

RISC-V Tabanlı İşlemci Tasarımı

Evrim Arda Kalafat, Doğa Turan

Fenerbahçe Üniversitesi

Bilgisayar Mühendisliği

İstanbul, Türkiye

e-mail: [evrim.kalafat@stu.fbu.edu.tr](mailto:evrim.kalafat@stu.fbu.edu.tr), doga.turan@stu.fbu.edu.tr

**Proje özeti:** Bu proje kapsamında başlangıç tasarım verilen bir RISC-V işlemcisinin ALU ve instruction decoder blokları temel SystemVerilog dili özellikleri kullanılarak tasarım ve doğrulama çalışmaları yapılacaktır.

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

**Abstract:** Within the scope of this project, we are going to design ALU and Instruction decoder blocks of a RISC-V processor by using the basic SystemVerilog language features.

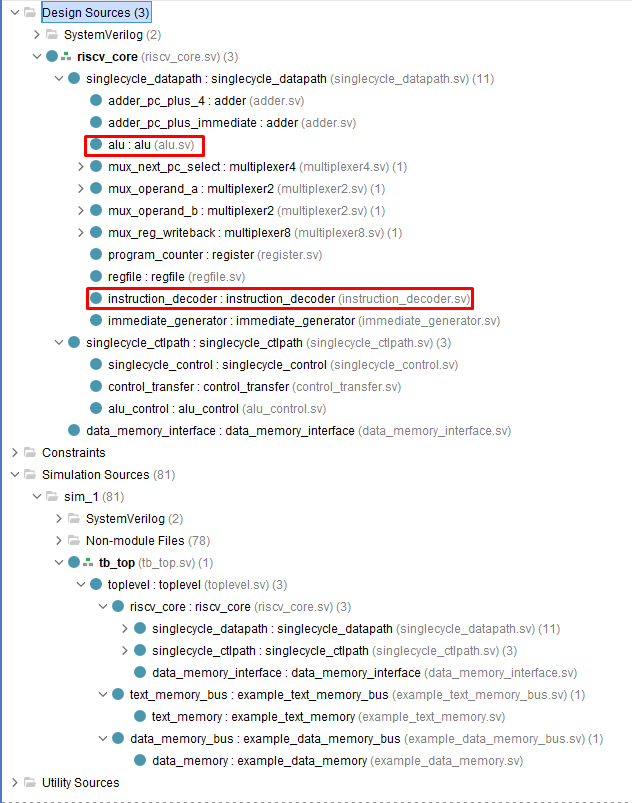
**Keywords:** FPGA, CPU, RISC-V, SystemVerilog, RTL.

1. **Giriş**

Projede bize verilen bir RISC-V işlemcisinin ALU ve Instruction decoder bloklarını SystemVerilog dilini kullanarak tasarlamak, bize verilen test kodları ile tamamlanan işlemcimizin doğruluğunu test etmek, temel SystemVerilog dili özelliklerinde kendimizi geliştirmek, RISC-V işlemcisinin yapısını daha iyi tanımak amaçlanmıştır.

1. **Sistem Mimarisi**

Aşağıdaki şekilde sistemin ana mimarisini görebiliyoruz. Biz bu mimarinin içinde boş bırakılmış “alu.sv” ve “instruction\_decoder.sv” dosyalarını tasarlıyoruz.

****

**RISC-V:** RISC(Reduced Instruction Set Computer) prensiplerini kullanan açık kaynak bir **Komut Seti Mimarisidir(ISA)**. University of California Berkeley'in oluşturduğu ve herkese açık (ister kişisel, ister ticari, ister akademik) bir ISA olan RISC-V, herhangi bir lisans parası ödemeden herkesin ortak kabul ettiği bir mimaride işlemci üretebilmenizi sağlıyor.

**Memory:** Saklama alanı.RISC-V işlemcisinde komutları ve verileri tutan 2 tane bellek bulunur.

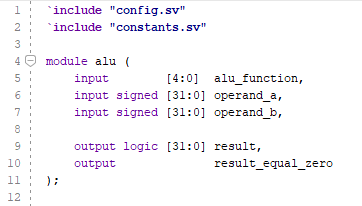
**PC(Program Counter):** Hangi adresteki komutun çalıştığını ifade eder.

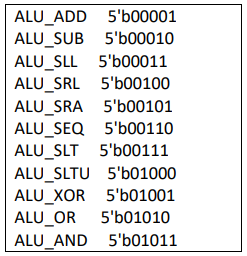
**Register File:** Saklayıcıların bulunduğu bir dizidir. RISC-V işlemcisinde her biri 32 bitlik 32 adet saklayıcı bulunur.

**ALU(Arithmetic Logic Unit):** aritmetik ve mantık işlemlerini gerçekleştiren bir dijital devredir.

**ALU Tasarımı:**

Aşağıdaki şekilde bize verilen kod parçasını görüyoruz. ALU; alu\_function, operand\_a ve operand\_b olmak üzere 3 adet giriş alıyor. Result ve result\_equal\_zero olmak üzere 2 adet çıkış veriyor.

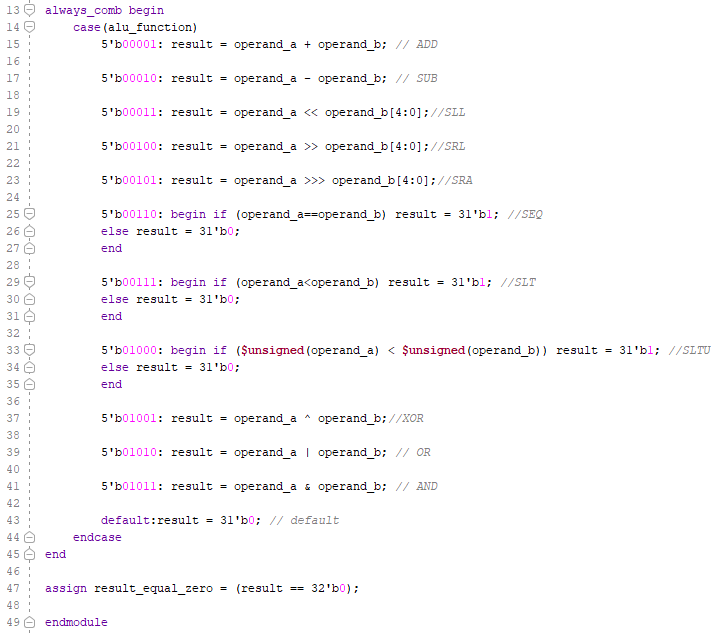


Aşağıda bu alu’nun desteklediği işlemler ve operasyon kodları verilmiştir.

**ALU’nun işlemlerinin detaylı gösterimi;**

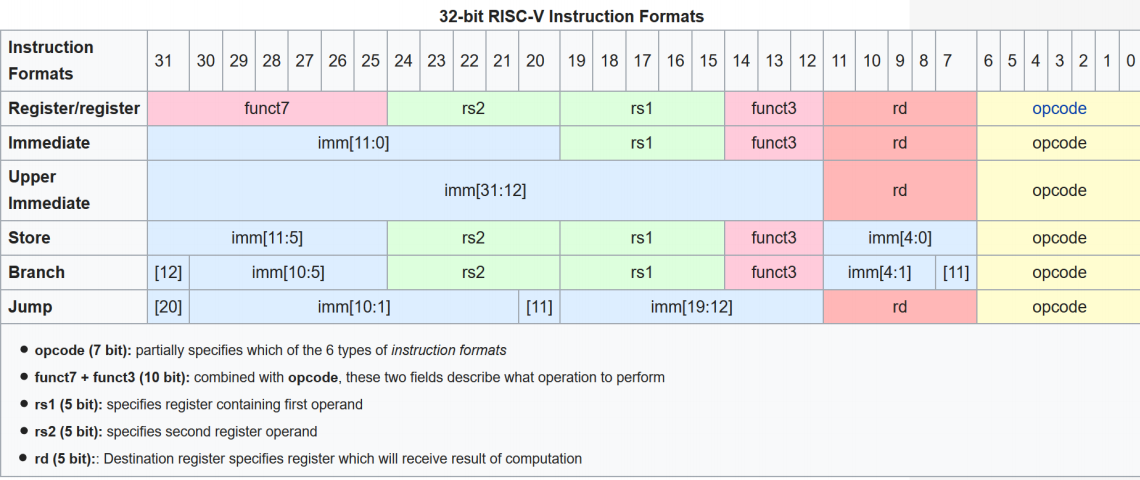
* ADD: A + B
* SUB: A - B
* SLL: A << B
* SLR: A >> B
* SRA: A >>> B
* SEQ: A == B
* SLT: A < B
* SLTU: $unsigned(A) < $unsigned(B)
* XOR: A ^ B
* OR: A | B
* AND: A & B

Aşağıda bizim tasarladığımız kod parçasını görüyoruz. ALU bir kombinasyonel devre olduğu için SystemVerilog dilindeki always\_comb’u kullandık. Case yapısının içinde tüm operasyon kodlarına karşılık gelen işlemleri yaptırdık ve default olarak result’u 0’a eşitledik. Case yapısından sonra result\_equal\_zero değerini result’a göre assign ettik.



**Instruction\_decoder Tasarımı:**

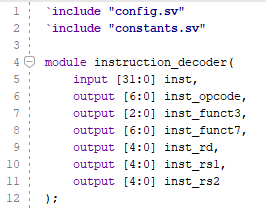
RISC-V Instructionları 32 bit uzunluğundadır. Instruction\_decoder, instruction’dan gelen 32 bitlik veriyi parçalayarak çıktı verir. Aşağıda bu 32 bitlik Instruction formatlarını görüyoruz.



Aşağıda şekil[1]’de bize verilen kod parçasını, şekil[2]’de ise bizim tasarladığımız kısmı görüyoruz.

Şekil[1]’de instruction\_decoder, inst isminde 32 bitlik bir girdi almakta ve bu girdiyi, inst\_opcode, inst\_funct3, inst\_funct7, inst\_rd, inst\_rs1 ve inst\_rs2 olmak üzere 6 parçaya bölmekte.

Şekil[2]’de ise inst girdisinden gelen 32 biti 6 parçaya ayırıyoruz. Bunun için SystemVerilog dilindeki assign’ı kullandık.

 Text

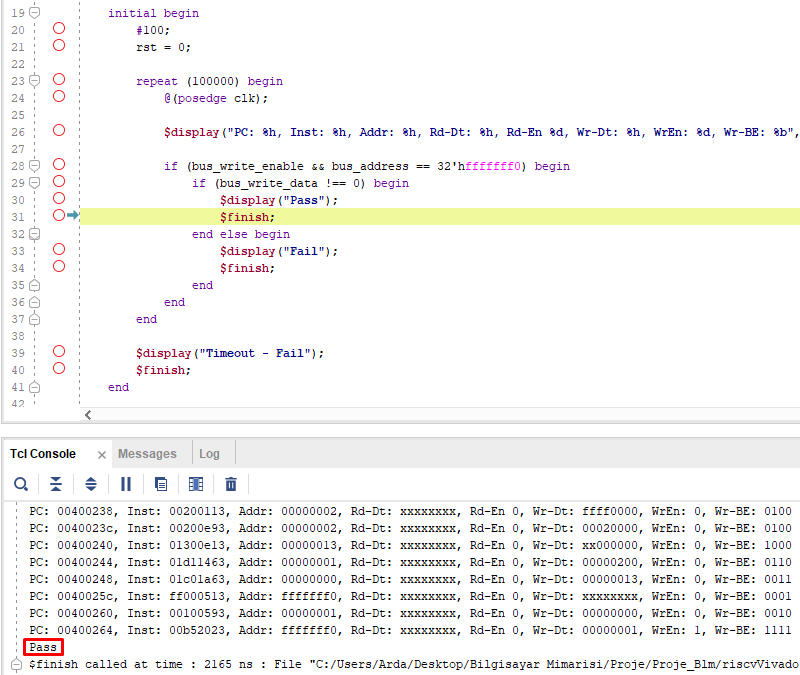
Description automatically generated

Şekil[1] Şekil[2]

1. **Kullanılan Yazılım**

Tasarımlarımızı yapmak için Xilinx tarafından geliştirilen Vivado Design Suite yazılımını kullandık. 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.

Alu ve Instruction\_decoder tasarımlarımızı bitirdikten sonra Simulation sekmesinin altından Run Simulation > Run Behavioral’a tıklayarak simülasyonumuzu çalıştırdık. Sonra yukarıdaki Run All(F3) tuşuna basarak testlerimizi gerçekleştirdik.



Yukarıda da gözüktüğü gibi “Pass” çıktısını aldık. Tasarımımız testleri başarıyla geçti.

1. **Sonuçlar**

Geliştirilen işlemci ADD, SUB, SLL, SLR, SRA, SEQ, SLT, SLTU, XOR, OR ve AND işlemlerini yapabilmekte. Bu projeyle birlikte basit bir şekilde bir RISC-V işlemcinin nasıl çalıştığını ve mimarisini öğrenmiş olduk. SystemVerilog dilinde kendimizi geliştirdik. Bir işlemcide Alu ve Instruction\_decorder tasarımını gerçekleştirdik. Bu işlemciyi test kodları ile sınadık ve doğru çalışıp çalışmadığını gözlemledik. İşlemcimiz testleri geçti ve başarıyla çalıştı.

1. **Proje Ekibi**

**Evrim Arda KALAFAT**, 25.09.2001 yılında istanbulda doğdu. 2019 yılında Kadıköy Final Temel Lisesi’nden mezun oldu. Şu anda Fenerbahçe Üniversitesi Bilgisayar Mühendisliği bölümünde lisans eğitimi almakta. Bilgisayar Mühendisliği Bölüm Temsilciği yapmaktadır. C, C++ ve Pyhton dillerinde bilgili. Programlama, yapay zeka ve siber güvenlik ile ilgileniyor.

**Doğa TURAN**, 18.09.2001 yılında Gölcük’te dünyaya geldi. 2019 yılında Şehit Özcan Kan Fen Lisesi’nden mezun oldu. Fenerbahçe Üniversitesi’nde Endüstri Mühendisliği anadal ve Bilgisayar Mühendisliği ikinci ana dal olmak üzere lisans eğitimi almaktadır. C ve Pyhton dilleri hakkında donanımlıdır.

1. **Referans Dosyalar**

<https://youtu.be/S0RCmj7MAG0>

<https://github.com/rhgod/RISC-V_Project>

1. **Kaynaklar**

[1] Asanović, K., & Patterson, D. A. (2014). Instruction sets should be free: The case for risc-v. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146.

[2] Waterman, A. S. (2016). Design of the RISC-V instruction set architecture (Doctoral dissertation, UC Berkeley).

[3] Traber, A., Zaruba, F., Stucki, S., Pullini, A., Haugou, G., Flamand, E., ... & Benini, L. (2016, January). PULPino: A small single-core RISC-V SoC. In 3rd RISCV Workshop.

[4] Sutherland, S., Davidmann, S., & Flake, P. (2006). SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling. Springer Science & Business Media.