

**FB-CPU RTL TASARIMI**

Behçet Muhammed, Adem Çolak, Hazem Kataie, Danial Erfani Akhlaghi, Caner Gül

Fenerbahçe Üniversitesi

Bilgisayar Mühendisliği

İstanbul, Türkiye

e-mail: {behcet.muhammed, adem.colak, hazem.kataie, [danial.erfani, caner.gul}@stu.fbu.edu.tr](mailto:danial.erfani,%20caner.gul%7d@stu.fbu.edu.tr)

Özetçe: 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 - GİRİŞ:** 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.

**2- SİSTEM MİMARİSİ:**

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

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**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.

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

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

**tablo içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

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.

**tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**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.

**tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu**

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.

**tablo içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Şekil 1 Şekil 2**

**Xilinx Vivado Design Suite**

Xilinx 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. Vivado Tasarım Aracı, Xilinx’in 7 ve daha yeni jenerasyon FPGA’leri için kullanılabilen bir geliştirme ortamıdır. Bu ortam Xilinx’in sunduğu çeşitli geliştirme ve doğrulama araçlarını barındırır.

Vivado:

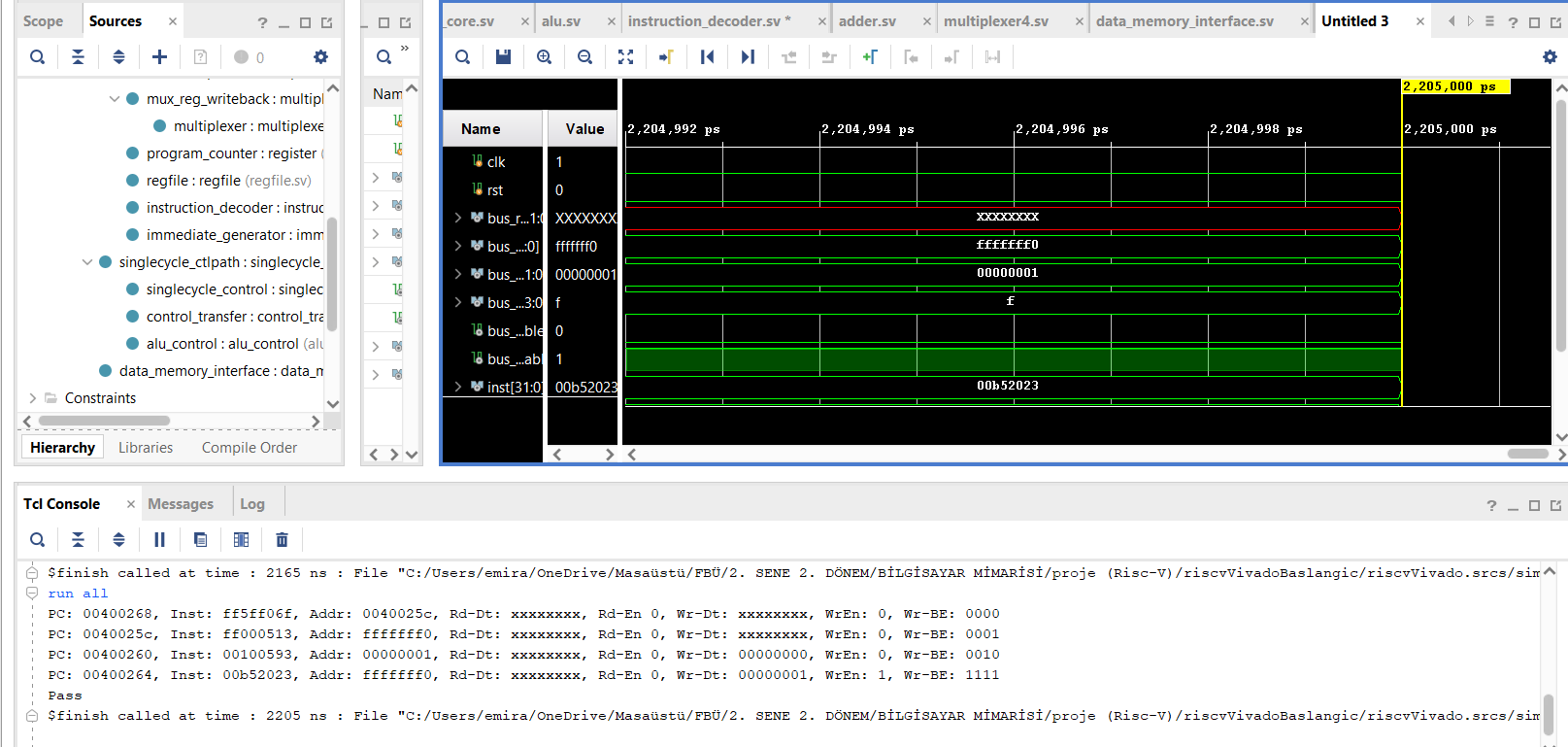
* + Verilog
  + **System Verilog**
* VHDL Dillerini desteklemektedir. Projede Verilog dili ile tasarımlar yapılacaktır.

**3- KULLANILAN YAZILIM:**

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.

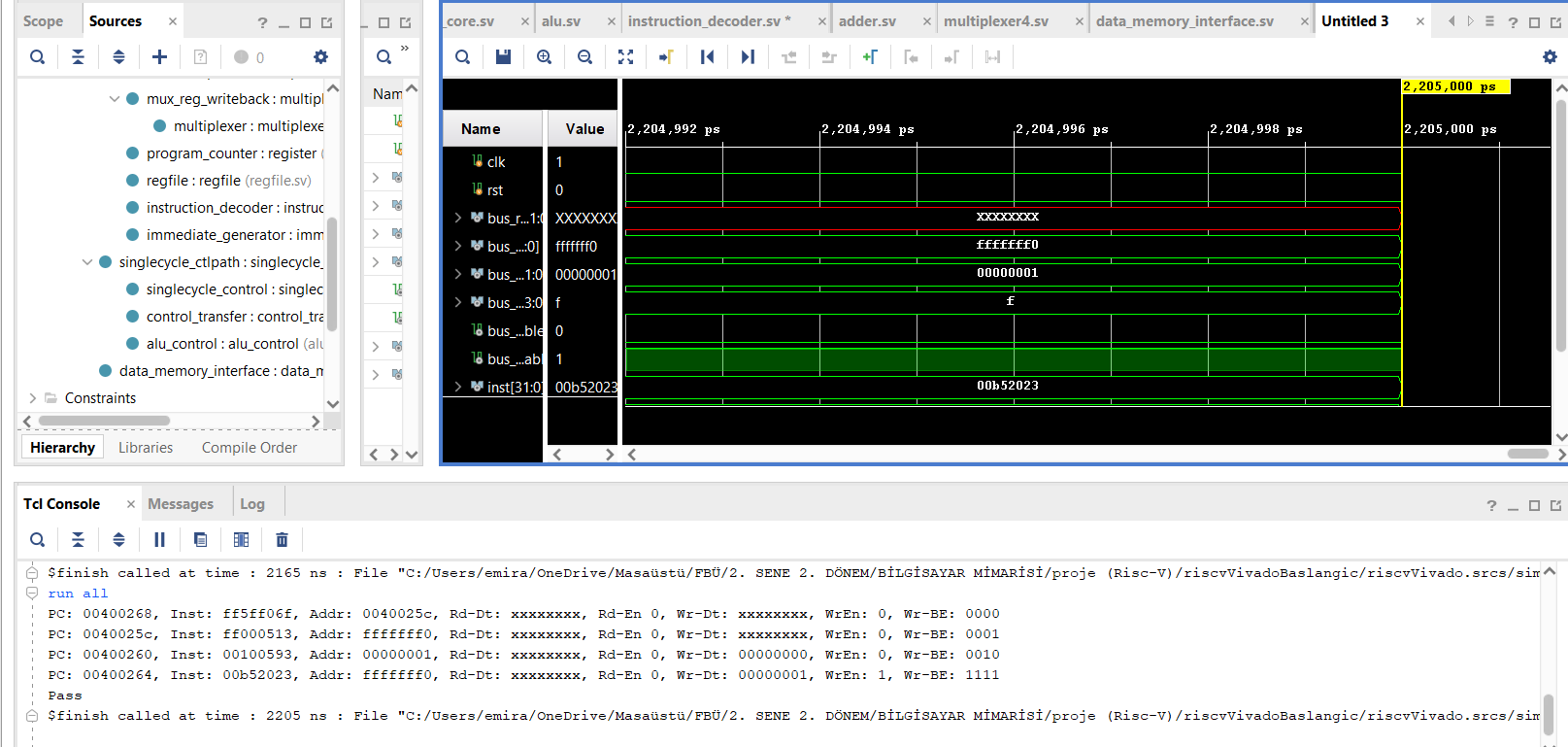
**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

****

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

**ÇIKTILAR**



**Tasarımımızın testbench modülleriyle test edilmesi sonucu PASS sonucunu vermesi doğru çalıştığını göstermektedir.**

**4- SONUÇLAR:**

Geliştirilen RISC-V CPU işlemcisi belirlenen koşulları sağladığında 11 adet komut (ADD, SUB, SLL, SLR, SRA, SEQ, SLT, SLTU, XOR, OR ve AND) işlemlerini yapabilmektedir. Bu projeyle birlikte basit bir şekilde bir RISC-V işlemcinin nasıl çalıştığını ve mimarisini öğrenmiş olduk. Yapılan işlemcinin RİSC-V temelli olmasının avantajları gördük. İnstructor ve alu ünitelerinin farklı bölümlere ayrılması projenin yönetilmesinin daha kolay olmasını ve hızlı çalışmasını sağlamıştır. işlemciyi yapabilmek için saklayıcılar, bellek, işlem ünitesi ve kontrol ünitesi hakkında gerekli bilgileri edindik. SystemVerilog dilinde kendimizi geliştirdik. Sonuç olarak bu işlemciyi test kodları ile sınadık ve doğru çalışıp çalışmadığını gözlemledik. İşlemcimiz testleri geçti ve başarıyla çalıştı.

**5-PROJE EKİBİ:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AD VE SOYAD** | **OKUL NUMARASI** | **DOĞUM YERİ** | **DOĞUM TARİHİ** | **MEZUN OLDUĞU LİSE** | **İNGİLİZCE SEVİYESİ** |
| **Behçet Muhammed**  **(Proje Ekip Sorumlusu)** | **200301109** | **Halep, Suriye** | **01/01/1997** | **Münif Paşa Çağdaş Lisesi** | **A1** |
| **Adem Çolak** | **200301037** | **Kadıköy, İstanbul** | **17/08/2002** | **Gökyüzü Koleji Anadolu Lisesi** | **A2** |
| **Hazem Kataie** | **200301115** | **Halep, Suriye** | **27/07/2001** | **Akşamsettin AİHL** | **A2** |
| **Danial Erfani** | **200301105** | **Meshhad, İran** | **02/01/2001** | **Kemal Hasoğlu Anadolu lisesi** | **C1** |
| **Caner Gül** | **200301023** | **Zonguldak, Türkiye** | **05/12/2001** | **-** | **A1** |

**6- REFERANS DOSYALAR:**

* **YOUTUBE:** https://www.youtube.com/watch?v=Y9ls41kow9k
* **GİTHUB:** <https://github.com/lilferdinand/RISC-V_PROJESI_-_BLM202>

**7- KAYNAKLAR:**

* Levent, Vecdi Emre (2019) “Von Neumann Mimarisi”, Bilgisayar Mühedisliğine Giriş-Ders Notları.
* Levent, Vecdi Emre (2021) “Sonlu Durum Makinaları”, Bilgisayar Mimarisi - Ders Notları.
* Levent, Vecdi Emre (2021) “RISC-V İşlemcisi”, Bilgisayar Mimarisi - Ders Notları.
* Levent, Vecdi Emre (2021) “Bellekler”, Bilgisayar Mimarisi - Ders Notları.
* Levent, Vecdi Emre (2021) “Pipelining ve Performans”, Bilgisayar Mimarisi - Ders Notları.
* Levent, Vecdi Emre (2021) “RISC-V İşlemci Tasarımı”, Bilgisayar Mimarisi - Ders Notları.
* Levent, Vecdi Emre (2021) “RISC, CISC ISA”, Bilgisayar Mimarisi - Ders Notları.
* 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.
* Waterman, A. S. (2016). Design of the RISC-V instruction set architecture (Doctoral dissertation, UC Berkeley).
* 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.
* 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.