**FENERBAHÇE ÜNİVERSİTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ**



**COMP102-BİLGİSAYAR MİMARİSİ**

**RISC-V Tabanlı İşlemci Tasarımı-PROJE ÖDEVİ**

**BARIŞ SUBAŞİ - 200301002**

**TENAY DİLARA ÖZDEMİR - 200301017**

**TUĞBA KARADENİZ - 200301025**

**AYBÜKE ŞEN - 200301034**

**ECE AYDINKAPTAN - 190301018**

İçindekiler

[**İÇERİK** 3](#_Toc103727585)

[**KONTEKST** 3](#_Toc103727586)

[**SİSTEM MİMARİSİ** 4](#_Toc103727587)

[**TANIMLAR** 4](#_Toc103727588)

[**RISC-V:** 4](#_Toc103727589)

[**ALU(Arithmetic Logic Unit):** 5](#_Toc103727590)

[**Instruction Decoder:** 5](#_Toc103727591)

[**ALU TASARIMI** 5](#_Toc103727592)

[**INSTURCTION DECODER TASARIMI** 8](#_Toc103727593)

[**KULLANILAN YAZILIM** 10](#_Toc103727594)

[**SONUÇLAR** 10](#_Toc103727595)

[**REFERANS DOSYALAR** 11](#_Toc103727596)

[**KAYNAKLAR** 11](#_Toc103727597)

# **İÇERİK**

Proje kapsamında temel hatları önceden oluşturulmuş olan bir RISC-V işlemcisinin Instruction Decoder ve ALU modüllerini SystemVerilog dili özellikleri kullanılarak işlemci üzerinde tasarım ve sistemin doğrulama çalışmaları yapılacaktır.

Projenin amacı RISC-V tabanlı bir işlemcinin tasarımını, çalışma prensibini, komutları alış ve kullanış biçimlerini öğrenmek ve uygulayabilmektir.

Anahtar Kelimeler: RISC-V, RTL, CPU, FPGA, SystemVerilog

# **KONTEKST**

Bu **belge kapsamında daha önce RISCV işlemcinin çalışma mekanizmaları** tanımlanmış ve sistem üzerinde tamamlanması gereken **kısımlar;** Instruction Decoder ile ALU **blokları üreterek sistem doğruluğunu kontrol etme** ve analiz **etme** 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 gerçekleştirilen sistemin doğrulunun test edilmesi adına yapılan çalışmaların adımları açıklanmıştır.

Söz konusu yazılım sürecini hayata geçirmek için tasarlanan sistem mimarisi, kullanılan geliştirme araçları ve geliştirilen proje çerçevesinde elde edilen sonuçlar da dokümanda detaylandırılmıştır.



# **SİSTEM MİMARİSİ**

Sistem ana mimarisininde toplamde üç adet modül bulunmaktadır. Bunlar;

- Single Cycle Data Path

- Single Cycle Control Path

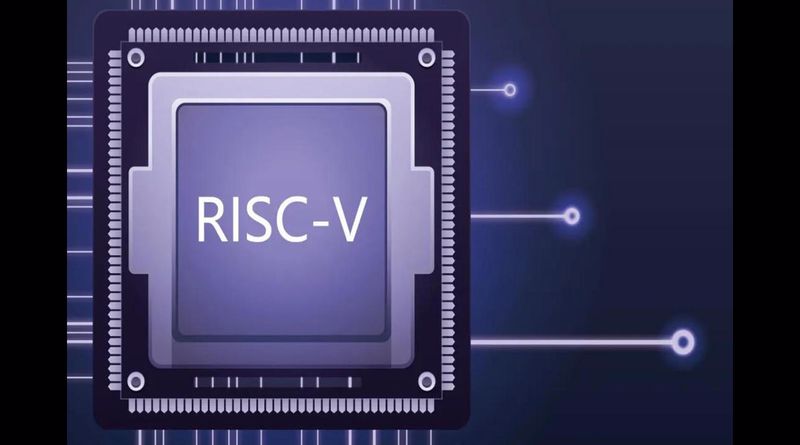
- Data Memory Interface

Single Cyle Data Path içerisinde ALU ve Instruction Decoder bloklarıyla birlikte multiplexerlar, program counter, regFile gibi bloklar bulunmaktadir.

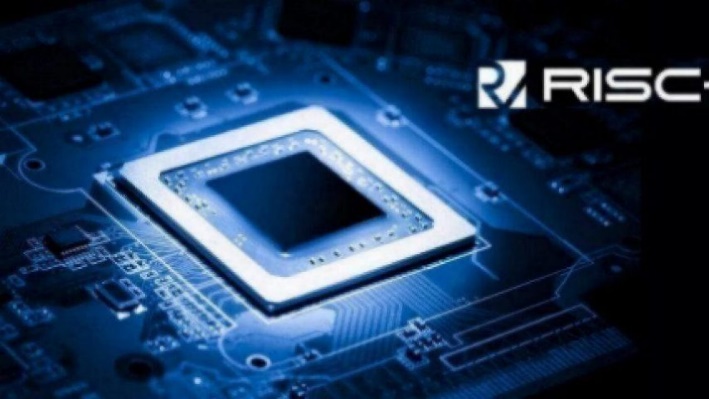
Sistemin çalışabilmesi ve tam performanslı hale gelebilmesi adına bu modül içerisinde olan ALU ve Instruction Decoder blokları çalışır hale getirilecektir.

# **TANIMLAR**

**RISC-V:** Yerleşik indirgenmiş komut seti bilgisayar ilkelerine dayanan açık standart bir komut seti mimarisidir. Diğer ISA tasarımlarının çoğundan farklı olarak, RISC-V, kullanım için ücret gerektirmeyen açık kaynak lisansları altında sağlanır.

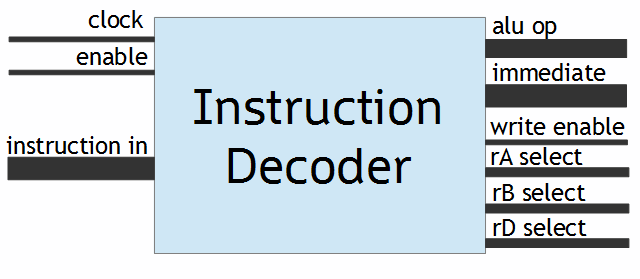


İşlemcilerini CISC mantığı ile yapan firmalara örnek olarak AMD ve Intel'i verebiliriz. İşlemcilerini RISC mantığı ile yapan firmalara örnek ise ARM verilebilir.



**ALU(Arithmetic Logic Unit):** Aritmetik Mantık 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.

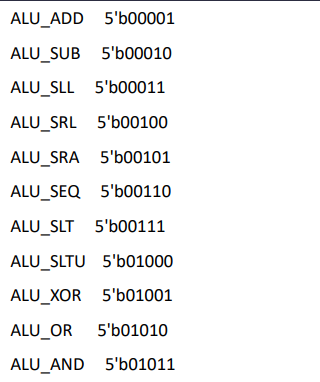
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

İşlemci ALU'nun destekleyeceği 11 işlem vardır. Bu işlemlerden hangisinin yapılması gerektiği, alu\_ Function bölümünden 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.



ALU ’nun desteklediği operasyon kodlarının yapacağı işlemler aşağıda verilmiştir.

• 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 dosyasında gerçekleştirilen ALU komut seti ile,

- alu\_function değeri okunur. Bu okunan değer operand a ve operand b değerlerini hangi mantık işlemi ya da aritmetik işlem içerisine dahil edeceğimizi gösterir.

- Bu değer belirlendikten sonra işleme alınacak operandlar ilgili işlem içerisine dahil edilir.

- İşlemin bitmesiyle ortaya çıkan değer yardımıyla result değeri belirlenir.

- Result değeri belirlenirken işlem sonucu temel alınır. Yapılan işlem sonucu sıfır ise result değeri bir, tam tersi durumunda yani işlem sonucu sıfırdan farklı ise result değeri sıfır olarak atanır.

metin, ekran görüntüsü, belge içeren bir resim

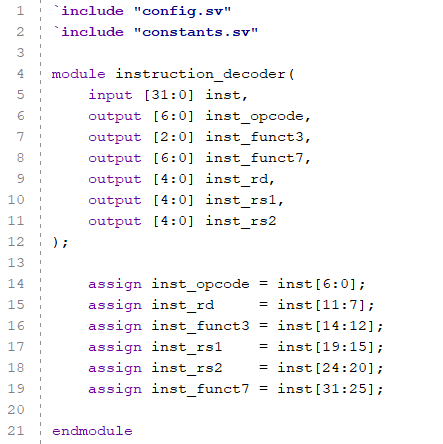
Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

# **INSTURCTION DECODER TASARIMI**

Aşağıda Instruction Decoder ünitesinin giriş ve çıkış sinyalleri gösterilmektedir.



Bu modülde giriş olarak 32 bitlik instruction word’u alınmaktadır. Çıkışta ise instruction’un 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 gerçekleştirdiğimiz 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

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

İlgili formatların 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**

Bu projede Vivado Design Suite kullanılmıştır.

Vivado Design Suite, FPGA geliştirme kartları üzerinde çalışmalar yapmak için gerekli olan tasarımı oluşturmak için kullanılmaktadır. Verilog, VHDL vb..donanım tasarım dillerini alarak, FPGA’e konfigüre edilebilecek (Xilinx firması FPGA’leri için .bit uzantılı dosyalar) tasarım dosyasını oluşturur.

Proje sırasında gerçekleştirdiğimiz ALU ve Instruction Decoder blokları; işlemci içerisinde yapılması planlanan aritmetik ve mantık işlemlerinin hesaplanabilmesi 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**

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 kurulmasını öğrendik.

RISC-V Tabanlı İşlemci Tasarımı ’nın yazılımsal ve donanımsal temelleri, işlemci içerisinde yer alan blokların çalışma prensipleri ve aralarındaki iletişimlerin doğru bir şekilde kurulmalarını öğrendik.

Vivado Design Suite tasarım aracını kullanarak bir işlemcinin nasıl donanım tasarımı yapıldığını öğrendik.

Geliştirilen RISC-V işlemcisinin ALU ve Instruction Decoder tasarımlarını tamamlayarak gerekli koşullar sağlandığında tasarımdaki 11 komutu yerine getirebildiğini öğrendik.

ALU ve instruction\_decoder tasarımını gerçekleştirdik. Bu işlemciyi test kodları ile denedik ve doğru çalışıp çalışmadığını gözlemledik.

# **REFERANS DOSYALAR**

Github linki: [github.com/barisubasi/FBU-RISC-V-DESIGN](https://github.com/barisubasi/FBU-RISC-V-DESIGN)

Youtube Sunum Video Linki: [youtube.com/watch?v=REiDWJjkDaE](https://www.youtube.com/watch?v=REiDWJjkDaE)

# **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
* Janick Bergerdon, "Writing Testbenches: Functional Verification of HDL Models", 2000, ISBN
* 0-7923-7766-4. (The HDL Testbench Bible)
* <https://riscv.org/>
* <https://en.wikipedia.org/wiki/RISC-V>
* <https://lab.deepcontrol.net/risc-v-nedir/>
* <https://www.chip.com.tr/haber/cpu-nedir-nasil-calisir_9198.html#:~:text=Komut%20%C3%87%C3%B6z%C3%BCc%C3%BC(Instruction%20Decoder)%3A,depolamak%20i%C3%A7in%20kullan%C4%B1lan%20haf%C4%B1za%20%C3%A7e%C5%9Fididir>.