

**FENERBAHÇE ÜNİVERSİTESİ**

**Bilgisayar Mühendisliği**

RISC-V Tabanlı İşlemci Tasarımı

Ahmet Hazar Haspolat

Mustafa Berk Taşkın

Cüneyt Balcı

Ömer Sait Yorulmaz

e-mail: {ahmet.haspolat, mustafa.taskin, cuneyt.balci, omer.yorulmaz}@stu.fbu.edu.tr

İçindekiler

[GİRİŞ 3](#_Toc72954736)

[İçerik 3](#_Toc72954737)

[Bağlam 3](#_Toc72954738)

[SİSTEM MİMARİSİ 4](#_Toc72954739)

[Tanımlar 4](#_Toc72954740)

[ALU TASARIMI 5](#_Toc72954741)

[INSTURCTION DECODER TASARIMI 7](#_Toc72954742)

[KULLANILAN YAZILIM 9](#_Toc72954743)

[SONUÇLAR 9](#_Toc72954744)

[PROJE EKİBİ 9](#_Toc72954745)

[REFERANS DOSYALAR 10](#_Toc72954746)

[KAYNAKLAR 10](#_Toc72954747)

# GİRİŞ

## İçerik

Proje kapsaminda temel hatları önceden oluşturulmuş olan bir RISC-V işlemcisinin Instruction Decoder ve ALU modüllerini SystemVerilog ile gerçekleştirilip, sistemin doğrulama çalışmaları yapılacaktır.

The Instruction Decoder and ALU modules of a RISC-V processor will be created with SystemVerilog features and the verification studies of the system will be observed.

Keywords: RISC-V, RTL, CPU, FPGA, SystemVerilog

## Bağlam

*Bu dokümanın kapsamında, önceden tanımlanmış olan RISC-V işlemcisinin çalışma mekanizmalarını detaylıca incelemek ve sistem üzerinde tamamlanması gereken kısımları; Instruction Decoder ile ALU bloklarını oluştururarak, sistemin doğruluğunun test edilmesi ve analiz edilme süreci anlatılacaktır.*

*Süreç akışında temel olarak;*

* *RISC-V işlemcisinin detaylarının incelenmesi ve tamamlanmamış kısımların çıkartılması*
* *Tespit edilen kısımların akış şemalarının oluşturulması ve analizi*
* *Bu analizler çerçevesinde işlemci üzerinde yapılacak gerçeklemelerin oluşturulması*
* *Son aşamada gerçeklenen sistemin doğrulunun test edilmesi üzerine çalışmaların yapılması*

*adımları açıklanmıştır.*

*Bahsi geçen yazılımsal sürecin gerçekleştirilmesi için tasarlanmış sistem mimarisi, kullanılan geliştirme araçları ve geliştirilen proje kapsamında elde edilen sonuçlar da doküman içerisinde detaylandırılmıştır.*

# SİSTEM MİMARİSİ

Sistem ana mimarisininde toplamde üç adet modül bulunmaktadır. Bunlar;

* Single Cycle Data Path
* Single Cycle Control Path
* Data Memory Interface

Single Cyle Data Path içerisinde ALU ve Instruction Decoder bloklarıyla birlikte multiplexerlar, program counter, regFile gibi bloklar bulunmaktadir. Sistemin çalışabilmesi ve tam performanslı hale gelebilmesi adına bu modül içerisinde olan ALU ve Instruction Decoder blokları çalışır hale getirilecektir. Aşağıdaki görsel ile birlikte system mimarisinin detayları görülebilir.

## Tanımlar

**RISC-V:** RISC(Reduced Instruction Set Computer) prensiplerini kullanan açık kaynak bir Komut Seti Mimarisidir(ISA). İşlemcilerini CISC mantığı ile yapan firmalara örnek olarak AMD ve Intel'i verebiliriz. İşlemcilerini RISC mantığı ile yapan firmalara örnek ise ARM verilebilir.

**ALU(Arithmetic Logic Unit):** Birimi aritmetik ve mantık işlemlerini gerçekleştiren bir dijital devredir. AMB en basit işlemi gerçekleştiren mikro denetleyiciden, en karmaşık mikroişlemciye sahip bir bilgisayara kadar tüm işlemcilerin yapıtaşıdır.

**Instruction Decoder:** İşlemcinin yapması gereken kodların icrası için gerekli işlemleri başlatır ve komutun çalıştırılması için gerekli işlemleri belirler.

## ALU TASARIMI

Aşağıda ALU ünitesinin giriş ve çıkış sinyalleri gösterilmektedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

İşlemcinin ALU’sunun destekleyeceği 11 adet işlem vardır. Bu işlemlerden hangisinin yapılacağı alu\_function girişinden gelmektedir. İşlemlere göre a ve b sayıları, result isminde sonuç çıkışı ve sonuç eğer sıfır ise, ayrı bir çıkış olarak sonucun sıfır olması durumunda 1 olan bir çıktı vardır.

Aşağıdaki tabloda ALU’nun desteklediği işlemler ve operasyon kodları verilmektedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Operasyonların açıklamaları aşağıda listelenmektedir.

• ADD: A + B

• SUB: A - B

• SLL: A << B

• SLR: A >> B

• SRA: A >>> B

• SEQ: A == B

• SLT: A < B

• SLTU: $unsigned(A) < $unsigned(B)

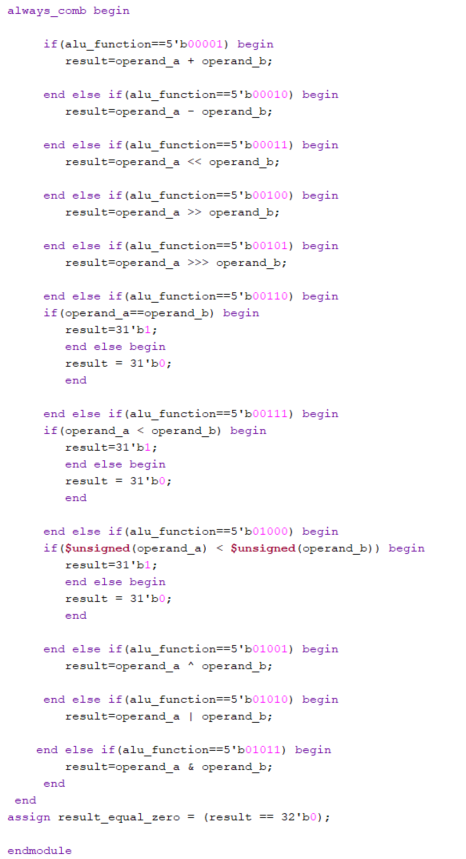
• XOR: A ^ B

• OR: A | B

• AND: A & B

Alu.sv dosyasi içerisinde *gerçekleşmiş* olduğumuz ALU komut setleriyle birlikte,

* alu\_function değeri okunur. Bu değer operand a ve operand b değerlerini hangi mantık işlemi ya da aritmetik işlem içerisine dahil edeceğimizi gösterir.
* Bu değer belirlendikten sonra işleme tabii tutulacak operandlar ilgili işlem içerisinde konumlandırılır.
* İşlem sonucunda ortaya çıkan değer üzerine result değeri belirlenir
* Result değeri belirlenirken işlem sonucu baz alınır. Gerçeklenen işlem sonucu sıfır ise result değeri bir, tam tersi durumunda yani işlem sonucu sıfırdan farklı ise result değeri sıfır olarak atanır.



## INSTURCTION DECODER TASARIMI

Aşağıda Instruction Decoder ünitesinin giriş ve çıkış sinyalleri gösterilmektedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu modülde giriş olarak 32 bitlik instruction word’u alınmaktadır. Çıkışte ise instruction’un parse edilmiş hali, yani decode edilmiş hali çıkış olarak verilmektedir.

• Opcode, instruction’un ilk 7 bitini yani [6:0]’ı temsil etmekte

• Func3, instruction’un 14-12 bitleri arasını [14:12];

• Func7, instruction’un 31-25 bitleri arasını [31:25];

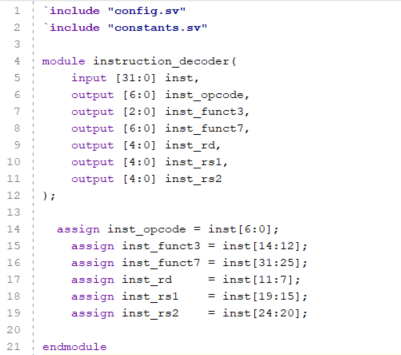
• Rd, instruction’un 11-7 bitleri arasını [11:7];

• RS1, instruction’un 19-15 bitleri arasını [19:15];

• RS2, instruction’un 24-20 bitleri arasını [24:20];

Instruction\_decoder.sv dosyasi içerisinde *gerçekleşmiş* olduğumuz Instruction Decoder’ın komut setleriyle birlikte,

* inst inputundan gelen 32 bitlik komut seti ilgili formatlara ayrılır. Aşağıdaki görselde 32 bitlik Instruction Formats detayları görülmektedir.



tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

* İlgili formatların ayrım detayları;
  + Inst\_opcode = inst[6:0]
  + Inst\_funct3 = inst[14:12]
  + Inst\_funct7 = inst[31:25]
  + Inst\_rd = inst[11:7]
  + Inst\_rs1 = inst[19:15]
  + Inst\_ rs2 = inst[2420]

Tanımlar;

* Opcode **(7 bits)**: 6 tip talimat formatını belirtir.
* Funct3 + Funct7 **(10 bits):** Bu iki alan, gerçekleştirilecek işlemi belirtir.
* Rs1 **(5 bits)**: İşleme girecek ilk operandı (operand\_a) belirtir.
* Rs2 **(5 bits)**: İşleme girecek ikinci operandı (operand\_b) belirtir.
* Rd **(5 bits)**: Hesaplama sonucunun yönlendirileceği hedefi belirtir.

# KULLANILAN YAZILIM

Projenin işleyişi aşamasında gerçeklenen ALU ve Instruction Decoder blokları; işlemci içerisinde yapılması planan aritmetik ve mantık işlemlerinin hesaplanabilmesi ve bu işlemlerin anlamlandırılabilmesi için emirlerin formatlanmasıyla birlikte sistemin kullanılabilmesi için Xilinx tarafından geliştirilen Vivado Design Suite yazılımı kullanılmıştır. Vivado Design Suite, HDL tasarımlarının sentezi ve analizi için üretilmiş bir yazılım paketidir ve Xilinx ISE'nin yerine çip geliştirme ve üst düzey sentez sistemi için ek özellikler sunar. Biz de SystemVerilog dilini kullanarak Vivado üzerinde tasarımımızı yaptık. IEEE 1800 olarak standartlaştırılmış SystemVerilog ise elektronik sistemleri modellemek, tasarlamak, simüle etmek, test etmek ve uygulamak için kullanılan bir donanım açıklaması ve donanım doğrulama dilidir.

# SONUÇLAR

Bu projenin tasarımı ve geliştirilmesi aşamasında öncelikle Vivado Design Suit ile çalışma gereksinimi keşfedildi ve sonrasında bu gereksinim üzerinden araştırma, deneme-yanılma, uygulama konusunda ilerleme kaydedildi. İkincil olarak projenin temel konusu olan ‘RISC-V Tabanlı İşlemci Tasarımı ’nın yazılımsal ve donanımsal temelleri, işlemci içerisinde yer alan blokların çalışma presipleri ve aralarındaki iletişimlerin doğru bir şekilde kurulmaları hususunda kazanımlar sağlandı. Bu bilgiler ışığında son aşamada yapılmış olan testlerinde başarılı bir şekilde sonuçlanmasıyla birlikte, işlemci/donanım alanlarına dair vizyon kazanımı gerçekleştirildi.

# PROJE EKİBİ

AHMET HAZAR HASPOLAT – Çamlıca Doğa Koleji’nde lise eğitimimi tamamlamış olup, lisans eğitimime Fenerbahçe Üniversitesi - Bilgisayar Mühendisliği bölümünde devam etmekteyim. Akademik eğitimime görüntü işleme ve yapay zeka alanlarında devam etmeyi hedefliyorum.

MUSTAFA BERK TAŞKIN – Bursa Doğa Koleji Anadolu Lisesi mezuniyetimin ardından, lisans eğitimime Fenerbahçe Üniversitesi - Bilgisayar Mühendisliği bölümünde devam etmekteyim. Python ve Javascipt dillerinde kendimi geliştirmeye devam ediyorum.

CÜNEYT BALCI – 28.08.2000 yılında İstanbul'da doğdum. 2018 yılında Final Temel Lisesi'nden mezun oldum. Şu anda Fenerbahçe Üniversitesi - Bilgisayar Mühendisliği bölümünde ve çift ana dal programı kapsamında Ekonomi (İngilicze) bölümünde lisans eğitimi almaktayım.

ÖMER SAİT YORULMAZ – Fenerbahçe Üniversitesi - Bilgisayar Mühendisliği öğrencisiyim. Akademik eğitimimle eşzamanlı olarak, 2018 yılından beri özel bir şirkette, Software Developer pozisyonunda görev almaktayım.

# REFERANS DOSYALAR

<https://www.youtube.com/watch?v=6Ug5TEkdGaw>

<https://github.com/cuneytbalci/riscv_design_project>

# KAYNAKLAR

* http://www.levent.tc/courses/computer-architecture
* Thomas, Donald, Moorby, Phillip "The Verilog Hardware Description Language" Kluwer Academic Publishers, Norwell, MA. [ISBN 0-7923-8166-1](https://tr.wikipedia.org/wiki/%C3%96zel:KitapKaynaklar%C4%B1/0792381661)
* Janick Bergerdon, "Writing Testbenches: Functional Verification of HDL Models", 2000, [ISBN 0-7923-7766-4](https://tr.wikipedia.org/wiki/%C3%96zel:KitapKaynaklar%C4%B1/0792377664). (The HDL Testbench Bible)