffff <_sp+0xceb7>
70   98: 80001eb7      lui t4,0x80001
71   9c: 408e8e93      addi t4,t4,1032 # 80001408 <OVERLAY_END_OF_OVERLAYS+0xa0001408>
72   a0: 01cea023      sw t3,0(t4)
73
74 000000a4 <next>:
75   a4: 800015b7      lui a1,0x80001
76   a8: 40058593      addi a1,a1,1024 # 80001400 <OVERLAY_END_OF_OVERLAYS+0xa0001400>
77   ac: 0005a283      lw t0,0(a1)
78   b0: 80001537      lui a0,0x80001
79   b4: 40450513      addi a0,a0,1028 # 80001404 <OVERLAY_END_OF_OVERLAYS+0xa0001404>
80   b8: 0102d293      srli t0,t0,0x10
81   bc: 00552023      sw t0,0(a0)
82   c0: fe0002e3      beqz zero,a4 <next>
83

```

Figür 60. LedsSwitches programının tersine çeviri sürümü

E. LedsSwitches_C-Lang programı

Program LedsSwitches_C-Lang.c (Figür 61), LedsSwitches.s programı (Figür 52) ile birebirdir ancak çevirici yerine C ile yazılmıştır.

```

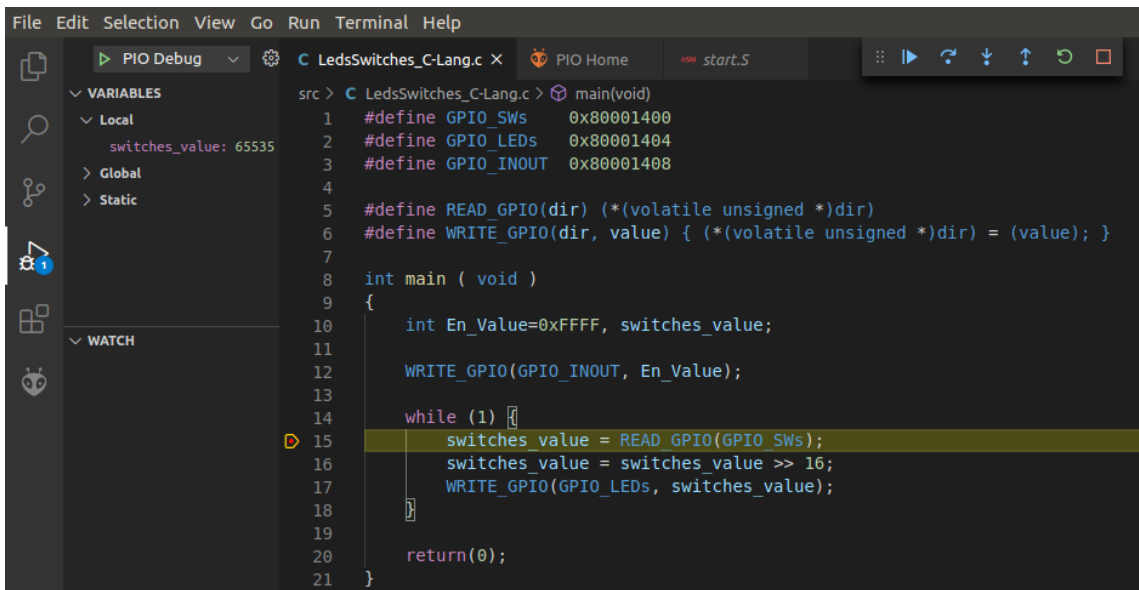
1 #define GPIO_SWs      0x80001400
2 #define GPIO_LEDs     0x80001404
3 #define GPIO_INOUT    0x80001408
4
5 #define READ_GPIO(dir) (*(volatile unsigned *)dir)
6 #define WRITE_GPIO(dir, value) { (*(volatile unsigned *)dir) = (value); }
7
8 int main ( void )
9 {
10     int En_Value=0xFFFF, switches_value;
11
12     WRITE_GPIO(GPIO_INOUT, En_Value);
13
14     while (1) {
15         switches_value = READ_GPIO(GPIO_SWs);
16         switches_value = switches_value >> 16;
17         WRITE_GPIO(GPIO_LEDs, switches_value);
18     }
19
20     return(0);
21 }

```

Figür 61. LedsSwitches_C-Lang.c

Bu programı FPGA kartında çalıştırıp ayıklamak için şu adımları izle:

1. Önceki örnekleri uyguladıysan RVfpga FPGA kartında programlanmıştır, bu nedenle yeniden programlaman gerekmez. Ancak, RVfpga'i kartta yeniden programlaman gerekiyorsa, AL_Operations örneği yerine LedsSwitches_C-Lang örneğini kullanarak Bölüm A'da açıklandığı gibi yap.
2. Üst menü çubuğunda, *File* → *Open Folder* üzerine tıkla, *[RVfpgaPath]/RVfpga/examples/* dizinine git. *LedsSwitches_C-Lang* dizinini seçip OK tıkla.
3. Ayıklayıcıyı çağırmadan önce C kodunda satır 15'te kesme noktası ayarla.
4. Ardından ayıklamayı başlat. Program yürütmeye başlayıp kesme noktasında duracaktır (Figür 62).



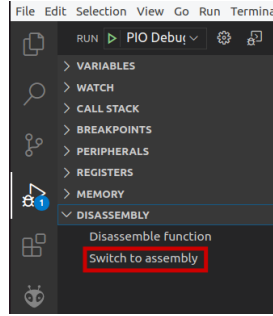
Figür 62. Kesme noktasında (breakpoint) yürütme durdurulmuş

5. Tıklamalar arasında anahtarları değiştirerek programın yürütmesini



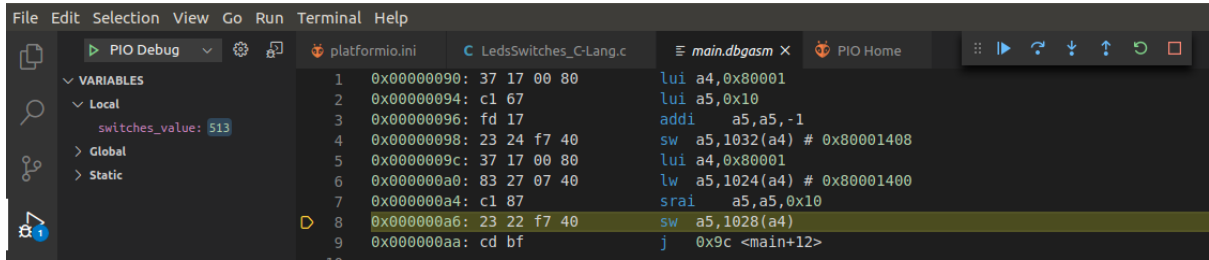
sürdür. LEDler anahtarların değerini gösterecektir.

6. C programının yürütmesini yukarıdaki gibi görebilir ya da derleyicinin oluşturduğu çeviricinin yürütmesini Switch to assembly tıklayarak görebilirsin, Figür 63'teki gibi.



Figür 63. Çeviriciye geç

7. Çeviricideki program (Figür 64) ilk bir yükleme yönergesiyle anahtarlardaki değeri okuyup (`lw a5,1024(a4)`) ardından bir depolama yönergesiyle LEDlere yazar (`sw a5,1028(a4)`). Adım adım yürütüp anahtarların değerini değiştirip LEDlerin yeni anahtar değerlerini yansıttığını doğrula.

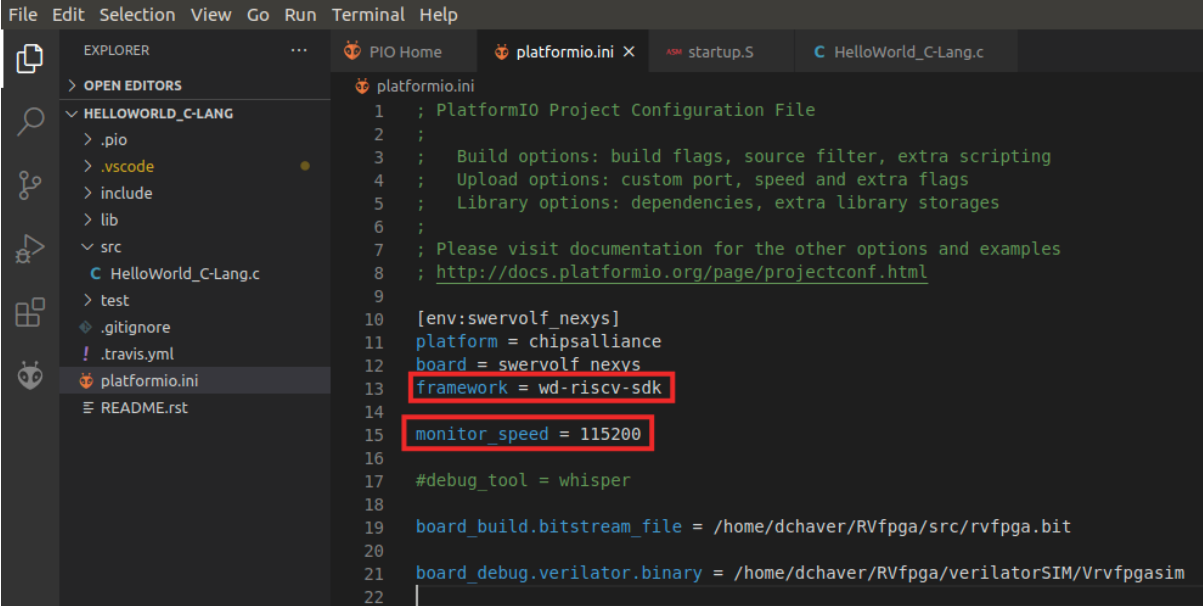


Figür 64. Çevirici programı

F. HelloWorld_C-Lang programı

İkinci C örneği istemciye dizesel port üzerinden kısa bir ileti yazdırır. Bu iletiyi görmek için istediğin bir terminal emülatörü kullanabilirsin *gtkterm*, *minicom*, gibi; ancak PlatformIO kendi dizesel monitörünü sağladığından burada onun kullanımını gösteriyoruz.

PlatformIO dizesel monitörünü yapılandırmak için birkaç parametre yapılandırılmalıdır; özellikle dizesel veri iletimi için veri oranı (saniye başına bit biriminde, ya da baud) belirlenmelidir, bunu da *platformio.ini* dosyası (önemli olarak bu dosya PlatformIO projelerinin içerisinde) içerisindeki *monitor_speed* parametresiyle yapabiliriz. Figür 65'e göz at.



```

1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter, extra scripting
4 ; Upload options: custom port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ;
7 ; Please visit documentation for the other options and examples
8 ; http://docs.platformio.org/page/projectconf.html
9
10 [env:swervolf_nexys]
11 platform = chipsalliance
12 board = swervolf_nexys
13 framework = wd-riscv-sdk
14
15 monitor_speed = 115200
16
17 #debug_tool = whisper
18
19 board_build.bitstream_file = /home/dchaver/RVfpga/src/rvfpga.bit
20
21 board_debug.verilator.binary = /home/dchaver/RVfpga/verilatorSIM/Vrvfpgasim
22

```

Figür 65. Dizisel monitör yapılandırması

Ek olarak kendini şu komutları bir terminalde çalıştırarak dialout, tty, uucp gruplarına eklemen gerekecek:

```

sudo usermod -a -G dialout $USER
sudo usermod -a -G tty $USER
sudo usermod -a -G uucp $USER

```

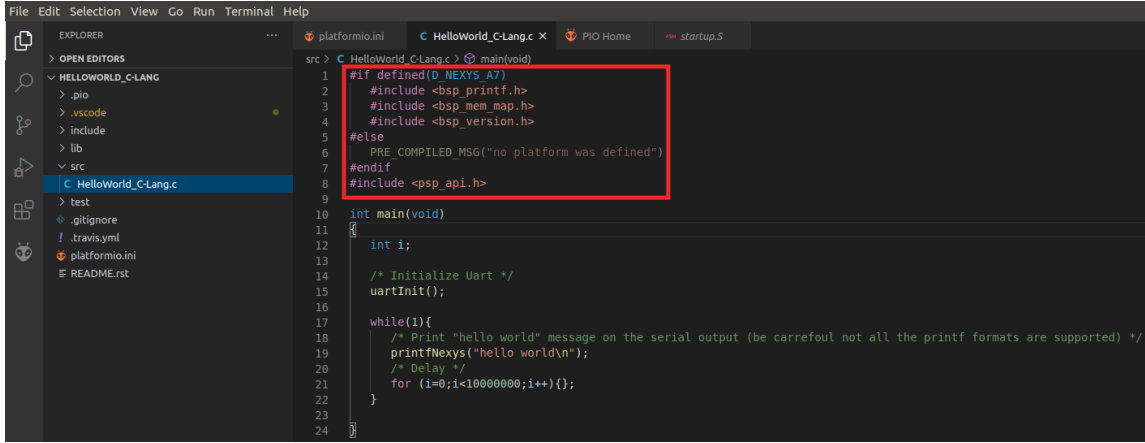
Bu üç komuttan sonra bilgisayarını yeniden başlat ki gruptaki değişiklikler etkisin.

Windows/macOS: Windows ile macOS kullanıcılarının yukarıdaki adımı yapması gerekmez.

Bu program WD'nin Firmware Package'inde sunduğu İşlemci Destek Paketi (PSP) ile Kart Destek Paketi (BSP) kullanır (<https://github.com/westerndigitalcorporation/riscv-fw-infrastructure>). Bu kütüphaneler projede platformio.ini'de belirli bir komutu kullanıp (framework = wd-riscv-sdk), Figür 65'te gösterildiği gibi, C programının başlangıcında doğru dosyaları içererek, Figür 66'da gösterildiği gibi, kullanılır. Kütüphanelerin hepsini sisteminde şu yollarda bulabilirsin:

- PSP: ~/.platformio/packages/framework-wd-riscv-sdk/psp/
- BSP: ~/.platformio/packages/framework-wd-riscv-sdk/board/nexys_a7_eh1/bsp/

Bu kütüphaneler kesintileri kullanma, karakter dizisi yazdırma, yazmaçları ayrı ayrı okuma/yazma gibi birçok işlevle makro sağlar. Bu örnekte dizisel monitörde bir ileti yazdırmak için printfNexys işlevini kullanacağız. Arkasından gelen örneklerle deneylerde diğer işlevlerle makroların başka amaçlar için nasıl kullanılacağını göstereceğiz.



Figür 66. HelloWorld_C-Lang.c içerisinde .h dosyalarını içer

Bu kodu FPGA kartında çalıştırıp ayıklamak için şu adımları izle:

1. Önceki örnekleri uyguladıysan RVfpga FPGA kartında programlanmıştır, bu nedenle yeniden programlaman gerekmez. Ancak, RVfpga'i kartta yeniden programlaman gerekiyorsa, AL_Operations örneği yerine HelloWorld_C-Lang örneğini kullanarak Bölüm A'da açıklandığı gibi yap.
2. VSCode'u aç. VSCode açtığında PlatformIO kendiliğinden açılmalı. Üst çubukta File → Open Folder üzerine tıklayıp [RVfpgaPath]/RVfpga/examples/ dizinine git. HelloWorld_C-Lang klasörünü seçip OK'a tıkla.
3. HelloWorld_C-Lang.C programı (Figür 67) UART'a ilk değerini verip (işlev **uartInit**) ardından **printfNexys** (bu işlevlerin gerçekleştirmesini `~/platformio/packages/framework-wd-riscv-sdk/board/nexys_a7_ah1/bsp/bsp_printf.c` dosyasının içerisinde bulabilirsin) işlevini kullanarak karakter dizisini dizesel port üzerinden yollar. Ardından döngünün başına dönmekten biraz geciktirir.

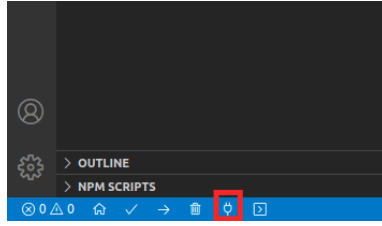
```

1  #if defined(D_NEXYS_A7)
2  #include <bsp_printf.h>
3  #include <bsp_mem_map.h>
4  #include <bsp_version.h>
5  #else
6  PRE_COMPILED_MSG("no platform was defined")
7  #endif
8  #include <psp_api.h>
9
10 int main(void)
11 {
12     int i;
13
14     /* Initialize Uart */
15     uartInit();
16
17     while(1){
18         /* Print "hello world" message on the serial output (be carrefoul not all the printf formats are supported) */
19         printfNexys("hello world\n");
20         /* Delay */
21         for (i=0;i<10000000;i++){
22         }
23     }
24 }

```

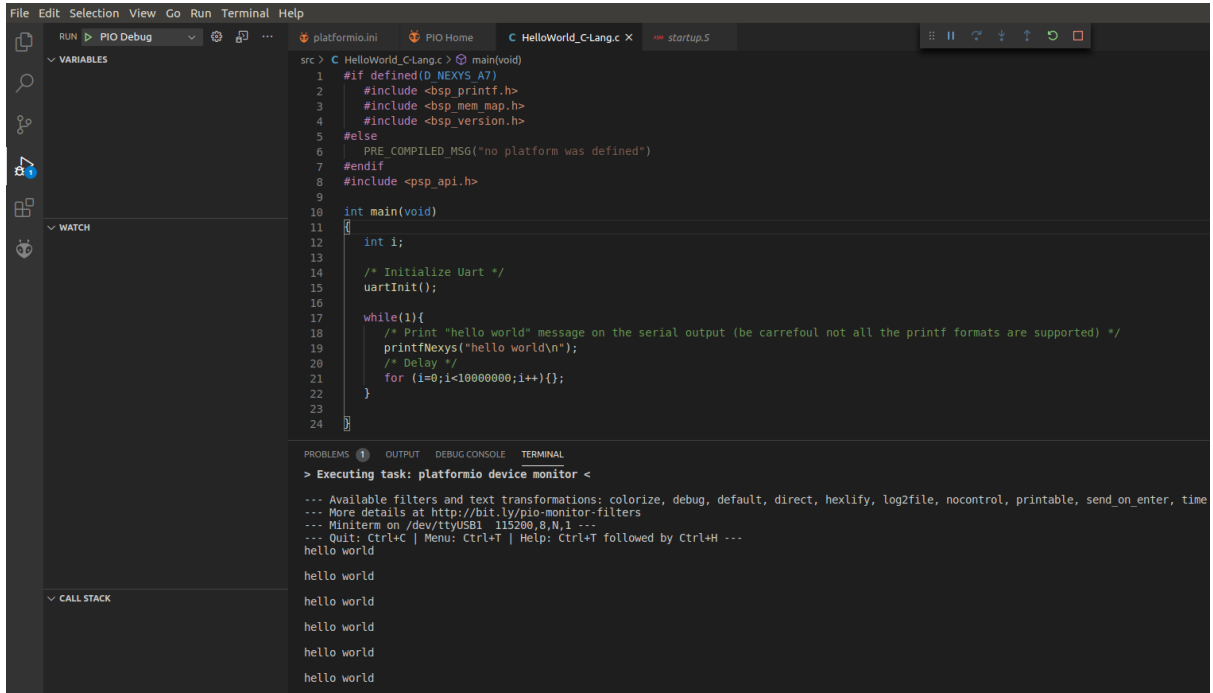
Figür 67. HelloWorld_C-Lang.C ana işlevi

4. Ayıklayıcıyı PlatformIO'da başlat. Program çalışmaya başlayınca VS Code'un aşağısından erişilebilen *kablo* butonuna tıklayarak dizesel monitörü aç (Figür 68).



Figür 68. Dizisel terminali aç

5. Dizisel monitör sürekli “HELLO WORLD !!!” iletisini yazdırır, Figür 69’dan görülebileceği gibi.



Figür 69. Programın yürütmesi

G. VectorSorting_C-Lang program

Son olarak bir vektörün, A, değerlerini en büyükten en küçüğe doğru sıralayıp sıralanmış değerleri ikinci bir vektöre, B, koyan bir program gösteriyoruz. Vektör A değerleri sıfırla değiştirilir. Figür 70 programı gösterir.

```
1 #define N 8
2
3 int A[N]={7,3,25,4,75,2,1,1};
4 int B[N];
5
6 int main ( void )
7 {
8     int max, ind, i, j;
9
10    for(j=0; j<N; j++){
11        max=0;
12        for(i=0; i<N; i++){
13            if(A[i]>max){
14                max=A[i];
15                ind=i;
16            }
17        }
18    }
```

```

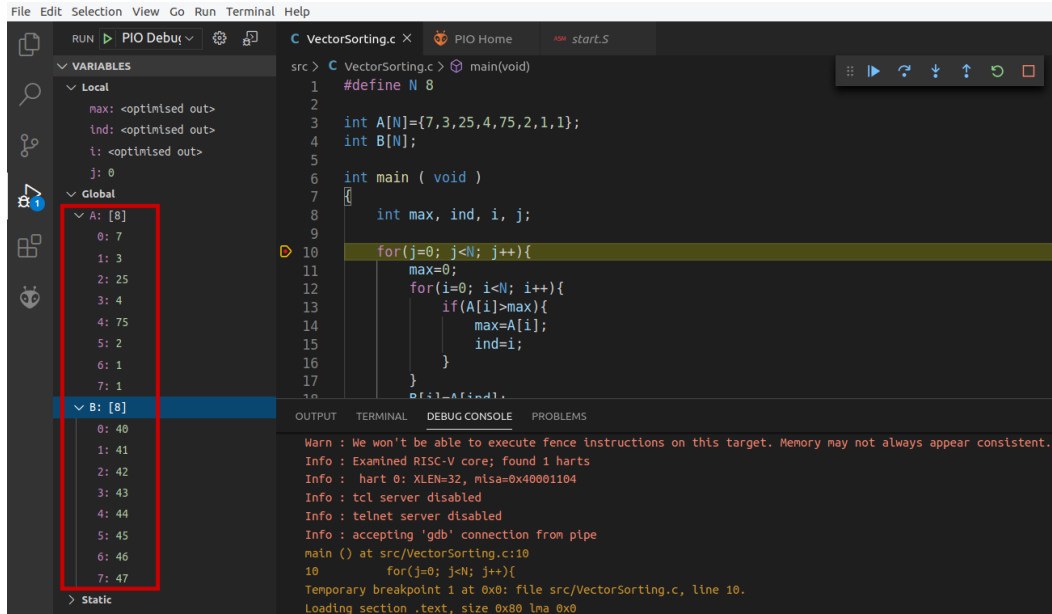
17     }
18     B[j]=A[ind];
19     A[ind]=0;
20 }
21
22 while(1);
23 }

```


Figür 70. VectorSorting_C-Lang.c

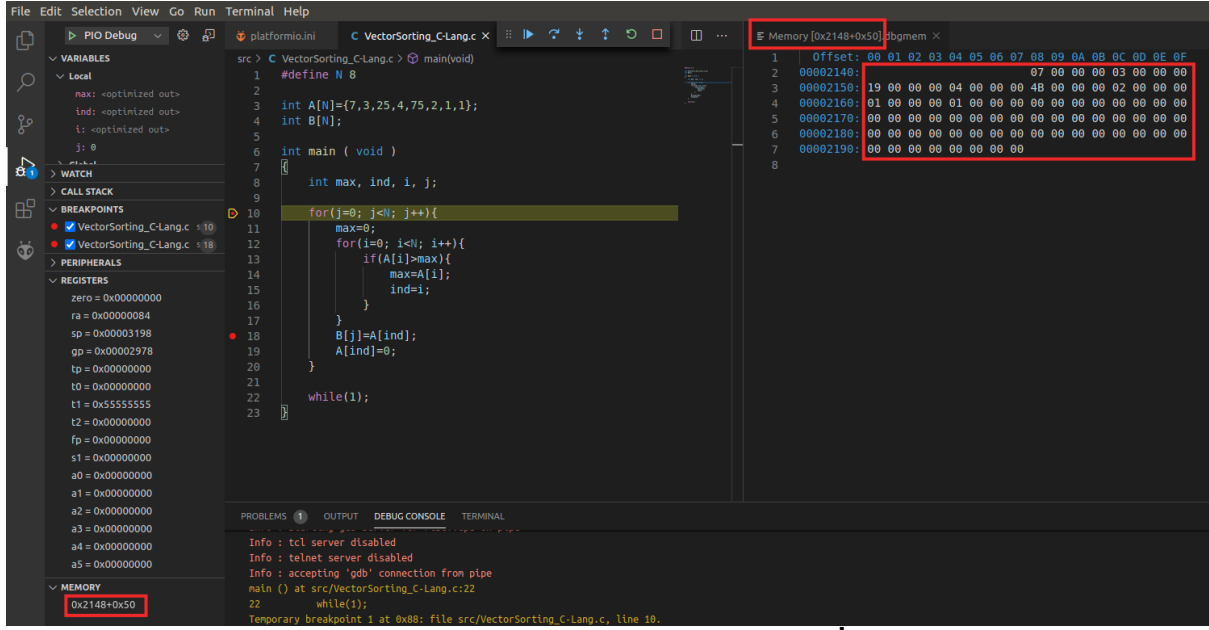
Bu programı FPGA kartında çalıştırıp ayıklamak için şu adımları izle:

1. Önceki örnekleri uyguladıysan RVfpga FPGA kartında programlanmıştır, bu nedenle yeniden programlaman gerekmez. Ancak, RVfpga'i kartta yeniden programlaman gerekiyorsa, AL_Operations örneği yerine VectorSorting_C-Lang örneğini kullanarak Bölüm A'da açıklandığı gibi yap.
2. Üst menü çubuğunda *File* → *Open Folder* üzerine tıklayıp *[RVfpgaPath]/RVfpga/examples/* klasörüne git. *VectorSorting_C-Lang* klasörünü seçip OK'a tıkla.
3. Satır 10'a bir kesme noktası yerleştirip ayıklamayı başlat. Yürütme *for* döngüsünün başında duracaktır (Figür 71). Debugger Side Bar (Ayıklayıcı Yan Çubuğu) içerisindeki *VARIABLES (DEĞİŞKENLER)* bölümünü genişletip A ile B dizilerinin değerlerini çözümü (Figür 71'de kırmızıyla parlatılmış).



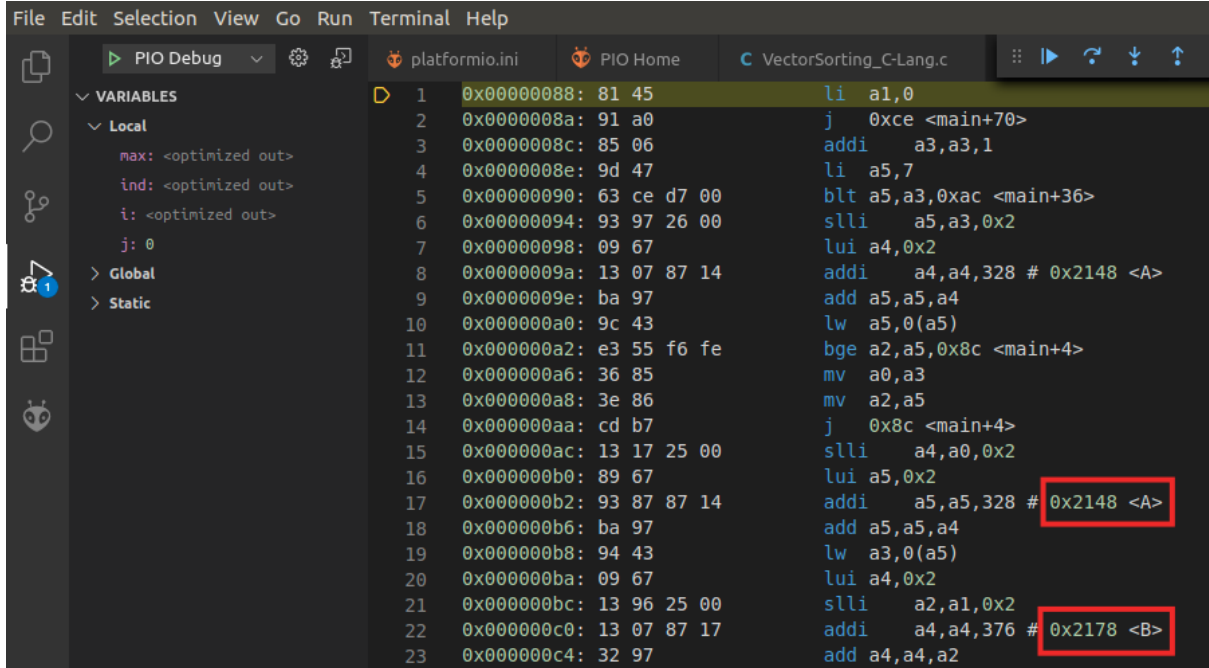
Figür 71. Yürütme programın başında durdurulmuş

4. Satır 18'e başka bir kesme noktası ekleyip yürütmeyi  üzerine tıklayarak sürdür (Figür 72). Memory Display açıp (LedsSwitches programı için açıklandığı gibi, Figür 54) bellekte vektör A'nın depolandığı adres 0x2148'den başlayarak 0x50 bayt göster (Figür 72). Vektör A (0x2148-0x2167 aralığında) ile B (0x2178-0x2197 aralığında) için ilk değerleri görüntüleyebilirsiniz.



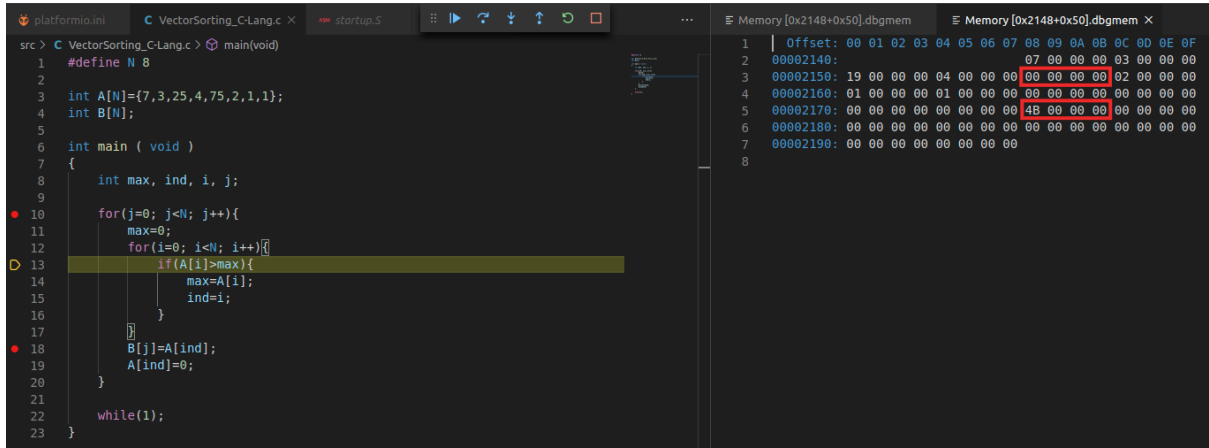
Figür 72. A ile B dizileri için bellek ekranı – İlk durum.

Önemli olarak Figür 63’de açıklandığı gibi vektör A ile B’nin bellekte nerede depolandığını çeviriciye geçip bu vektörlere erişilen yönergeleri çözümleyerek kolayca bulabilirsin (Figür 73). Figürde görebileceğin üzere çoğunlukta yorumlar bu bilgiyi sağlar; ancak o yönergelere adımlayıp yazmakta tutulan değeri görebilirsin de.



Figür 73. A ile B’nin bellekte depolandığı adres.

Step Over (Üzeri Adım) butonuna iki kez tıkladığında (🔄) B’nin ilk bileşeninin bellekte depolanıp denk gelen A değerinin 0’a ayarlandığını göreceksin (Figür 74).



```

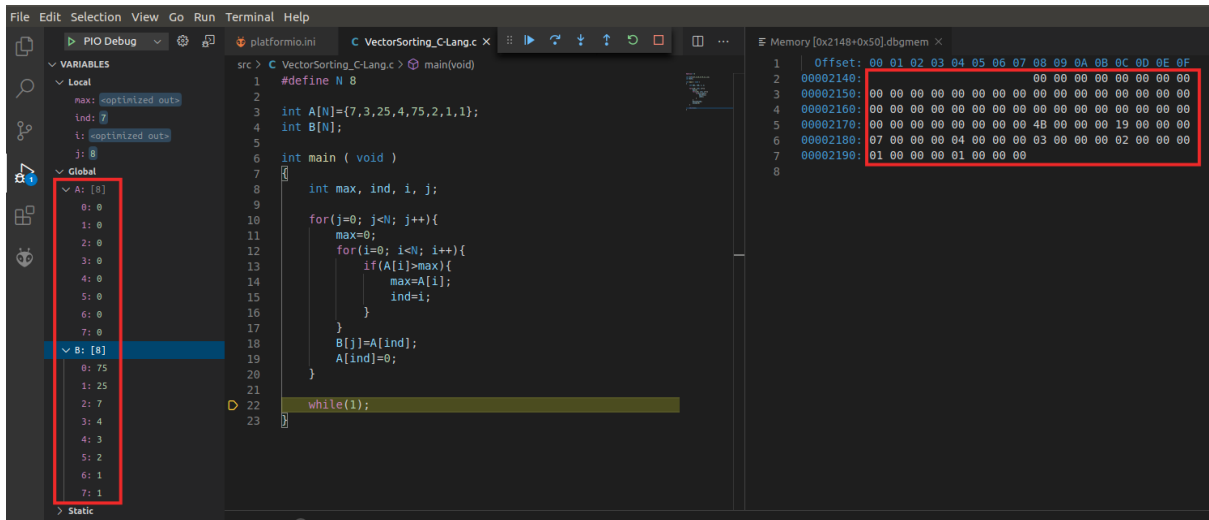
src > C VectorSorting_C-Lang.c > main(void)
1 #define N 8
2
3 int A[N]={7,3,25,4,75,2,1,1};
4 int B[N];
5
6 int main ( void )
7 {
8     int max, ind, i, j;
9
10    for(j=0; j<N; j++){
11        max=0;
12        for(i=0; i<N; i++){
13            if(A[i]>max){
14                max=A[i];
15                ind=i;
16            }
17        }
18        B[j]=A[ind];
19        A[ind]=0;
20    }
21
22    while(1);
23 }

```

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00002140:	19	00	00	00	04	00	00	00	07	00	0A	0B	0C	0D	0E	0F
00002150:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00002160:	01	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00
00002170:	00	00	00	00	00	00	00	00	4B	00	00	00	00	00	00	00
00002180:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00002190:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figür 74. A ile B dizileri için Memory Display – B'nin ilk bileşenini depolayıp denk gelen A bileşenini sıfırla.

- Bütün kesme noktalarını kaldır, yürütmeyi sürdür, birkaç saniye sonra beklet – burada program yürütmeyi bitirmiş olacaktır. Yeniden A ile B dizilerinde tutulan değeri çözümü. Figür 75'te gösterildiği gibi vektör B, orijinal vektör A'dan değerleri en büyüktен en küçüğe sıralı tutup vektör A bütünüyle sıfır tutmaktadır (bunu soldaki değişkenler listesinde de sağdaki bellek konsolunda da görebilirsin).



```

src > C VectorSorting_C-Lang.c > main(void)
1 #define N 8
2
3 int A[N]={7,3,25,4,75,2,1,1};
4 int B[N];
5
6 int main ( void )
7 {
8     int max, ind, i, j;
9
10    for(j=0; j<N; j++){
11        max=0;
12        for(i=0; i<N; i++){
13            if(A[i]>max){
14                max=A[i];
15                ind=i;
16            }
17        }
18        B[j]=A[ind];
19        A[ind]=0;
20    }
21
22    while(1);
23 }

```

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00002140:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00002150:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00002160:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00002170:	00	00	00	00	00	00	00	00	4B	00	00	00	00	00	00	00
00002180:	07	00	00	00	04	00	00	00	03	00	00	00	02	00	00	00
00002190:	01	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00

Figür 75. Yürütme program sonunda durur

H. DotProduct_C-Lang

Son örnek program, DotProduct_C-Lang.c, iki vektörün iç çarpımını hesaplar. Programın iki işlevi vardır: *main* ile *dotproduct*. İlk işlev ikinciyi üç girdi argümanla çalıştırır: vektör boyutuyla iki vektörün ilk adresleri. Ardından *dotproduct* işlevi iki vektörün iç çarpımını hesaplayıp sonucu döndürür.

```

1  #define DIM 3
2
3  double dot;
4
5  double dotproduct(int n, double a[], double b[]){
6      volatile int i;
7      double sum=0;
8
9      for (i=0; i<n; i++) {
10         sum += a[i]*b[i];
11     }
12     return sum;
13 }
14
15 void main(void) {
16     double x[DIM] = {3.1, 4.3, 5.9};           // x is an array of size 3 (DIM)
17     double y[DIM] = {1.4, 2.2, 3.7};         // same as x
18
19     dot = dotproduct(DIM, x, y);
20
21     return;
22 }

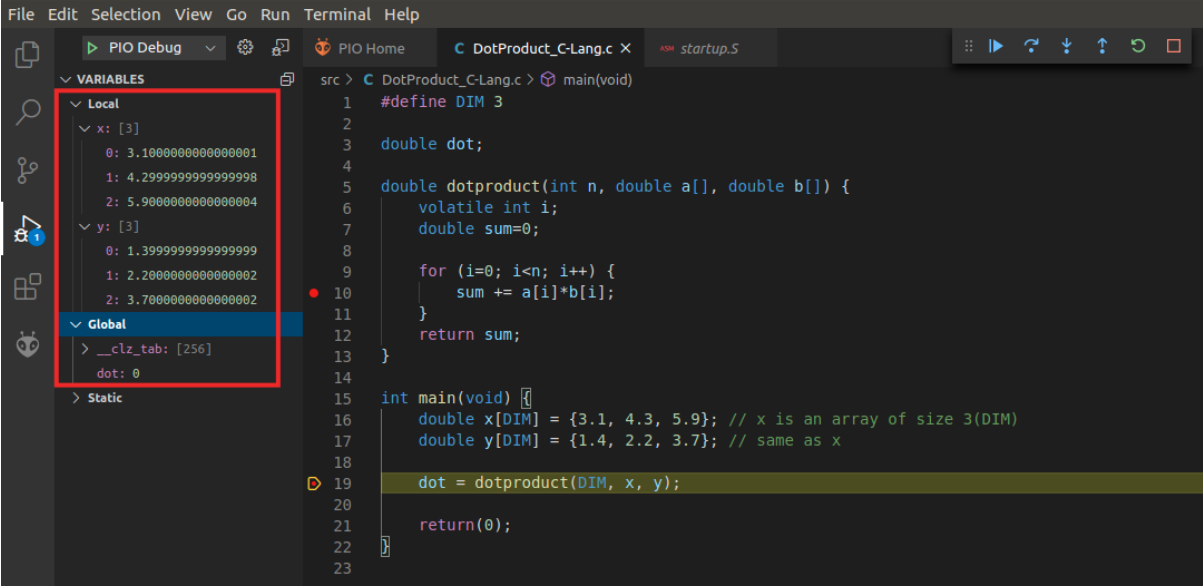
```

Figür 76. DotProduct_C-Lang.c

Bu örnekte gerçek sayılar üzerinde işlem yapıyoruz (önemli olarak *x*, *y* ile *dot* değişkenleri için veri türü *double*). Ancak SweRV EH1 işlemcisinde kayan nokta desteği yok. Dolayısıyla örnek *gcc*'nin sağladığı yazılım kayan nokta kütüphanesiyle kayan nokta emülasyonu kullanır (<https://gcc.gnu.org/onlinedocs/gccint/Soft-float-library-routines.html>). Bu kütüphane kayan nokta yönergelerinin oluşumunu engellemek için `-msoft-float` içerildiğinde kullanılır.

Bu kodu FPGA kartında çalıştırıp ayıklamak için şu adımları izle:

1. Önceki örnekleri uyguladıysan RVfpga FPGA kartında programlanmıştır, bu nedenle yeniden programlaman gerekmez. Ancak, RVfpga'i kartta yeniden programlaman gerekiyorsa, AL_Operations örneği yerine DotProduct_C-Lang örneğini kullanarak Bölüm A'da açıklandığı gibi yap.
2. Üst menü çubuğunda *File* → *Open Folder* tıklayıp *[RVfpgaPath]/RVfpga/examples/* dizinine git. *DotProduct_C-Lang* dizinini seçip OK'a tıkla.
3. Ayıklayıcıyı çağırmadan önce satır 10 ile 19'a kesme noktaları ayarla (Figür 77).
4. Ardından ayıklamayı başlat. Program yürütmeye başlayacaktır; ilk kesme noktasında durdur (Figür 77).
5. Ayıklayıcı yan çubuğunda Variables (Değişkenler) bölümünü genişlet (Figür 77). İki vektör *main*'de atanan ilk değerleri barındırır. *dot* değişkeni 0'a ilk değerlenmiştir.




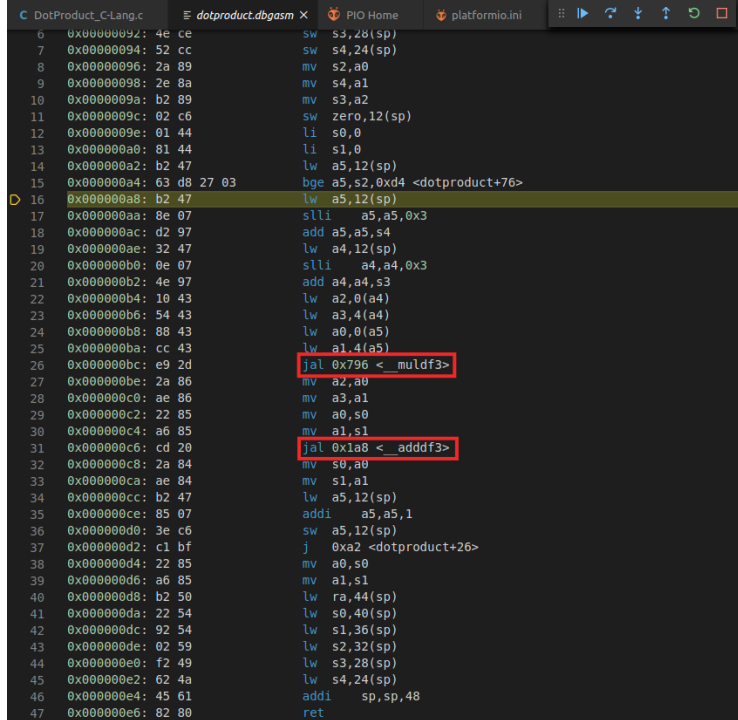
```

File Edit Selection View Go Run Terminal Help
PIO Debug PIO Home DotProduct_C-Lang.c startup.S
VARIABLES
  Local
    x: [3]
      0: 3.1000000000000001
      1: 4.2999999999999998
      2: 5.9000000000000004
    y: [3]
      0: 1.3999999999999999
      1: 2.2000000000000002
      2: 3.7000000000000002
    Global
      __clz_tab: [256]
      dot: 0
  Static
src > C DotProduct_C-Lang.c > main(void)
1 #define DIM 3
2
3 double dot;
4
5 double dotproduct(int n, double a[], double b[]) {
6     volatile int i;
7     double sum=0;
8
9     for (i=0; i<n; i++) {
10        sum += a[i]*b[i];
11    }
12    return sum;
13 }
14
15 int main(void) {
16     double x[DIM] = {3.1, 4.3, 5.9}; // x is an array of size 3(DIM)
17     double y[DIM] = {1.4, 2.2, 3.7}; // same as x
18
19     dot = dotproduct(DIM, x, y);
20
21     return(0);
22 }
23

```

Figür 77. DotProduct_C-Lang programı: ilk kesme noktasında değişkenlerin değerleri

6. Programın yürütmesini sürdür  . Program ikinci kesme noktasında duracaktır (satır 10).
7. Çeviriciye geç (Figür 63'te yaptığın gibi). Kayan nokta emülasyon yordamlarını görüp içeri adım yaparak detaylıca çözümleyebilirsin (Figür 78).



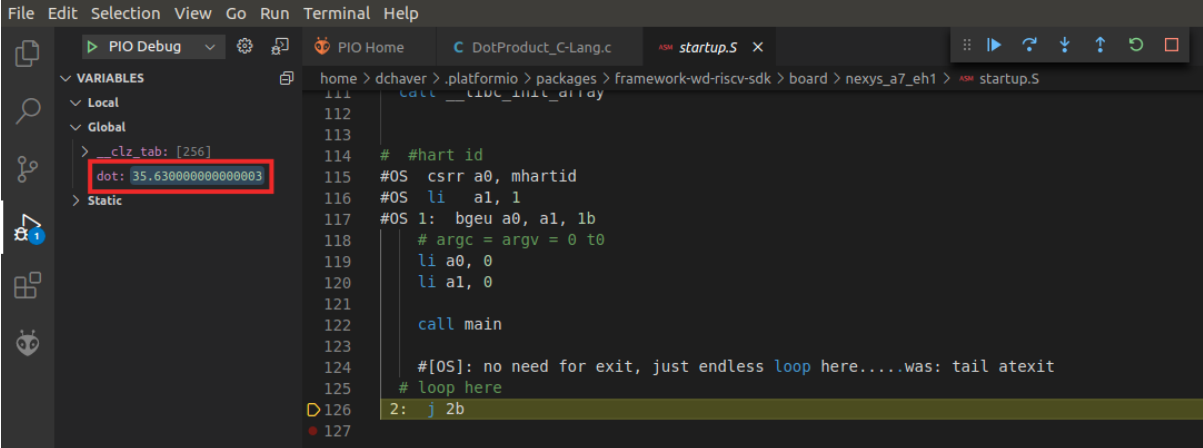
```

C DotProduct_C-Lang.c dotproduct.dbgasm PIO Home platformio.ini
6 0x00000092: 4e ce sw s3,28(sp)
7 0x00000094: 52 cc sw s4,24(sp)
8 0x00000096: 2a 89 mv s2,a0
9 0x00000098: 2e 8a mv s4,a1
10 0x0000009a: b2 89 mv s3,a2
11 0x0000009c: 02 c6 sw zero,12(sp)
12 0x0000009e: 01 44 li s0,0
13 0x000000a0: 81 44 li s1,0
14 0x000000a2: b2 47 lw a5,12(sp)
15 0x000000a4: 63 d8 27 03 bge a5,s2,0xd4 <dotproduct+76>
16 0x000000a8: b2 47 lw a5,12(sp)
17 0x000000aa: 8e 07 slli a5,a5,0x3
18 0x000000ac: d2 97 add a5,a5,s4
19 0x000000ae: 32 47 lw a4,12(sp)
20 0x000000b0: 0e 07 slli a4,a4,0x3
21 0x000000b2: 4e 97 add a4,a4,s3
22 0x000000b4: 10 43 lw a2,0(a4)
23 0x000000b6: 54 43 lw a3,4(a4)
24 0x000000b8: 88 43 lw a0,a5
25 0x000000ba: cc 43 lw a1,4(a5)
26 0x000000bc: e9 2d jal 0x796 <_muldf3>
27 0x000000be: 2a 86 mv a2,a0
28 0x000000c0: ae 86 mv a3,a1
29 0x000000c2: 22 85 mv a0,s0
30 0x000000c4: a6 85 mv a1,s1
31 0x000000c6: cd 20 jal 0x1a8 <_adddf3>
32 0x000000c8: 2a 84 mv s0,a0
33 0x000000ca: ae 84 mv s1,a1
34 0x000000cc: b2 47 lw a5,12(sp)
35 0x000000ce: 85 07 addi a5,a5,1
36 0x000000d0: 3e c6 sw a5,12(sp)
37 0x000000d2: c1 bf j 0xa2 <dotproduct+26>
38 0x000000d4: 22 85 mv a0,s0
39 0x000000d6: a6 85 mv a1,s1
40 0x000000d8: b2 50 lw ra,44(sp)
41 0x000000da: 22 54 lw s0,40(sp)
42 0x000000dc: 92 54 lw s1,36(sp)
43 0x000000de: 02 59 lw s2,32(sp)
44 0x000000e0: f2 49 lw s3,28(sp)
45 0x000000e2: 62 4a lw s4,24(sp)
46 0x000000e4: 45 61 addi sp,sp,48
47 0x000000e6: 82 80 ret

```

Figür 78. DotProduct_C-Lang programı: ikinci kesme noktasında çevirici kodu

8. C'ye dönüp iki kesme noktasını sil. Yürütmeyi sürdürüp bekle. *dot* değişkeninin değerinin iki vektörün iç çarpımına dönüştüğünü göreceksin (Figür 79).



The screenshot shows the PIO Debug interface. On the left, the 'VARIABLES' panel is expanded, showing a variable 'dot' with a value of 35.630000000000003. The main window displays the assembly code for 'startup.S', which includes comments and instructions for calculating the dot product of two arrays. The code is as follows:

```

111  call __libc_init_array
112
113
114  # #hart id
115  #0S csrr a0, mhartid
116  #0S li a1, 1
117  #0S 1: bgeu a0, a1, 1b
118      # argc = argv = 0 t0
119      li a0, 0
120      li a1, 0
121
122      call main
123
124      #[0S]: no need for exit, just endless loop here....was: tail atexit
125      # loop here
126      2: j 2b
127

```

Figür 79. DotProduct_C-Lang programı: iç çarpımın sonucu

9. Programla oynamayı bitirince *File* (Dosya) → *Close Folder* (Klasörü Kapat) üzerine tıklayarak projeyi kapat.

7. VERILATOR'DE SİMÜLASYON

Bu bölümde önceki bölümde kullanılan ilk programı (*AL_Operations*) Verilator'ü kullanarak RVfpgaSIM'de çalıştıracaksın. Verilator, RVfpga'i (*[RVfpgaPath]/RVfpga/src* içerisinde erişilebilir) tanımlayan Verilog'un simülasyonunu yapan bir donanım tanım dilidir (Hardware Description Language) (HDL). SoC'yi böyle çalıştırmak sistemin iç sinyallerini çözümlemene olanak sağlar ki bu SoC'ye iç işlemler ya da yeni donanım eklediğimiz gelecek deneylerle alıştırmalarda özellikle kullanışlı olacak.

Burada *AL_Operations*'ın, Bölüm 5 (Figür 43) içerisinde yürütüp çalıştırdığın ilk yalın çevirici programı, yönergeleriyle yazmaç değerlerini dönüm dönüm görmek için Verilator'ün nasıl kullanılacağını gösteriyoruz. PlatformIO kullanarak simülasyon izi oluşturup, ardından saat, süperskaler işlemcinin iki yönü için de yönergeler, simülasyon dalga biçimine yazmaç $\times 28$ (bir diğer deyişle yazmaç $\div 3$) sinyallerini ekleyip, GTKWave ile yönerge ile yazmaç sinyallerinin program yürütmesi ilerledikçe değiştiğini göreceksin.

SİMÜLASYON VERİ DOSYASINI OLUŞTUR, Vrvfpgasim:

[RVfpgaPath]/RVfpga/verilatorSIM dizini RVfpgaSIM için simülasyon ikili dosyası oluşturmak için *Makefile* ile *script (swervolf_0.7.vc)* barındırır. *script* Verilator için, diğer bilgilerin yanı sıra, SoC kaynaklarını nerede bulacağıyla ilgili bilgiler barındırır, ki bu durumda *[RVfpgaPath]/RVfpga/src* içerisinde. Ardından *AL-Operations* RVfpga'de çalışırken simülasyon izini oluşturmak için kullanılacak RVfpga ikili veri dosyasını nasıl oluşturabileceğini gösteriyoruz.

1. Bir terminal penceresinde şu komutları yürüterek simülasyon veri dosyasını oluştur:

```
cd [RVfpgaPath]/RVfpga/verilatorSIM
make clean
make
```

Vrvfpgasim (RVfpga simülasyon veri dosyası) dosyası
[RVfpgaPath]/RVfpga/verilatorSIM dizininin içerisinde oluşturulmuş olmalı.

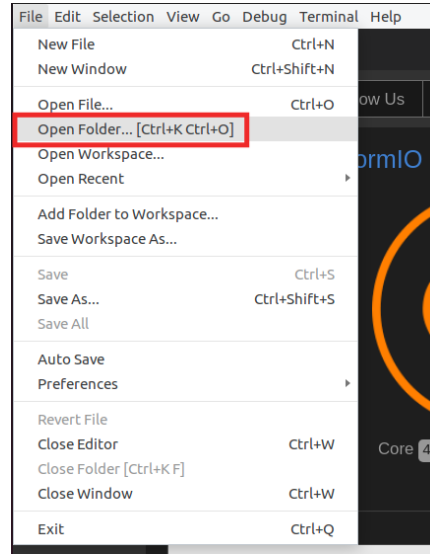
Windows: Windows kullanıyorsan bu adımları Cygwin terminalinde yapman gerekir (detaylı yönergeler için Ek B'ye bak). Önemli olarak C: Windows klasörü Cygwin'de şuradan bulunabilir: */cygdrive/c*. Bu bölümde kalan diğer yönergelerin hepsi Linux için tanımlananlarla aynıdır.

macOS: Detaylı yönergeler için Ek D'ye bak.

Vrvfpgasim KULLANARAK PLATFORMIO'DAN SİMÜLASYON İZİNİ OLUŞTUR:

Simülasyon ikili veri dosyası (Vrvfpgasim) oluşturulduğunda, onu PlatformIO içerisinde *AL_Operations* programının simülasyon izini (*trace.vcd*) oluşturmak için kullanacaksın.

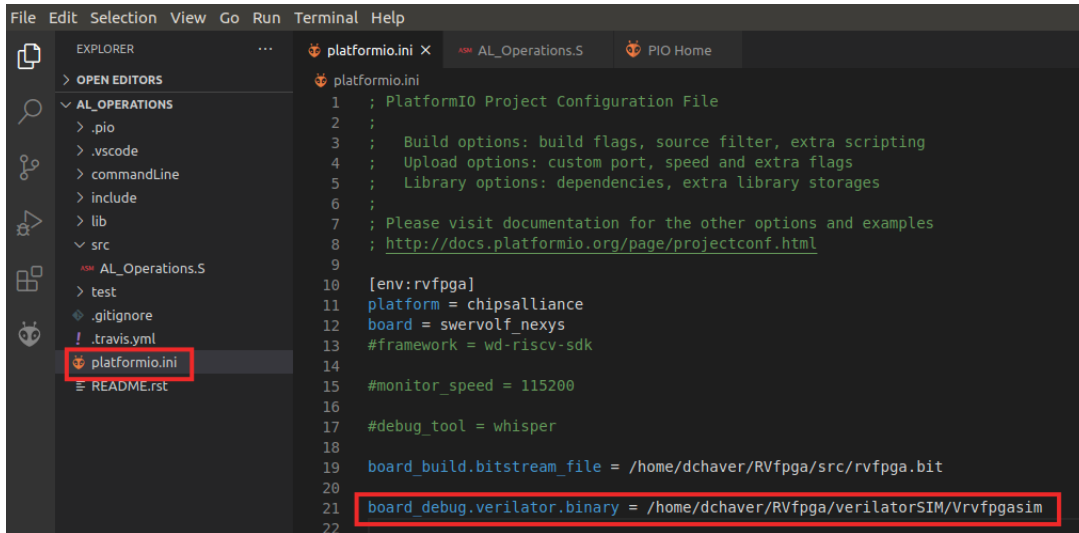
2. VSCode'u ardından PlatformIO'yu bilgisayarında aç.
3. Üst çubukta Dosya (File) → Klasörü Aç... (Open Folder...) üzerine tıklayıp (Figür 80), şu klasöre git *[RVfpgaPath]/RVfpga/examples/*



Figür 80. AL_Operations.S örneğini aç

4. *AL_Operations* klasörünü seç (açma, yalnızca seç) ardından OK'a tıkla. Örnek PlatformIO'da açılacak.
5. *platformio.ini* dosyasını aç. Şu satırı düzenleyerek ilk aşamada oluşturulan RVfpga simülasyon ikili veri dosyasına (Vrvfpgasim) giden yolu belirle (Figür 81'e göz at).


```
board_debug.verilator.binary =  
[RVfpgaPath] /RVfpga/verilatorSIM/Vrvfpgasim
```

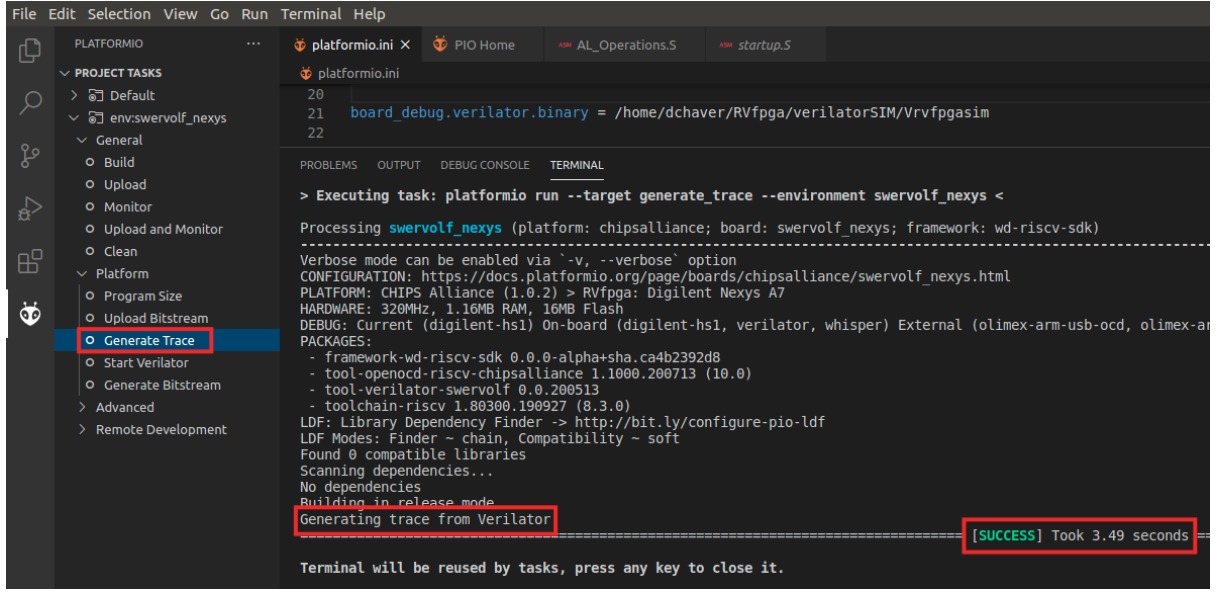


Figür 81. PlatformIO ilk değerlendirme dosyası: platformio.ini


Windows: Windows'ta RVfpga simülasyon yürütülebilirine Vrvfpgasim.exe denir. Yani:

```
board_debug.verilator.binary = [RVfpgaPath] \RVfpga\verilatorSIM\Vrvfpgasim.exe
```

6. Sol şerit menüdeki PlatformIO ikonuna  tıklayarak simülasyonu çalıştır, ardından Proje Görevleri (Project Tasks) → env:swervolf_nexys → Platform genişletip İz Oluştur (Generate Trace) tıkla, Figür 82de gösterildiği gibi.



Figür 82. Verilator'den iz oluşturma

Alternatif olarak izi bir PlatformIO terminal pencersinden oluşturabilirsin. Yeni bir terminal penceresi açmak için PlatformIO penceresinin aşağısındaki  butonuna tıkla (PlatformIO: Yeni Terminal butonu), ardından şu komutu PlatformIO terminaline yaz (ya da kopyala): `pio run --target generate_trace`

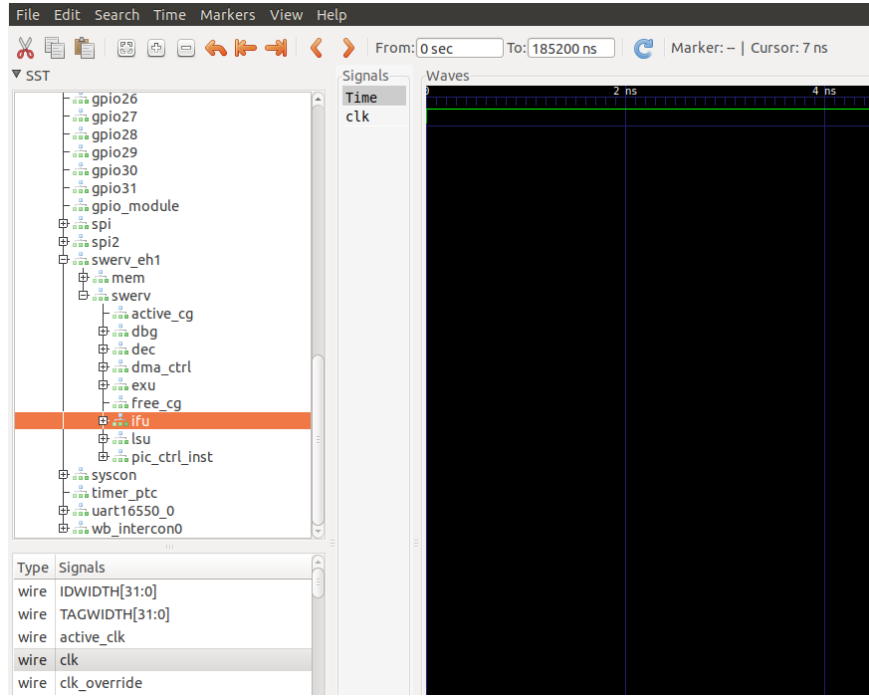
- Önceki adımdan birkaç saniye sonra dosya `trace.vcd`, `[RVfpgaPath]/RVfpga/examples/AL_Operations/.pio/build/swervolf_nexys` içerisinde oluşturulmuş olmalı, `GTKWave` ile açabilirsin.

```
gtkwave [RVfpgaPath]/RVfpga/examples/AL_Operations/.pio/build/swervolf_nexys/trace.vcd
```

WINDOWS: indirdiğin `gtkwave64` klasörü `bin` klasöründe `gtkwave.exe` adlı bir program içerir. `GTKWave`'i o uygulamanın üzerine çift tıklayarak aç. Uygulamanın üstünde **File (Dosya) – Open New Tab (Yeni Sekme Aç)** tıkla, ardından `trace.vcd` dosyasını oluşturulduğu şu klasörden aç `[RVfpgaPath]/RVfpga/examples/AL_Operations/.pio/build/swervolf_nexys`.

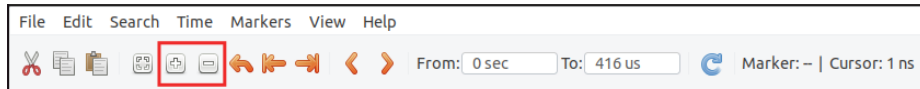
GTKWAVE'DE SİMÜLASYON İZİNİ ÇÖZÜMLE:

- Şimdi saat, yönerge, yazmaç sinyallerini ekleyeceksin. `GTKWave`'in sol üst panelinde sinyali çizgeye ekleyebilmek için SoC hiyerarşisini genişlet. Hiyerarşiyi **TOP** → **rvfpgasim** → **swervolf** → **swerv_eh1** → **swerv** içerisine doğru genişlet, **ifu** modülüne tıkla (Figür 83'te gösterildiği gibi parlatılacaktır), `clk` sinyalini seç (çekirdek için kullanılan saat), sağdaki beyaz Signals (Sinyaller) ya da siyah Waves (Dalgalar) paneline sürükle.



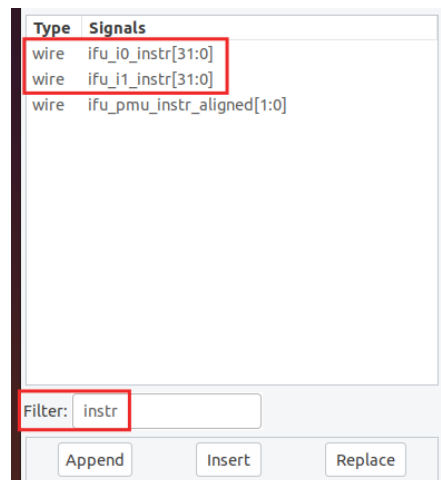
Figür 83. Sinyal *clk*'yi çizgeye ekle

9. Zoom Fit yapıp ardından birkaç kez Zoom yap ki saat sinyalinin değiştiğini görebil (Figür 84).



Figür 84. Zoom in

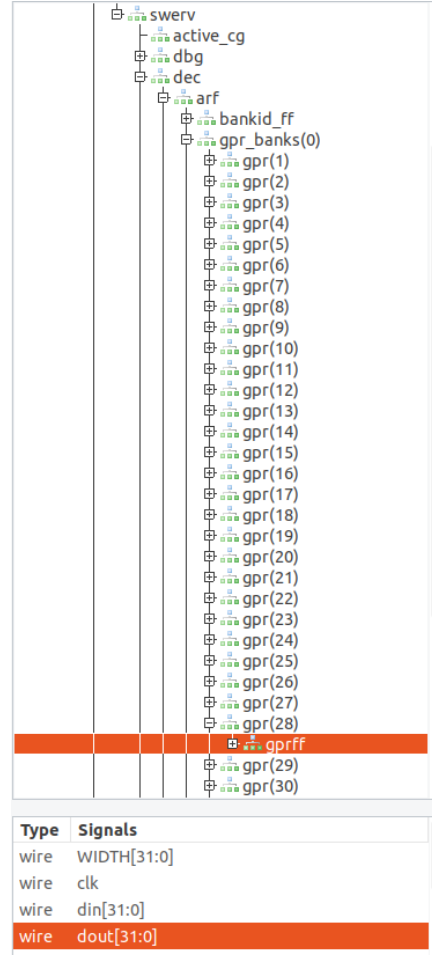
10. Şimdi iki yönlü süperskaler RISC-V çekirdeğinin bütün yönlerinde yürütülen yönergeleri gösteren sinyalleri ekle. Aynı modüle (*ifu*) *ifu_i0_instr[31:0]* ile *ifu_i1_instr[31:0]* (Figür 85) sinyallerini bul, Waves (Dalgalar) paneline sürük. *ifu* ön eki yönerge getirme ünitesini, *i0* süperskaler yol 0'ı, *i1* süperskaler yol 1'i gösterir; *instr[31:0]* 32-bit yönergeyi gösterir.



Figür 85. *ifu_i0_instr[31:0]* ile *ifu_i1_instr[31:0]* sinyallerini zamanlama dalga biçimine ekle

11. Şimdi t_3 yazmacının değerini tutan sinyali ekle (bir diğer deyişle yazmaç 28, x_{28}).

swerv altındaki hiyerarşiyi **dec** → **arf** → **gpr_banks(0)** → **gpr(28)** içerisine genişletip **gprff** (şu figürde gösterildiği gibi parlatılacaktır) modülüne tıkla, **dout[31:0]** (x28 yazmacının içeriklerini gösterir, *AL_Operations.S* örneğinde kullanılır) sinyalinin seçip **Waves (Dalgalar)** paneline sürük (Figür 86).



Figür 86. *dout[31:0]* sinyalini çizmeye ekle

Figür 87 *AL_Operations.S* programıyla denk makine kodunu gösterir.

# RISC-V assembly	# comment (t3 = x28)	# machine code
li t3, 0x0	# t3 = 0	# 0x00000E13
REPEAT:		
addi t3, t3, 6	# t3 = t3 + 6	# 0x006E0E13
addi t3, t3, -1	# t3 = t3 - 1	# 0xFFFE0E13
andi t3, t3, 3	# t3 = t3 AND 3	# 0x003E7E13
beq zero, zero, REPEAT	# Repeat the loop	# 0xFE000CE3
nop	# nop	# 0x00000013

Figür 87. *AL_Operations.S* ile denk makine kodu

Şimdi program yürütülürken sinyallerin değişimini gözlemle. Program çalıştıkça yönergelerle **t3** (yazmaç x28) Figür 88’de gösterilen değerlere dönüşmesini bekliyoruz:

	li t3, 0x0	# t3 = 0	# 0x00000E13
REPEAT:	addi t3, t3, 6	# t3 = 0 + 6 = 6	# 0x006E0E13
	addi t3, t3, -1	# t3 = 5	# 0xFFFE0E13
	andi t3, t3, 3	# t3 = 5 & 3 = 1	# 0x003E7E13

```

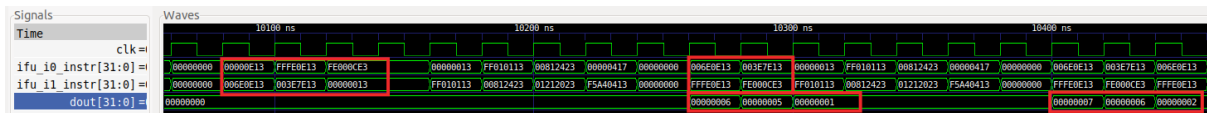
    beq  zero, zero, REPEAT    # Repeat the loop    # 0xFE000CE3
    nop                                # nop          # 0x00000013
REPEAT:  addi t3, t3, 6         # t3 = 1 + 6 = 7     # 0x006E0E13
        addi t3, t3, -1        # t3 = 7 - 1 = 6     # 0xFFFE0E13
        andi t3, t3, 3         # t3 = 6 & 3 = 2     # 0x003E7E13
        beq  zero, zero, REPEAT # Repeat the loop    # 0xFE000CE3
    ...

```

Figür 88. Yönerge akışıyla AL_Operations yürütmesi sırasında t3 (x28) yazmacının değerleri

12. 10100 ns çevresinde Zoom in yap, burada üç aritmetik-mantık yönergelerinin döngüdeki ilk ile ikinci yinelemesinin yürütmesinin çözümlemesini yapacaksın (Figür 89). İlk olarak ilk iki yönerge getirilir (li t3, 0x0 = 0x00000E13 ile addi t3, t3, 6 = 0x006E0E13), ifu_i0_instr[31:0] ile ifu_i1_instr[31:0] sinyallerinde gösterildiği gibi süperskaler RISC-V işlemcisinin yönlerine biri denk gelecek biçimde. Sonraki dönmde sonraki iki yönerge getirilir (addi t3, t3, -1 = 0xFFFE0E13 ile and.i t3, t3, 3 = 0x003E7E13). Sonraki dönmde son iki yönerge getirilir (beq zero, zero, REPEAT = 0xFE000CE3 ile nop = 0x00000013).

SweRV çekirdeğinin 9-aşamalı boruhatlı işlemcisiyle bağımlılıklarından dolayı yönergelerin etkisi yönerge getirildikten sekiz ya da daha çok dönm sonra görülür. İlk yönergeyle ikinci yönerge getirildikten sekiz dönm sonra x28 (t3) ilk yönergeden dolayı 0 olur (ki öyleydi): li t3, 0x0 (0x00000E13). Bir dönm sonra x28 sonraki yönergeden dolayı 0x6'a güncellenir: addi t3, t3, 6 (0x006E0E13). Ardından x28 sonraki yönergeden dolayı 5'e güncellenir: addi t3, t3, -1 (0xFFFE0E13). Son olarak x28 sonraki yönergeden dolayı 1'e güncellenir: andi t3, t3, 3 (0x003E7E13). Ardından sonraki iki yönerge getirilir: beq zero, zero, REPEAT (0xFE000CE3) ile nop (0x00000013), dallandırma olur, döngü tekrar eder. Bu Figür 88'de kestirildiği gibidir. Böyle bir yaklaşımla ikinci yinelemeyi çözümleyebilirsin, ki bu da Figür 84'te parlatılıp Figür 83'te kestirilmişti.



Figür 89. Örnekteki üç Aritmetik-Mantık yönergelerinin yürütmesi

8. WHISPER'DA SİMÜLASYON

Whisper (<https://github.com/chipsalliance/SweRV-ISS>), Western Digital'in SweRV mikro-denetleyicisini onaylamak (verification) için geliştirdiği bir RISC-V yönerge kümesi simülatörüdür (ISS). Kullanıcıların RISC-V donanımı olmadan RISC-V kodu çalıştırmasını sağlar. Whisper'ı kullanarak PlatformIO kullanıp Nexys A7 FPGA kartına gerek duymadan C ya da çevirici programlarını deneyip, çalıştırıp, ayıklayabilirsiniz.

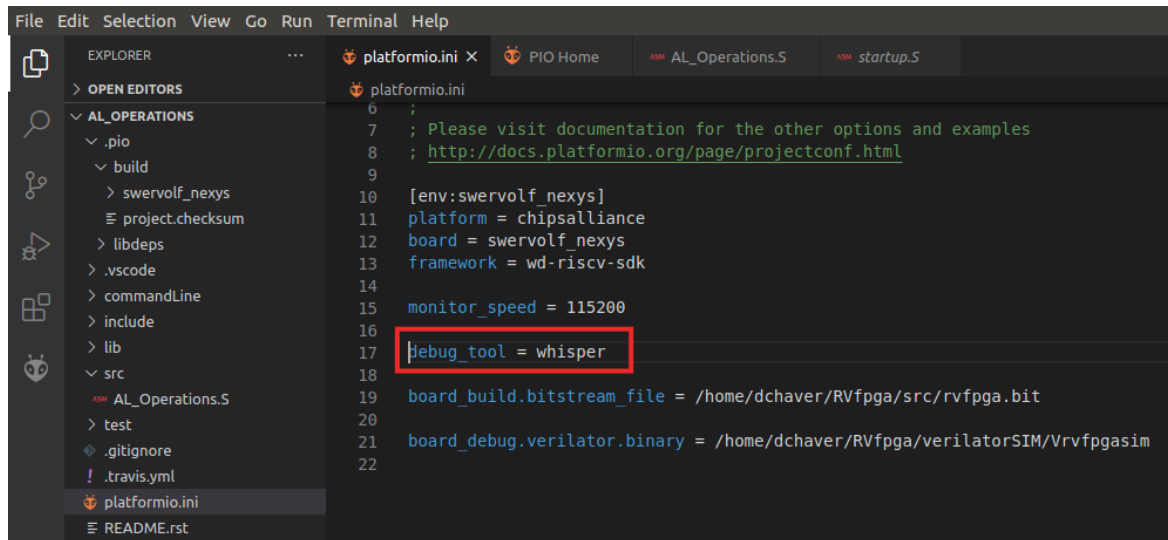
Windows: Bu bölümdeki yönergelerin hepsi Windows'ta çalışır (Whisper'ı Windows'a ilk portlayan Jean-François Monestier'a teşekkür etmek isteriz: <https://jean-francois.monestier.me/porting-western-digital-swerv-iss-to-windows/>). Önemli olarak bir açılan pencere sana Whisper'a Windows güvenlik duvarı üzerinden izin vermeni sorabilir.

macOS: Whisper MacOS'de erişilebilir değildir. Yüksek olasılıkla yakında erişilebilir olup Chips Alliance platformuna eklenecektir.



Whisper, bir komut istemcisi ya da Eclipse ya da PlatformIO gibi bir IDE kullanılarak yürütülebilir. Bu bölümde bir programın PlatformIO'da Whisper'la nasıl simülasyonunun yapılacağını gösteriyoruz. Bu adımları diğer programların simülasyonunu yapmak için de kullanabilirsiniz.

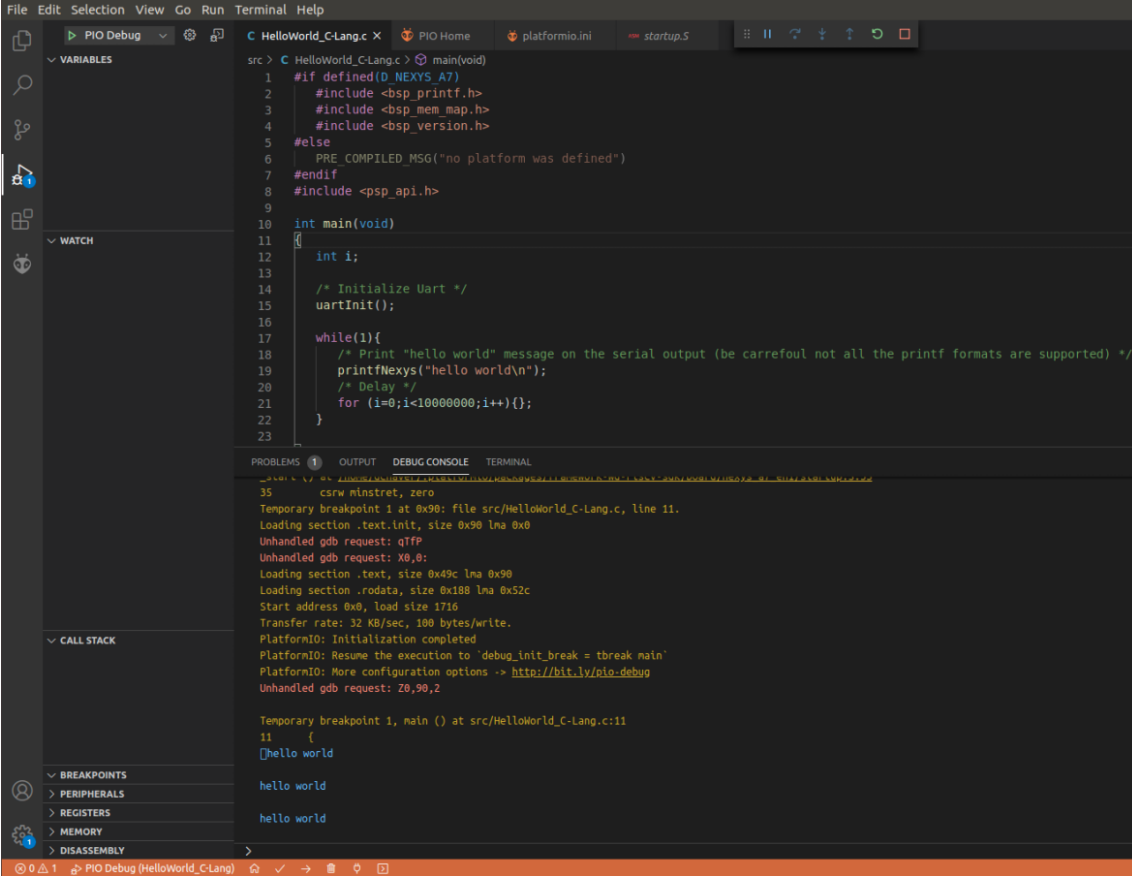
Bölüm 5'te yürütüp ayıkladığınız yalın çevirici programı olan *AL_Operations'u* simüle etmek için Whisper ISS'i kullanarak başlıyoruz (Figür 43'e göz at). Bu kodu Whisper'da çalıştırıp ayıklamak için şu adımları izle:

1. VSCode'u (PlatformIO'yu da) aç. Üst menü çubuğunda, Dosya (File) → Klasörü Aç (Open Folder) üzerine tıklayıp *[RVfpgaPath]/RVfpga/examples/* dizinine git, *AL_Operations* dizinini seç (ancak açma), ardından OK'a tıkla.
2. Dosya (File) → Klasörü Aç (Open File) üzerine tıklayıp *[RVfpgaPath]/RVfpga/examples/AL_Operations/platformio.ini* üzerine çift-tıklayıp, satır 17'yi yorum olmaktan çıkararak **whisper**'ı ayıklama aracı olarak ayarla (Figür 90'a göz at). Dosyayı kaydet (Ctrl-s'e bas).



Figür 90. Satır 17'yi yorum olmaktan çıkar.

3.  üzerine tıklayıp ardından  **PIO Debug** üzerine tıklayarak ayıklayıcıyı başlat.
4. Artık programı Bölüm 5.B'de yaptığın gibi ayıklayabilirsin ancak bu kez program Nexys A7 FPGA kartı yerine Whisper içerisinde simülasyonda çalışıyor.
5. Eğer bir program, HelloWorld_C-Lang örneği (Bölüm 6.F) gibi, *printfNexys* işlevini Whisper'da kullanıyorsa PlatformIO dizesel monitörünü açmamalısın, onun yerine iletiler DEBUG konsolunda gösterilir (Figür 91'e göz at).



```

src > C HelloWorld_C-Lang.c > main(void)
1  #if defined(D_NEXYS_A7)
2  #include <bsp_printf.h>
3  #include <bsp_mem_map.h>
4  #include <bsp_version.h>
5  #else
6  PRE_COMPILED_MSG("no platform was defined")
7  #endif
8  #include <bsp_api.h>
9
10 int main(void)
11 {
12     int i;
13
14     /* Initialize Uart */
15     uartInit();
16
17     while(1){
18         /* Print "hello world" message on the serial output (be careful not all the printf formats are supported) */
19         printfNexys("hello world\n");
20         /* Delay */
21         for (i=0;i<10000000;i++){};
22     }
23
35     csrw minstret, zero
Temporary breakpoint 1 at 0x90: file src/HelloWorld_C-Lang.c, line 11.
Loading section .text.init, size 0x90 lma 0x0
Unhandled gdb request: qTFF
Unhandled gdb request: X0,0
Loading section .text, size 0x49c lma 0x90
Loading section .rodata, size 0x188 lma 0x52c
Start address 0x0, load size 1716
Transfer rate: 32 KB/sec, 100 bytes/write.
PlatformIO: Initialization completed
PlatformIO: Resume the execution to 'debug_init_break = tbreak main'
PlatformIO: More configuration options -> http://bit.ly/pio-debug
Unhandled gdb request: Z0,90,2

Temporary breakpoint 1, main () at src/HelloWorld_C-Lang.c:11
11  {
    [hello world

hello world

hello world

```

Figür 91. HelloWorld_C-Lang örneğinin Whisper'da yürütmesi

9. EKLER

İzleyen ekler Linux'ta yerli (native) RISC-V araç zinciriyle OpenOCD (PlatformIO yerine) nasıl kullanılacağını, Windows'ta PlatformIO kullanarak veri akışını indirmek için sürücülerin nasıl kurulacağını, Windows ile MacOS makinelerde Verilator ile GTKWave'in nasıl kurulacağını, Vivado kullanarak RVfpga'in nasıl programlanacağını gösterir. Tablo 9 bu RVfpga İlk Kullanım Kılavuzunda bulunan bütün ekleri listeler.

Tablo 9. Eklerin Listesi

Ek	Tanım	İşletim Sistemi
A	Yerli RISC-V Araç Zinciri ile OpenOCD'yi Ubuntu 18.04'da Kullanma	Linux
B	PlatformIO'yu kullanmak için Windows'ta sürücülerini kurma	Windows
C	Windows'ta Verilator ile GTKWave kurma	Windows
D	macOS'te Verilator ile GTKWave kurma	macOS
E	RVfpga'i bir FPGA'e indirmek için Vivado kullanma	Windows, Linux
F	RVfpga'i endüstriyel bir IoT uygulamasında kullanma	Hepsi

Ek A, yerel gcc/gdb araçları ve OpenOCD kullanarak programları yerel olarak derlemek ve çalıştırmak/ ayıklamak isteyenler tarafından kullanılmalıdır. Ancak, RVfpga kullanıcılarının bu İlk Kullanım Kılavuzunda açıklandığı gibi **PlatformIO kullanmaları önerilir**.

Windows kullanıcıları **Ek B ile C'deki yönergeleri izlemelidir**. Ek B'deki yönergeler Windows sistemlerin programlarla RVfpga'i Nexys A7 FPGA kartına PlatformIO kullanarak indirebilmek için sürücülerin nasıl indirileceğini gösterir. Ek C Windows kullanıcılarının RVfpga RTL'inin (RVfpga'i tanımlayan Verilog ile SystemVerilog kodu) simülasyonunu yapabilmek Verilator'ün, GTKWave'in nasıl kurulacağını gösterir.

macOS kullanıcıları, Verilator ve GTKWave kullanarak RVfpga RTL'yi simüle etmek için **Ek D'deki yönergeleri izlemelidir**.

RVfpga sistemini Nexys A7 FPGA kartına indirmek için PlatformIO kullanımı önerilir (veri dosyasında tanımlandığı gibi, rvfpga.bit). Bu veri dosyası (bitfile) (rvfpga.bit) Vivado ya da PlatformIO ile oluşturulabilir. Ek E'de tanımlandığı gibi Vivado da RVfpga sistemini Nexys A7 FPGA kartına indirmek için kullanılabilir. Ancak RVfpga'i karta indirmek için Vivado kullanımı **önerilmez** – özellikle **Windows** kullanıcıları için, sürekli sürücü yer değiştirmesi gerektirdiğinden.

Ek A: Yerli RISC-V Araç Zinciri ile OpenOCD'yi Ubuntu 18.04'da Kullanma

PlatformIO kullanımını öneriyor olsak da bu bölümde yerli RISC-V araç zincirini nasıl kurup, çalıştırıp, kullanacağını, OpenOCD'yi RVfpga'yi Nexys A7 FPGA kartına indirmek için kullanacağını, gdb'yi RVfpga'de programları çalıştırıp ayıklamaya kullanacağını gösteriyoruz. Araç zinciri gnu derleyici, ayıklayıcı, çevirici program gibi parçalardan oluşur. RISC-V araç zinciriyle OpenOCD'yi Ubuntu 18.04 işletim sistemi (OS) içerisinde nasıl kuracağını gösteriyoruz ancak bu işlem diğer Linux dağıtımları için de çalışacaktır. Bu yönergeler taze bir Ubuntu sistemini varsayar.

Bu kılavuzda önceden tanımlandığı gibi eğer PlatformIO kullanıyorsan bu adımlar gerekmez. PlatformIO, Vivado ile Verilator ya da Whisper kullanımı, RISC-V programlarının çalıştırılması, ayıklanması, simülasyonu yapılması için önerilen yöntemdir, ancak bu yönergeler yerli RISC-V araç zinciriyle OpenOCD'yi PlatformIO ile Vivado Donanım Yöneticisi yerine kullanmaya ilgilenen kişiler için sağlanmıştır.

I. Bir Linux Ubuntu OS'te yerli (native) kurulum

Bu bölümde, Ubuntu 18.04 makineye RISC-V araç zinciri, OpenOCD ve Whisper'ın nasıl indirileceğini anlatıyoruz. Bu araçlar yalnızca PlatformIO'nun yerine geçer; Vivado ve Verilator'un yüklenmesi, bu GSG'nin Bölüm 5'inde açıklandığı gibi gereklidir.

RISC-V Araç Zinciri

Burada eksiksiz RISC-V Araç Zincirini bilgisayarında nasıl kuracağını gösteriyoruz – bir diğer deyişle gnu derleyici, ayıklayıcı gibi. Kurulum yönergelerini RISC-V International şurada sağlar: <https://github.com/riscv/riscv-gnu-toolchain>. Bu yönergeler aşağıda özetlenmiştir.

ÖNEMLİ: RISC-V araç zinciriyle OpenOCD'yi kurmak birkaç saat sürebilir – çoğunluğu araç zinciri inip, derlenip kurulurken geçer

Bir terminalde şunu yaz (bu işlem bir saatten çok sürebilir, ancak çoğunluğu programlar indirilip kurulurken geçer):

- `sudo apt-get install git autoconf automake autotools-dev curl libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex texinfo gperf libtool patchutils bc zlib-dev libexpat-dev`
- `git clone --recursive https://github.com/riscv/riscv-gnu-toolchain`
- `cd riscv-gnu-toolchain/`
- `./configure --prefix=/opt/riscv --with-arch=rv32imc`
- `sudo make` (Oluysa derleme süresini çok azalttığı için `sudo make -j$(nproc)` kullan)
- `export PATH=$PATH:/opt/riscv/bin` (sistemindeki yolu değiştir)

OpenOCD

OpenOCD kullanıcıların gömülü hedef aygıtları programlayıp ayıklamasını sağlayan bir açık yongada ayıklayıcıdır. Şu adımları izleyerek RISC-V OpenOCD'yi bilgisayarına kur:

- `sudo apt-get install libusb-1.*`
- `sudo apt-get install pkg-config`
- `git clone https://github.com/riscv/riscv-openocd.git`
- `cd riscv-openocd/`
- `./bootstrap`
- `./configure --prefix=/opt/riscv --program-prefix=riscv- --enable-ftdi`

- ```
--enable-jtag_vpi
```
- make
  - sudo make install

### **Whisper**

Whisper'ı bilgisayarına kurmak için şu adımları izle (yönergeler şuradadır: <https://github.com/chipsalliance/SweRV-ISS> ancak aşağıda özetlenmiştir):

- apt-cache policy libboost-all-dev
- sudo apt-get install libboost-all-dev
- cd [RVfpgaPath]
- git clone <https://github.com/chipsalliance/SweRV-ISS>
- cd SweRV-ISS
- make BOOST\_DIR=/usr/include/boost
- export PATH=\$PATH:[RVfpgaPath]/SweRV-ISS/build-Linux ([RVfpgaPath]'i gereğince değiştir).

## **II. OpenOCD kullanarak Nexys A7 FPGA kartını kullanarak RVfpga'de bir program yürütme**

### **Adım A. RVfpga', (Figür 24) Nexys A7'ye indir**

1. RVfpga için veri dosyasını içeren proje dizinine git:  
`cd [RVfpgaPath]/RVfpga/src`
2. OpenOCD kullanarak RVfpga'i karta indir:  
`riscv-openocd -c "set BITFILE rvfpga.bit" -f  
SweRVolfSoC/OtherSources/swervolf_nexys_program.cfg`

### **Adım B. Anahtarları okuyup durumlarını LEDlere yazdıran LedsSwitches programını yürüt**

3. LedsSwitches/commandLine dizinine git:  
`cd [RVfpgaPath]/RVfpga/examples/LedsSwitches/commandLine`  
  
Klasörde kaynakları derlemek için Makefile, bağlama senaryosu, bir python senaryosu, *LedsSwitches.S* programına bir sembolik bağ bulacaksın.
4. .elf dosyasını kurgula:  
`make clean  
make LedsSwitches.elf`
5. OpenOCD'yi SoC'ye bağla:  
`riscv-openocd -f  
../../../../src/SweRVolfSoC/OtherSources/swervolf_nexys_debug.cfg`

OpenOCD çalışmaya başlayınca şunu da içeren birkaç ileti göreceksin:  
Info : Listening on port 4444 for telnet connections

6. Yeni bir terminal, açık program dizinine gidip (`cd [RVfpgaPath]/RVfpga/examples/LedsSwitches/commandLine`), şu komutu çalıştır:  
`telnet localhost 4444`



Ardından, telnet bağlantısının içerisinde, şunu yaz:

```
load_image LedsSwitches.elf
reg pc 0
resume
```

Bu üç komut (1) LedsSwitches.elf programını RVfpga'e yükler, (2) program sayacını (PC) 0'a ayarlar (programın ilk yönergesinin adres yeri), (3) yürütmeyi sürdürür.

Program Nexys A7 FPGA kartına Adım 2'de yüklenmiş olan RVfpga üzerine çalışmaya başlayacaktır. Program LEDlere anahtarların durumunu gösterir. Anahtarlara bastıkça LEDlerin anahtarların değerini yansıtacak biçimde değişecektir.

### **Adım C. Yalın aritmetik-mantık işlemlerini yürüten AL Operations CommandLine programını ayıkla**

Şimdi bir başka programın (AL\_Operations\_CommandLine) OpenOCD ile gdb kullanılarak nasıl ayıklanacağını göstereceğiz.

7. OpenOCD bağlantısını açık tut (Adım 5'e göz at).
8. telnetin çalıştığı diğer terminalde (Adım 6'dan), şunu yazarak telnet bağlantısından çık:  

```
exit
```

9. AL\_Operations/commandLine barındıran proje dizinine geçiş:  

```
cd ../../AL_Operations/commandLine
```

Klasörde kaynakları derlemek için Makefile, bağlama senaryosu, bir python senaryosu, *AL\_Operations.S* programına bir sembolik bağ bulacaksınız.

10. .elf dosyasını kurgula:  

```
make clean
make AL_Operations.elf
```
11. Ardından bu terminalde şunu yazarak gdb'yi başlat:  

```
riscv32-unknown-elf-gdb AL_Operations.elf
```

12. gdb konsolunda şunu yaz:  

```
target remote localhost:3333
load
```

Bu OpenOCD'ye bağlanıp *AL\_Operations.elf* programını belleğe yükleyecektir.

13. Şimdi programı ayıklayabiliyor olmalısın. Şu diziye yazıp çıktıları çözümü:

i. 

```
disas 0,20
```

Bu (adres 20'yi içermeden) adres 0'dan 20'ye çevirici kodunu gösterir.

```
(gdb) disas 0,20
Dump of assembler code from 0x0 to 0x14:
=> 0x00000000 <_start+0>: li t3,0
 0x00000004 <REPEAT+0>: addi t3,t3,6
 0x00000008 <REPEAT+4>: addi t3,t3,-1
 0x0000000c <REPEAT+8>: andi t3,t3,3
 0x00000010 <REPEAT+12>: beqz zero,0x4 <REPEAT>
End of assembler dump.
```

Figür 92. Çevirici programını gör

ii. `i r t3`

Bu `t3` yazmacının içeriğini gösterir. Uzun sürümünü de yazabilirsiniz: `info reg t3`.

```
(gdb) i r t3
t3 0x0 0
```

Figür 93. `t3` yazmacında barındırılan değeri yazdır

iii. `i r pc`

Bu program sayacının (`pc`) içeriğini gösterir.

```
(gdb) i r pc
pc 0x0 0x0 <_start>
```

Figür 94. İlk yönergeyi gösteren PC yazmacındaki değeri yazdır

iv. `stepi`  
`i r t3`  
`stepi`  
`i r t3`  
`stepi`  
`i r t3`  
`stepi`  
`i r t3`

`stepi` programa bir yönerge yürüttür. `i r t3` ardından `t3` yazmacının içeriğini gösterir.

```
(gdb) stepi
0x00000004 in REPEAT ()
(gdb) i r t3
t3 0x0 0
(gdb) stepi
0x00000008 in REPEAT ()
(gdb) i r t3
t3 0x6 6
(gdb) stepi
0x0000000c in REPEAT ()
(gdb) i r t3
t3 0x5 5
(gdb) stepi
0x00000010 in REPEAT ()
(gdb) i r t3
t3 0x1 1
```

Figür 95. Birkaç yönergeyi bir bir yürütüp `t3` yazmacına bak

gdb'yi kullanarak programla yazmaçları ayıklamayı, gezmeyi bitirdiğinde gdb terminalinde **quit** yazarak gdb'den çıkıp OpenOCD terminalinde **^C** yazarak OpenOCD'den çık.

### III. Verilator kullanarak bir programı RVfpga'de simülasyonunu yapma

1. Ubuntu'da bir terminal aç
2. Bir terminal penceresinde şu komutları yürüterek simülatör ikili dosyasını oluştur:

```
cd [RVfpgaPath]/RVfpga/verilatorSIM
make clean
make
```

Dosya *Vrvfpgasim* (RVfpga simülasyon ikili dosyası)  
[RVfpgaPath]/RVfpga/verilatorSIM dizininin içerisinde oluşturulmuş olmalı.

3. Örnek programı barındıran klasöre git:  
`cd [RVfpgaPath]/RVfpga/examples/AL_Operations/commandLine`

4. Simülasyon için onaltılık programı oluştur.

```
make clean
make AL_Operations.elf
make AL_Operations.bin
make AL_Operations.vh
```

5. Simülatörü yürüt.

```
../../../../verilatorSIM/Vrvfpgasim
+ram_init_file=AL_Operations.vh +vcd=1
```

Birkaç saniye sonra terminalde **^C** yazarak simülasyonu durdur. Dosya *trace.vcd* oluşturulmuş olmalı, *GTKWave* ile açabilirsin.

```
gtkwave trace.vcd
```

6. Çizgeye sinyaller ekleyip bunları çözümlemek için Bölüm 7'nin adım 8'inden 12'sine yönergeleri izle.

### IV. Bir programın Whisper'da simülasyonunu yapma

1. Ubuntu'da bir terminal aç
2. Örnek programı barındıran klasöre git:  
`cd [RVfpgaPath]/RVfpga/examples/AL_Operations/commandLine`
3. Tersine çeviri programını oluştur.  
`make AL_Operations.dis`
4. *AL\_Operations.dis*'i bir editörde aç. Göreceğin şudur:

```
<_start>:
 0: 00000e13 li t3,0
<REPEAT>:
 4: 006e0e13 addi t3,t3,6
 8: fffe0e13 addi t3,t3,-1
```

```

c: 003e7e13 andi t3,t3,3
10: fe000ae3 beqz zero,4 <REPEAT>
14: 00000013 nop

```

##### 5. Simülatörü etkileşimli modda yürüt.

```
whisper --interactive AL_Operations.elf
```

##### 6. Programı ayıkla.

```

whisper> step
#1 0 00000000 00000e13 r 1c 00000000 addi x28, x0, 0x0

whisper> peek r x28
0x00000000

whisper> step
#2 0 00000004 006e0e13 r 1c 00000006 addi x28, x28, 0x6

whisper> peek r x28
0x00000006

whisper> step
#3 0 00000008 fffe0e13 r 1c 00000005 addi x28, x28, -0x1

whisper> peek r x28
0x00000005

whisper> step
#4 0 0000000c 003e7e13 r 1c 00000001 andi x28, x28, 0x3

whisper> peek r x28
0x00000001

```

Whisper kullanarak ayıklamayı, programı, yazmaçları gezmeyi bitirdiğinde terminalde **quit** yazarak çık.

## Ek B: PlatformIO'yu kullanmak için Windows'ta sürücülerini kurma

Zadig yürütülebilirini indirmek için şu websitesine git (Figür 96):

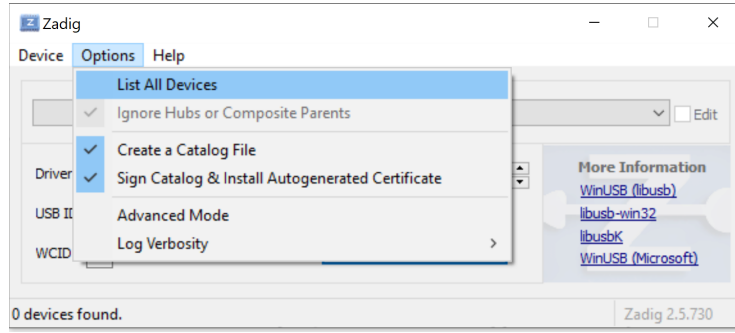
<https://zadig.akeo.ie/>



Figür 96. PlatformIO'nun kullandığı Nexys A7 kart sürücüsünü kur

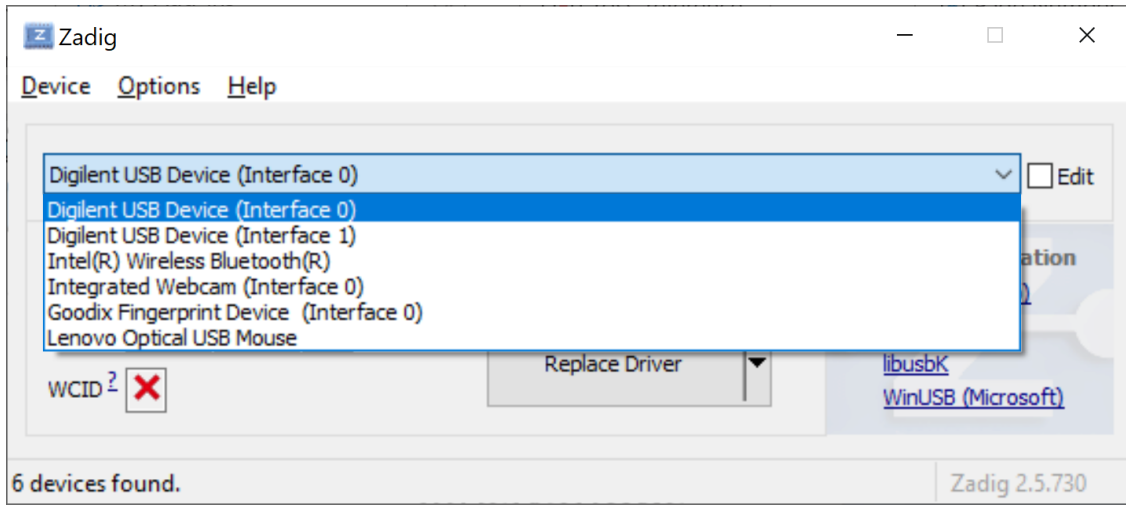
Zadig 2.5 üzerine tıklayıp yürütülebilirini kaydet. Ardından indirdiğin yerden çalıştır (zadig-2.5.exe). Bulmak için Başlat menüsüne zadig de yazabilirsin. Yüksek olasılıkla Zadig'e bilgisayarına değişiklikler yapma ile güncelleme konusunda izin verip vermediğin sorulacaktır. İkisinde de Yes (Evet) tıkla.

Nexys A7 Kartını bilgisayara bağlayıp aç. Zadig'de, Options (Seçenekler) → List All Devices (Bütün Aygıtları Listele) (Figür 97).



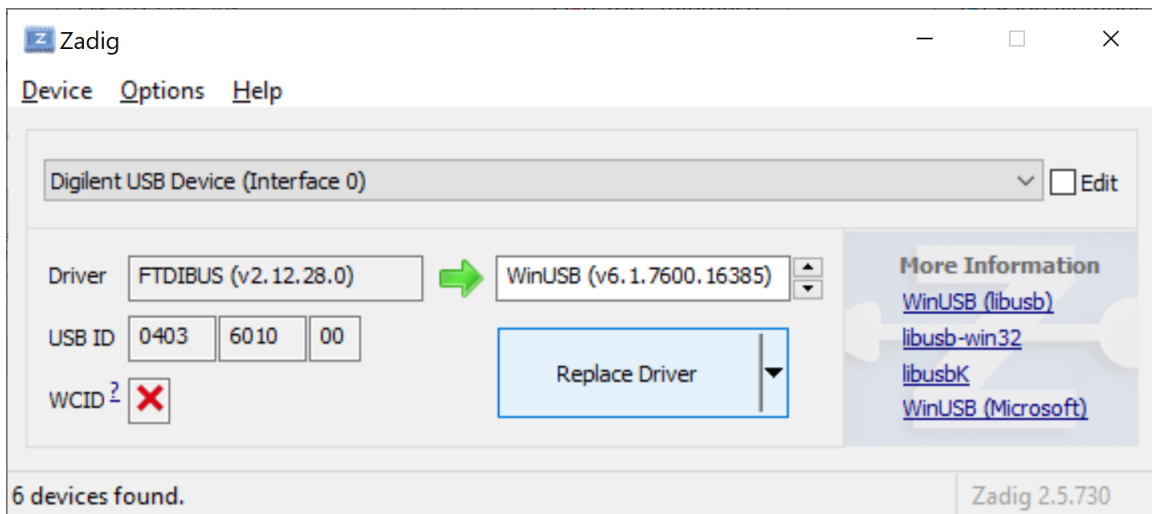
**Figür 97. List all devices in Zadig**

Açılan menüye tıklayınca Digilent USB Device (Arayüz 0), Digilent USB Device (Arayüz 1) listelenir. Yalnızca Digilent USB Aygıtı (Arayüz 0) için yeni sürücüler yükle (Figür Figür 98



**Figür 98. Digilent USB Aygıtı (Interface 0) için WinUSB sürücüsünü kur**

Şimdi FTDI sürücüsünü WinUSB sürücüsüyle değiştireceksin, Figür 99'da gösterildiği gibi. Diligent USB Device (Interface 0) için Sürücüyü Değiştir (ya da Sürücüyü Kur) üzerine tıkla. Nexys A7 kartı için sürücüyü kuruyorsun, ya da önceden Vivado'yu kurduysan, Vivado'nun kullandığı FTDI sürücüsünü PlatformIO'nun kullandığı WinUSB sürücüsüyle değiştiriyorsun.



**Figür 99. Nexys A7 kartı için sürücüyü değiştir**

Bir süre sonra, genellikle birkaç dakika sonra, Zadiğ sürücünün doğru yüklü olduğunu gösterir. Close'a (Kapat) tıklayıp, ardından Zadiğ penceresini kapat.

PlatformIO'yu sonraki kullanışında sürücüyü yeniden kurman gerekmeyecek. Ancak, önemli olarak, **bu sürücü Windows'ta Vivado ile uyumlu değildir**. Dolayısıyla artık veri dosyalarını FPGA kartına indirmek için Vivado'yu kullanamazsın. Eğer veri dosyalarını indirmek için Vivado'yu kullanmak istersen (önerilmez) sürücüleri Vivado'yla indirilen orijinal sürümüne çekmen gerekir, Ek E'de tanımlandığı gibi.

## Ek C: Windows'ta Verilator ile GTKWave kurma

Bu bölümde, Windows 10'a Verilator ve GTKWave'in nasıl yükleneceğini açıklıyoruz. Windows'ta Verilator'u yüklemek için Cygwin'i kullanmak gerekir, bu nedenle önce bu programlama/çalışma zamanı ortamının nasıl yükleneceğini açıklıyoruz.

### Cygwin kurulumu:

Web sayfasında tanımlandığı gibi (<https://www.cygwin.com>) Cygwin, Windows'ta bir Linux dağıtımındaki gibi GNU ile Açık Kaynak araçlardan oluşur. Cygwin'i Windows 10'da kurmak için şu adımları izle.

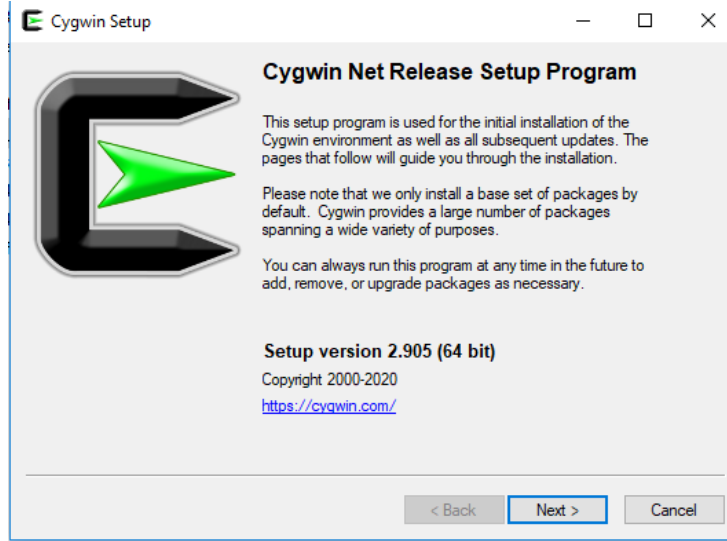
1. Kurulum web sayfasına gidip (<https://cygwin.com/install.html>) kurulum dosyasını indir, *setup-x86\_64.exe* (Figür 100).



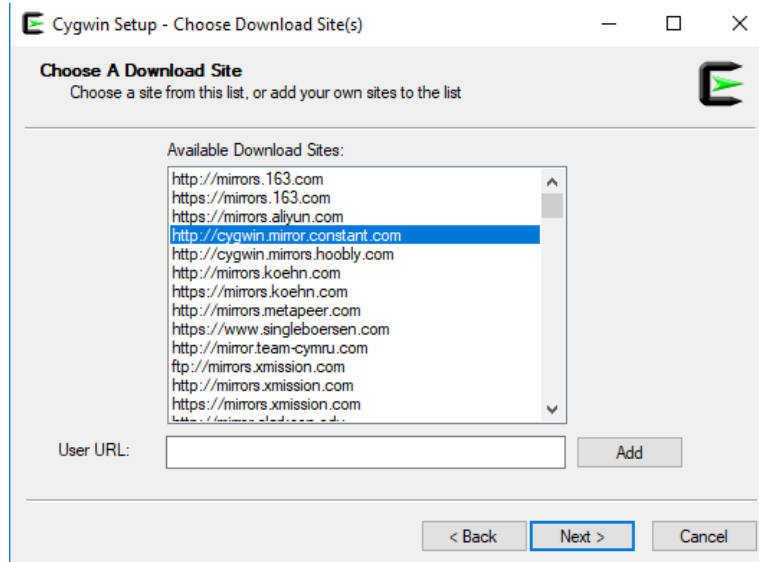
**Figür 100. Cygwin kurulum websayfası**

2. Kurulum dosyasına çift-tıklayarak makinende yürüt (Figür 101). Varsayılan seçenekleri koruyarak **Next (İleri)** butonuna birkaç kez tıkla. Kurucu **Choose a Download Site (bir İndirme Sitesi Seç)** (Figür 102) seçmeni soracaktır, istediğini seçebilirsin.



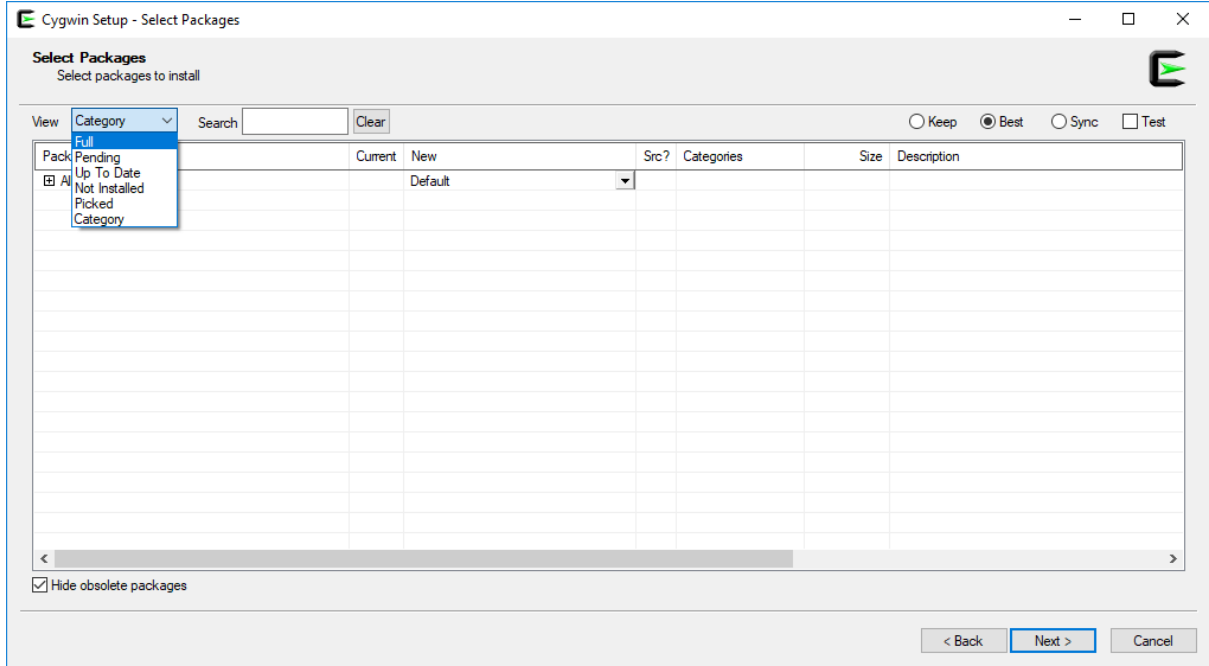


**Figür 101. Cygwin installation window**



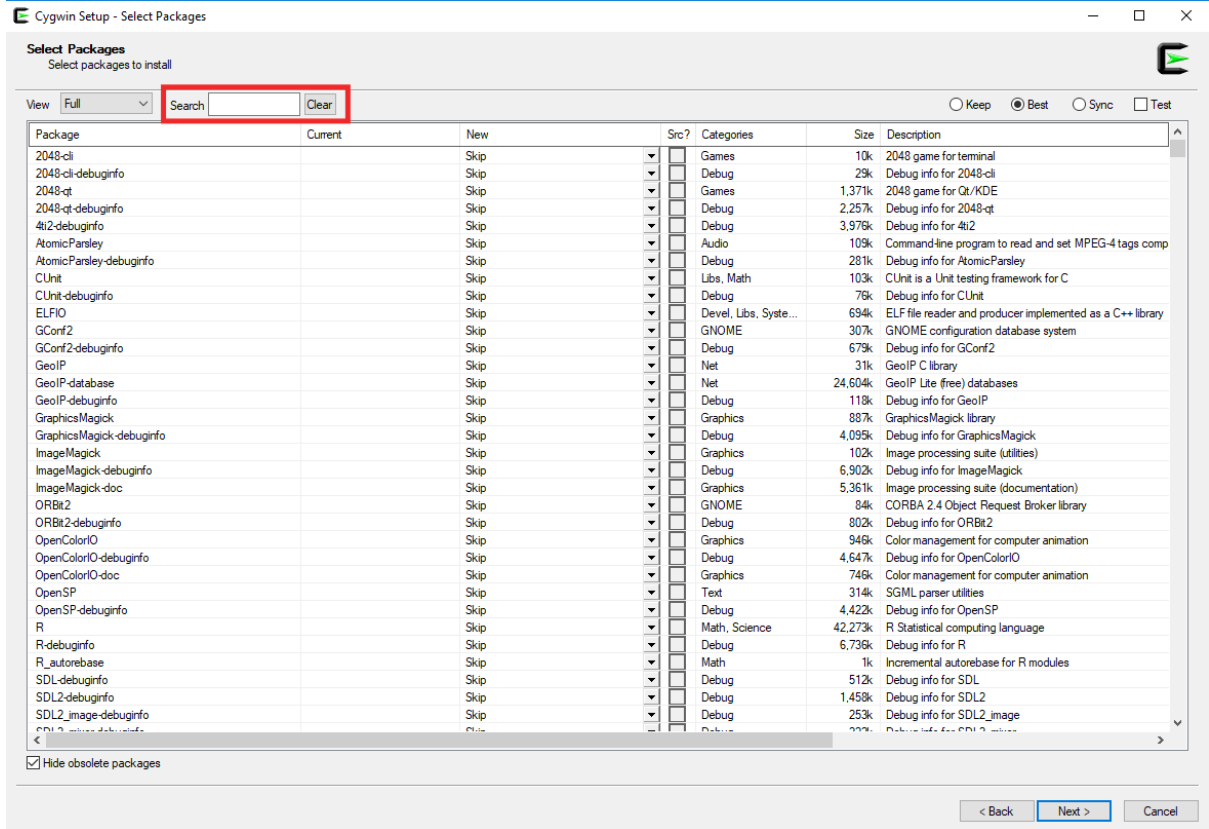
**Figür 102. İndirme Sitesini Seç**

3. Birkaç adım sonra **Select Packages (Paketleri Seç)** penceresine (Figür 103) ulaşacaksın. **Full (Bütün)** görünümü seç, Figür 103'de gösterildiği gibi.



**Figür 103. Paketleri Seç penceresi**

4. Kurabileceğin paketlerin bütün listesi belirecek (Figür 104). **Arama** kutusunda kurmak istediğin paketleri seç.



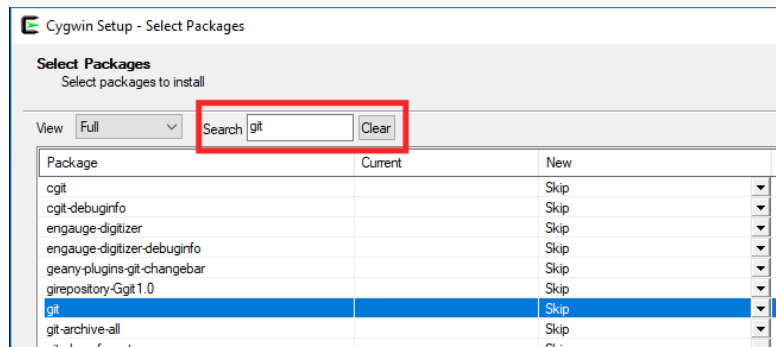
**Figür 104. Paketleri Seç penceresi – Bütün görünüm**

Verilator'u derleyip yeni simülör ikili verisi oluşturabilmek için şu paketleri kurman gerekecek:

- git
- make
- autoconf
- gcc-core
- gcc-g++
- flex
- bison
- perl
- libargp-devel

Cygwin kurulumunda en azından bu paketleri ier. Ŗu adımları izleyerek bir bir se (yalnızca git iin detaylı adımları gsteriyoruz; diğerklerinin adımları aynı):

- **Arama** kutusunda git paketini ara (Figr 105).



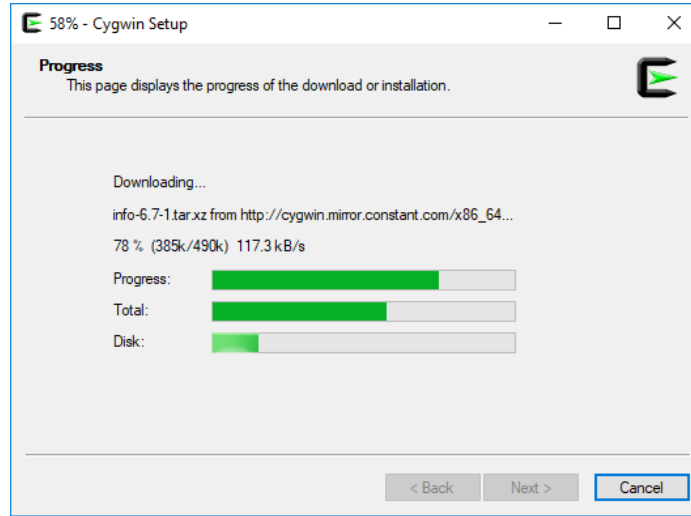
**Figr 105. git paketini ara**

- En gncel srm aılır menden seip kutuyu tikle (Figr 106).

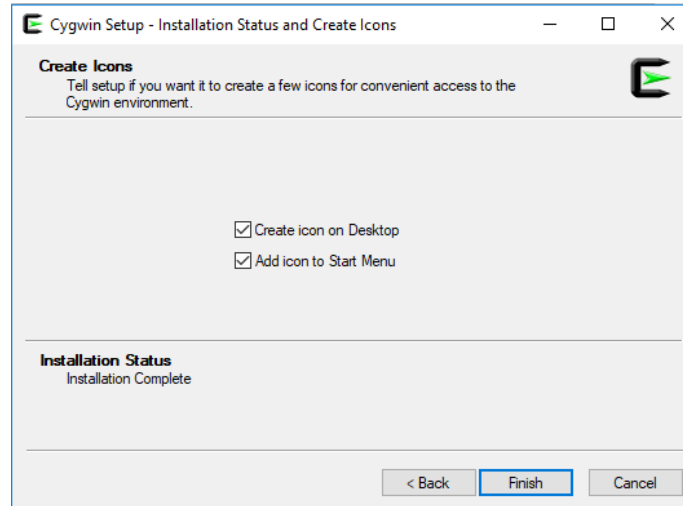


**Figr 106. En gncel srm seip kutuyu iřaretle**

- Yukarıdaki listedeki paketlerin kalanı iin bu adımları yeniden izle.
5. Dokuz paketi setikten sonra Cygwin kurulumunda bu paketleri iermek iin sonraki pencerede **Next (İleri)** tıklayıp (kurulum sreci, Figr 107, birkaç dakika srebilir) Finish (Bitir) tıklayarak kurulumu bitir (Figr 108).



**Figür 107. Cygwin kurulum**



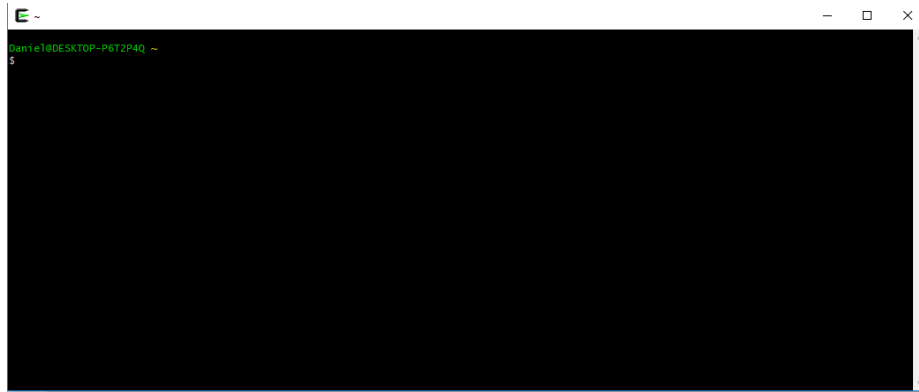
**Figür 108. Kurulumu bitir**

6. Cygwin kurulumun için bir paket eklemen gerekirse adımlar 2-5'i o paket için tekrarla.

### **Verilator kurulumu:**

Windows 10'da Verilator'ü kurmak için şu adımları izle.

1. Windows Desktop (Masaüstü) ya da Start (Başlat) menüsünden erişilebilen Cygwin terminali aç (Figür 109).



**Figür 109. Cygwin terminali**

2. Şu adımları izleyerek Verilator'ü kurgulayıp kurun. Bu bilgisayarınızın hızına bağlı olarak belli bir süre (belki saatler) alacaktır:

- `git clone https://git.veripool.org/git/verilator`
- `cd verilator`
- `git pull`
- `git checkout v4.020`
- `autoconf`
- `./configure`
- `make`
- `make install`

### **GTKWave kurulumu:**

GTKWave şuradan önden derlenmiş bir paket olarak indirilebilir <https://sourceforge.net/projects/gtkwave/files/>. En yeni Windows paketini ara (bu doküman yazılırken **gtkwave-3.3.100-bin-win64** idi), indir, zipten çıkar (sıkıştırmayı yok et. *bin* klasörünün içerisinde *gtkwave* diye bir yürütülebilir dosya bulabilirsin, bu Windows makinende yürütölüp kullanılabilir.

## Ek D: macOS'te Verilator ile GTKWave kurma

Bu bölümde macOS'de Verilator ile GTKWave nasıl kurulacağını açıklıyoruz. Bu yönergeler macOS Catalina 10.15.6 ile denenmiştir ancak OS'in diğer sürümleriyle de çalışması beklenir. Homebrew (<https://brew.sh/>) paket yöneticisi kurulum için kullanılır. Bunlar gibi adımlar MacPorts için de bulunabilir, macOS'teki diğer yaygın paket yöneticisi (<https://www.macports.org/>).

### gcc kurulumu:

Verilator ile yeni bir simülatör kurgulamak için sisteminde bir derleyici araç zinciri kurulu olmalıdır. Düzgün bir derleyici kurmanın türlü yolları vardır. İkisini aşağıda veriyoruz:

1. XCode Command Line Tools kur. Önemli olarak bu LLVM kuracaktır, ancak bir *gcc* komutu kurulumdan sonra nasılsa erişilebilir olacaktır. Bunun için şunu bir Terminal penceresine yaz:

- `xcode-select -install`

2. Homebrew kullanarak *gcc*'yi kur. Şu recipe'yi kullan:

- o `brew install gcc@9`

### Verilator kurulumu:

Homebrew'le Verilator'ü açık bir Terminal'e şu komutu yazarak kurabilirsin:

- `brew install verilator`

### gtkwave kurulumu:

*gtkwave* kurmak için yine Homebrew kullanacağız. Ancak bu kez *cask* kullanmamız gerekecek, nedeni ise GUI macOS uygulaması olması. Açık bir Terminal'de şu komutları yaz:

- `brew tap homebrew/cask`
- `brew cask install xquartz`
- `brew cask install gtkwave`

Kurulumdan sonra Uygulamalar klasöründe *gtkwave.app* için bir ikon belirmeli. Komut istemcisinden kullanmak için Perl'in Switch modülünü kurman gerekebilir:

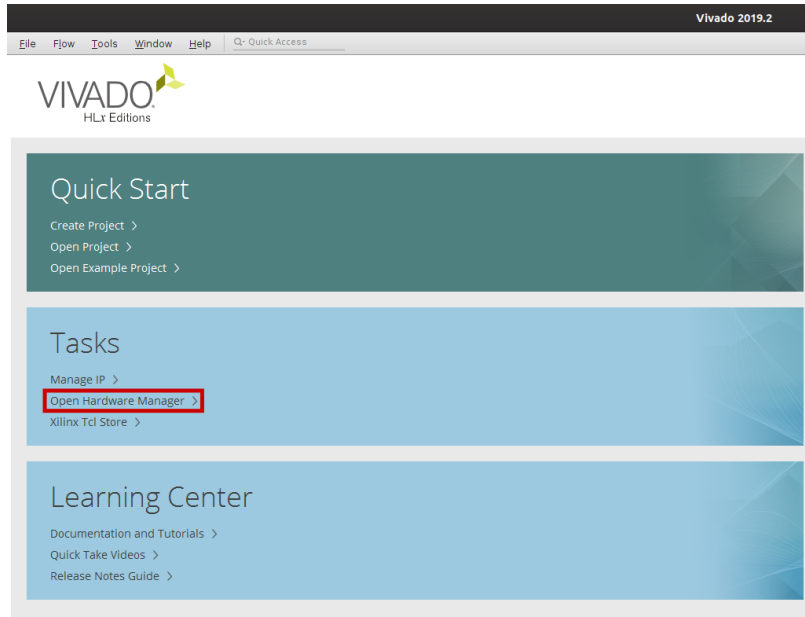
- `cpan install Switch`

## Ek E: RVfpga'i bir FPGA'e indirmek için Vivado'yu Kullanma

Vivado'yu kullanarak FPGA'i RVfpga SoC'si ile programlamak için şu adımları izle:

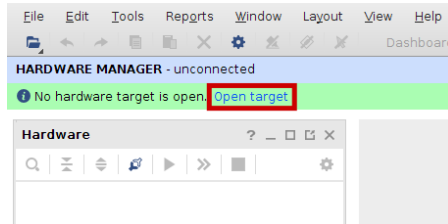
**WINDOWS:** Bu adımları izlemeden önce Windows'ta sürücülerini Vivado'nun kullandıklarına geri çekmek gerekir, nasıl yapılacağı bu ekin sonunda açıklanmıştır (Ek E).

- Nexys A7 kartını bilgisayarına bağla.
- Nexys A7 kartını sol üstteki anahtarlar çalıştır.
- Vivado 2019.2'yi aç.
- Vivado'daki *Donanım Yöneticisini* aç (Figür 110).



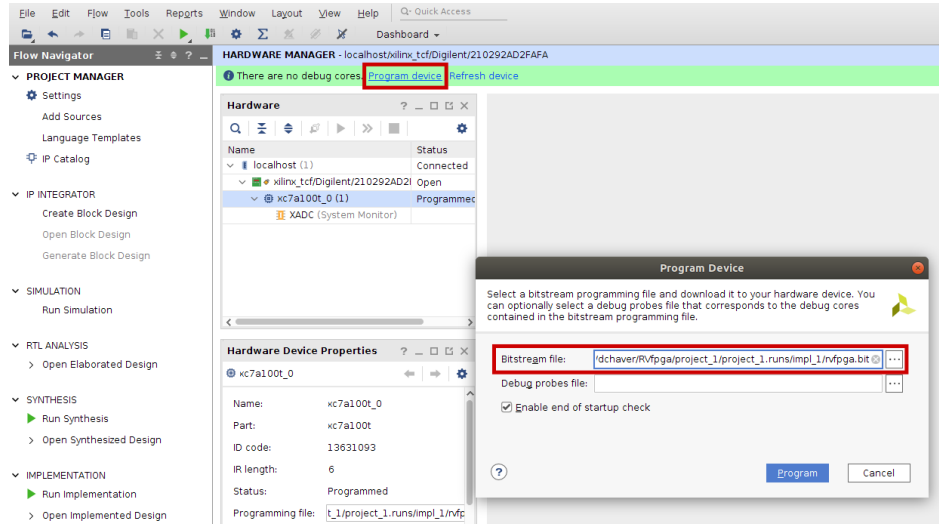
**Figür 110. Donanım Yöneticisini aç**

- Donanım Yöneticisi açılıp seni açık hedef donanım olmadığı konusunda bilgilendirilecektir. *Open target – Auto connect* (Figür 111) tıklayarak hedefi aç.



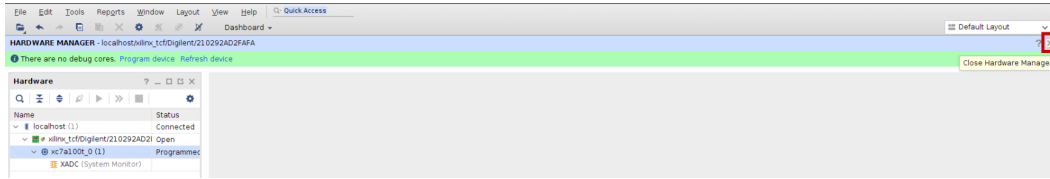
**Figür 111. Hedefi aç**

- Program device seç*
- Figür 112'de** görüldüğü gibi.
- Şimdi RVfpga'in SweRVolf SoC sürümünü FPGA'e yükleyeceksin. Yeni pencerede *Bitstream file'ı* [RVfpgaPath]/RVfpga/src/rvfpga.bit olarak seç. *Program* tıkla.



**Figür 112. Aygıtı programla**

- i. Birkaç saniye sonra FPGA RVfpga, **FPGA'e hedefli SweRVolf SoC'si**, ile programlanmış olacak (Figür 24).
- j. Son olarak Vivado'nun kartı bırakması için Vivado'nun Donanım Yöneticisi panelinin sağ üstündeki X butonuna tıklayarak **Donanım Yöneticisini kapat** (Figür 113).

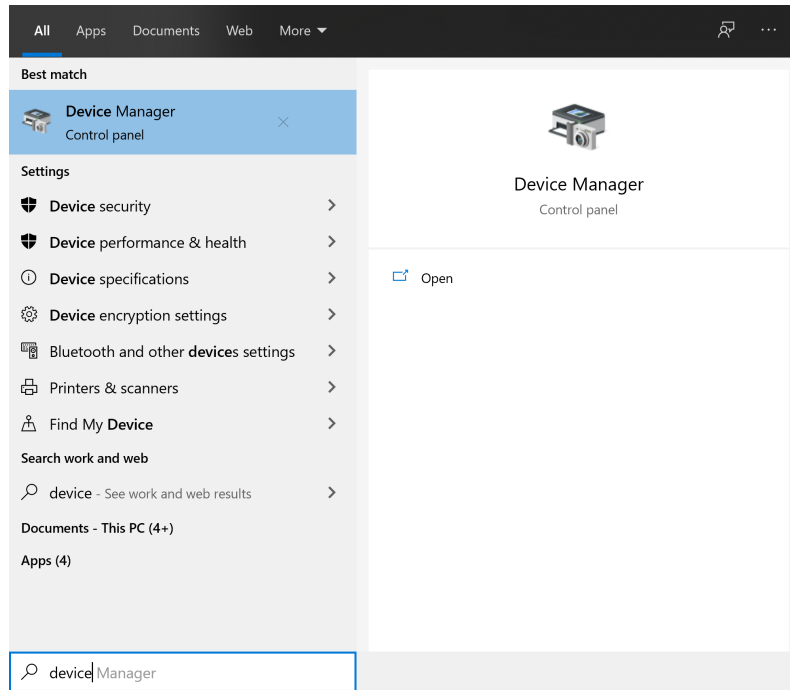


**Figür 113. Donanım Yöneticisini Kapat**

## Sürücüler Windows'ta Vivado'nun kullandıklarına nasıl geri döndürülür

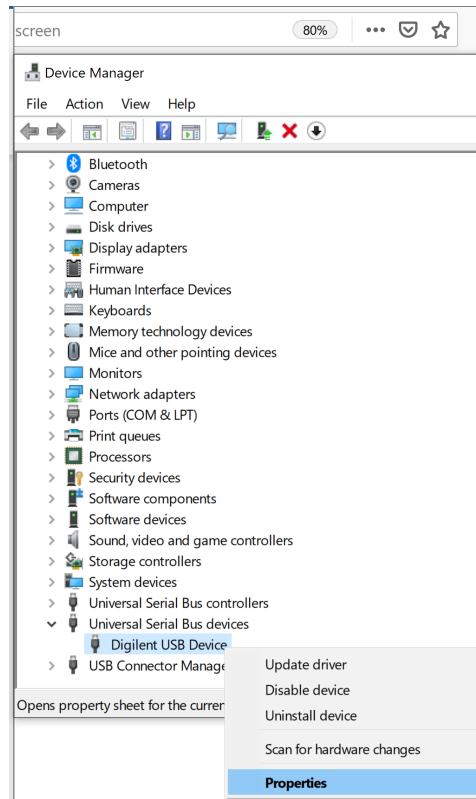
Windows'ta Nexys A7 FPGA kartının sürücüleri Vivado ile PlatformIO'da ayrıdır. **Bu GSG'nin Bölüm 5.A'sında açıklandığı gibi FPGA'i programlama için PlatformIO'yu kullanman önerilir.** Ancak veri dosyalarını indirmek için Vivado'yu kullanmak istiyorsan Ek B'de kurduğun sürücüleri Nexys A7 FPGA kartı için Vivado (FTDI) sürücülerine döndürmen gerekir. Bunun için Aygıt Yöneticisini Başlat menüsüne tıklayıp, arama kutusuna aygıt yöneticisi yazıp, Aygıt Yöneticisine tıklayarak aç (Figür 114).





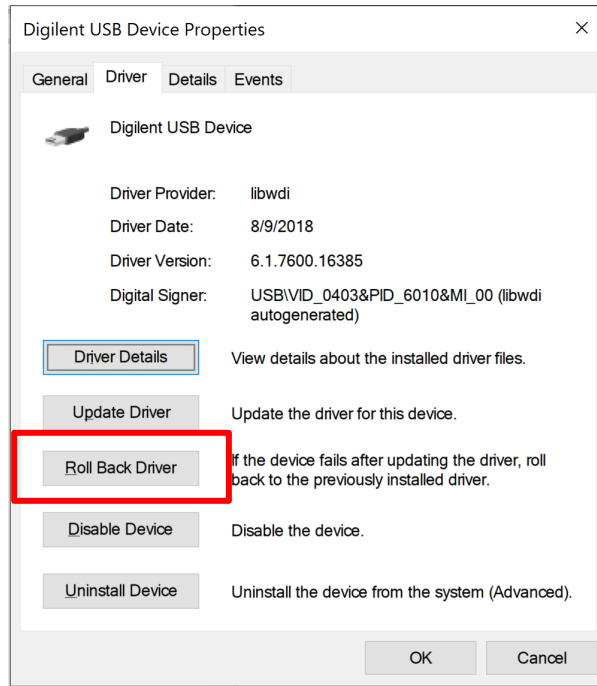
**Figür 114. Cihaz Yöneticisini aç**

Sonra Evrensel Dizisel Veri Yolu Aygıtlarını genişlet, Diligent USB Device'a sağ-tıkla, Özellikleri seç (Figür 115).



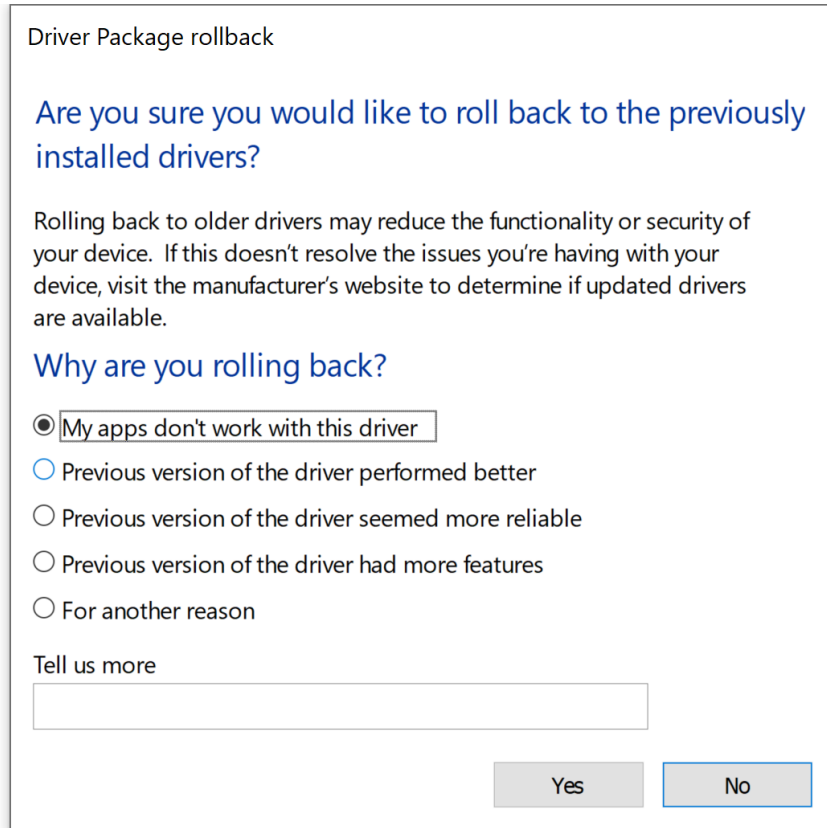
**Figür 115. Diligent'in Nexys A7 FPGA kartı için sürücü özelliklerini aç**

Özellikler penceresinde Sürücüler sekmesine tıklayıp Sürücüyü Geri Al seç (Figür 116).



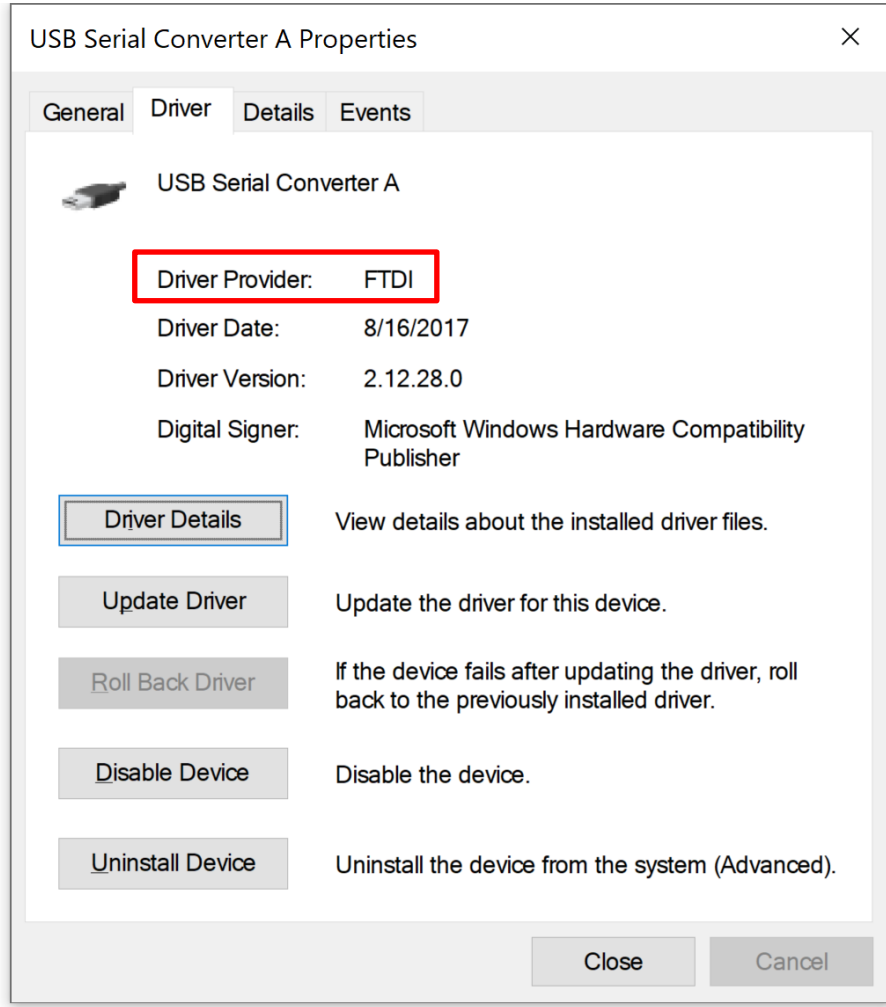
**Figür 116. Sürücüyü geriye alma**

Neden sürücüyü geriye aldığını soran bir pencere açılacaktır. Bir neden seçip Yes (Evet) tıkla (Figür 117).



**Figür 117. Geriye almayı onayla**

Sürücü önceki sürücüyü geri alındıktan sonra Sürücü Sağlayıcısı FTDI olarak listelenmeli (Figür 118'e göz at).



**Figür 118. FTDI sürücüsü sağlanan sürücü olarak gösterilir**

Şimdi Vivado'yu kullanarak veri dosyalarını FPGA'e yükleyebilirsiniz. Ancak yine PlatformIO'nun programı RVfpga'e indirebilmesi için Nexys A7 kartının sürücüsünü değiştirmek için Zadig'i kullanman gerekecek. Dolayısıyla veri dosyalarını indirmek için de PlatformIO'yu kullanman önerilir (Vivado yerine) – böylelikle sürekli sürücü değiştirmen gerekmez.

## Ek F: RVfpga'i endüstriyel bir IoT uygulamasında kullanma

Temmuz 2020'de, Daniel León González, University Complutense of Madrid'de bir yüksek lisans öğrencisi "FPGA implementation of an ad-hoc RISC-V system-on-chip for industrial IoT" başlıklı yüksek lisans tezini bitirdi. Bu çalışma RVfpga'in gerçek bir endüstriyel IoT uygulamasında kullanımını gösteriyor. Proje özetini aşağıda sağlıyoruz, tezin bütününe ise şuradan erişilebilir: [https://eprints.ucm.es/62106/1/DANIEL\\_LEON\\_GONZALEZ\\_DL\\_-\\_FPGA\\_Implementation\\_of\\_an\\_ad-hoc\\_RISC-V\\_SoC\\_for\\_Industrial\\_IoT\\_Graded\\_4286351\\_962908330.pdf](https://eprints.ucm.es/62106/1/DANIEL_LEON_GONZALEZ_DL_-_FPGA_Implementation_of_an_ad-hoc_RISC-V_SoC_for_Industrial_IoT_Graded_4286351_962908330.pdf).

### Endüstriyel IoT için geçici bir RISC-V yongadaki sistemin FPGA gerçekleştirilmesi

**Özet:** IoT için düğüm cihazlarının enerji açısından verimli, uygun maliyetli olması gerekir, ancak çok sayıda senaryoda yüksek bir hesaplama gücü gerektirmezler. Bu, büyük sensor kullanımının, olayların hızlı temposunun daha fazla işlem gücü gerektirdiği bir Endüstriyel IoT ortamında önemli ölçüde değişir. Verimli bir işlemci ile yüksek performanslı, özellikli bir işletim sistemi kullanan özel olarak geliştirilmiş bir düğüm, bu gereksinimleri dengeleyebilir, en uygun çözümü sunabilir. Bu proje, RISC-V işlemci mimarisine dayalı bir prototip IoT düğümünün bir Artix-7 FPGA kullanarak donanım gerçekleştirmesini ele alır. Proje, özel bir sınır yönlendiricisi etrafındaki bir yıldız ağına yerleştirilen bir kavram kanıtı uygulamasını desteklemek için gerçekleştirmesi yapılan özel SoC'yle gerekli Zephyr OS sürücülerinin geliştirilmesini sunar. Uçtan uca mesajlar, düğüm, ThingSpeak bulut platformu arasında gönderilip alınabilir. Bu tez, erişilebilir RISC-V işlemci gerçekleştirmelerinin bir analizini, gerekli öğelerin bir açıklamasını, ortam yapılandırması ile proje tasarımı için ayrıntılı bir kılavuzu içerir.