RISC-V Tabanlı İşlemcTasarımı

Hayat Zehra Demir,Ahmet Batuhan Yılmaz, Mert Meriç Karadeniz,Rauf Füzün

Fenerbahçe Üniversitesi Bilgisayar Mühendisliği

İstanbul, Türkiye

E-mail: hayat.demir@stu.fbu.edu.tr, ahmet.yilmaz@stu.fbu.edu.tr, mert.karadeniz@stu.fbu.edu.tr, mehmet.füzün@stu.fbu.edu.tr

Proje Özeti:

Bu projede başlangıç tasarım verilen bir RISC-V işlemcisinin ALU ve İnstruction Decoder blokları temel SystemVerilog dili özellikleri kullanılarak tasarım ve doğrulama çalışmaları yapıldı.

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

Abstract:

In this project, the design and verification studies were carried out using the basic SystemVerilog language features of the ALU and Instruction Decoder blocks of a RISC-V processor, which was given an initial design..

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

GİRİŞ

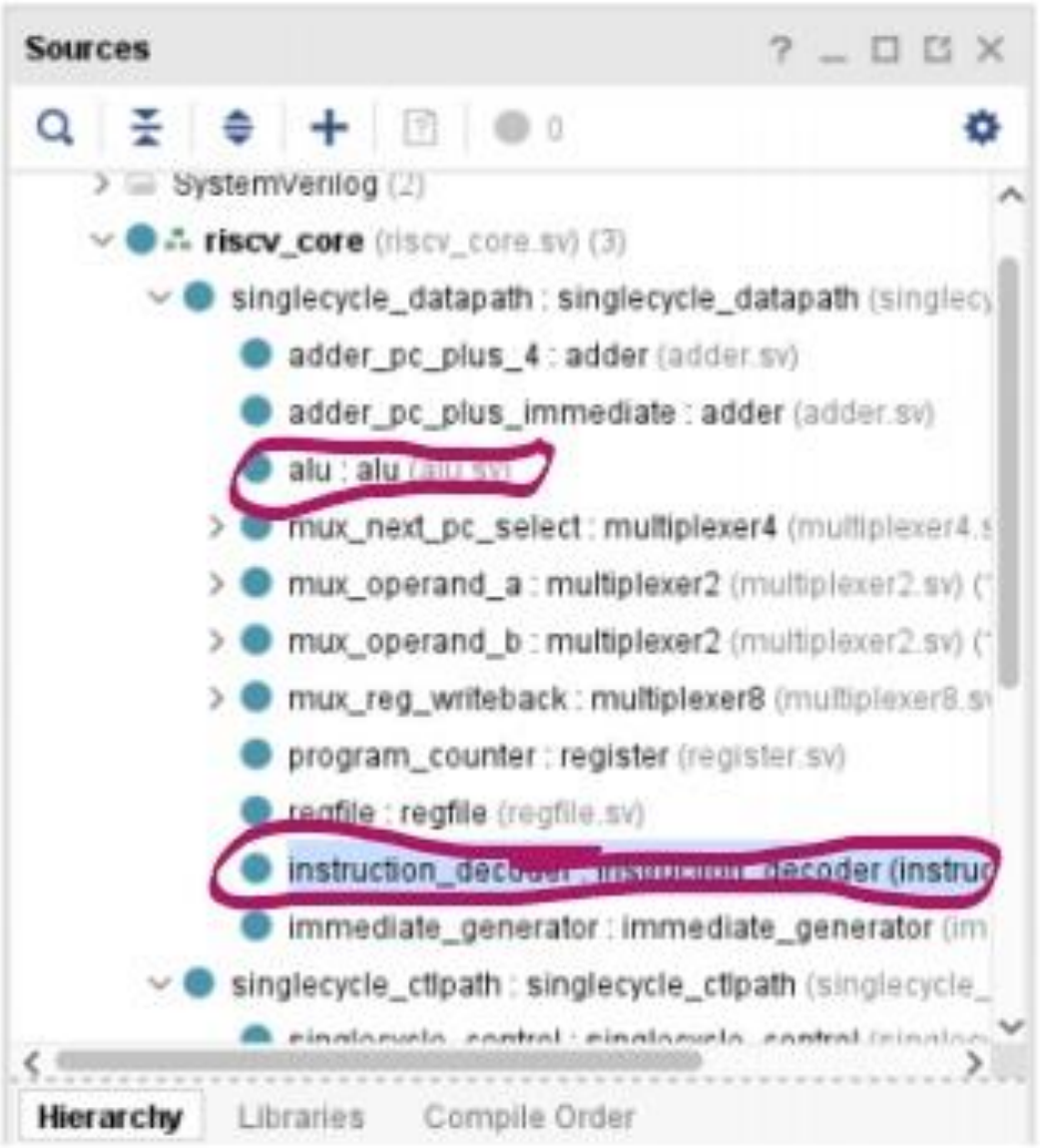
Projede bize bir RISC-V işlemcinin ALU ve Instruction decoder'ı verildi ve blokları SystemVerilog dili ve sağlanan test programları kullanarak tasarladık.

Tamamlanmış işlemcimizin doğruluğunu test etmek için basit SystemVerilog dili kullanıldı.

RISC-V işlemcisinin yapısı, özelliklerimizi iyileştirmek için geliştirildi.

SİSTEM MİMARİSİ

İçi tasarlanmamış “alu.sv” ve “instruction_decoder.sv” dosyalarını tamamlıyoruz.



RISC-V:

RISC-V, yerleşik azaltılmış komut seti bilgisayarı (RISC) ilkelerine dayanan bir açık standart komut seti mimarisidir (ISA).Diğer ISA tasarımlarının çoğunun aksine, RISC-V ISA, kullanım için ücret gerektirmeyen açık kaynak lisansları altında sağlanır.Bir dizi şirket RISC-V donanımı sunuyor veya duyurdu, açık kaynaklı işletim sistemleri mevcuttur ve komut seti birkaç popüler yazılım araç zincirinde desteklenmektedir.

RISC-V'nin Farkı Nedir?

ARM ve X86 ile karşılaştırıldığında, RISC-V aşağıdaki avantajlara sahiptir:

Ücretsiz: RISC-V açık kaynaklıdır, IP için ödeme yapmaya gerek yoktur.

Basit: RISC-V, diğer ticari ISA'lardan çok daha küçüktür.

Modüler: RISC-V, birden çok standart uzantıya sahip küçük bir standart temel ISA'ya sahiptir.

Kararlı: Temel ve ilk standart uzantılar zaten dondurulmuş. Büyük güncellemeler için endişelenmenize gerek yok.

Genişletilebilirlik: Uzantılara göre belirli işlevler eklenebilir. Vector gibi geliştirilmekte olan daha birçok uzantı var.

Memory:

Bellek bilgisayarı oluşturan 3 ana bileşenden biridir. İşlemcinin çalıştırdığı [[Bilgisayar programı programlar ve programa ait bilgiler bellek üzerinde saklanır. Bellek geçici bir depolama alanıdır. Bellek üzerindeki bilgiler güç kesildiği anda kaybolurlar.RISC-V işlemcisinde komutları ve verileri tutan 2 tane bellek bulunur.

Program Counter:

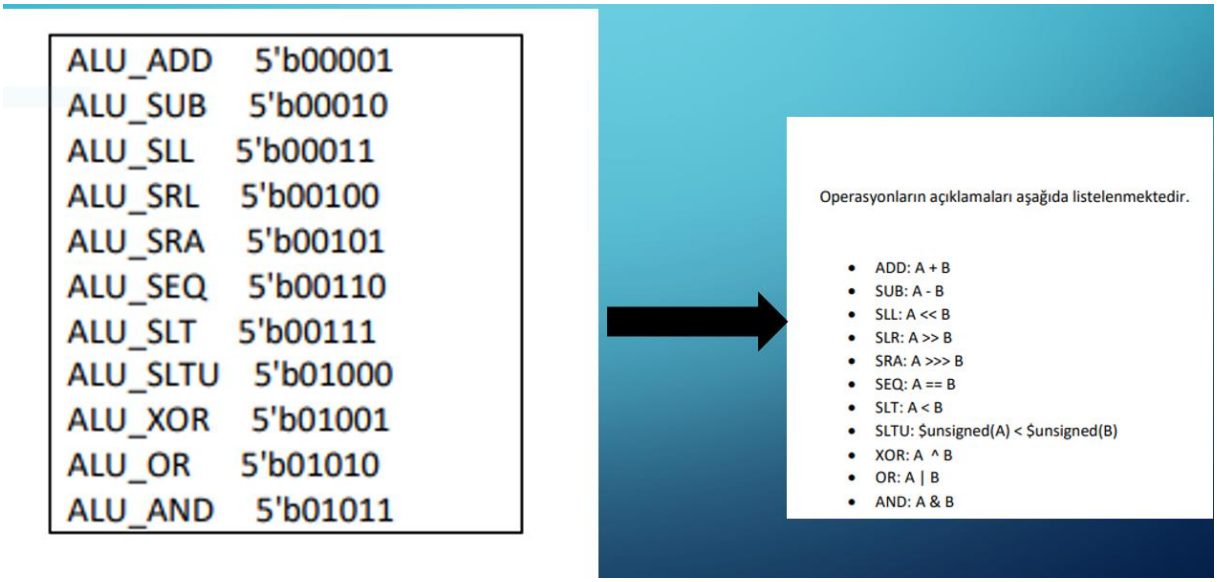
Program sayacı, mevcut 32 bitlik talimat adresini korur ve bunu “pc” veriyolunda çıkarır. Program sayacı, saat sinyalinin "clk" yükselen kenarında güncellenen senkronize bir birimdir.

Register File:

Kayıt Dosyası, CPU içinde bulunan bir bellek alanıdır. İkincil bellek cihazlarından verileri almak ve tutmak için CPU tarafından kullanılır. İşlemci içinde bulunduğundan diğer bellek cihazlarına göre daha hızlıdır.

ALU(Arithmetic Logic Unit):

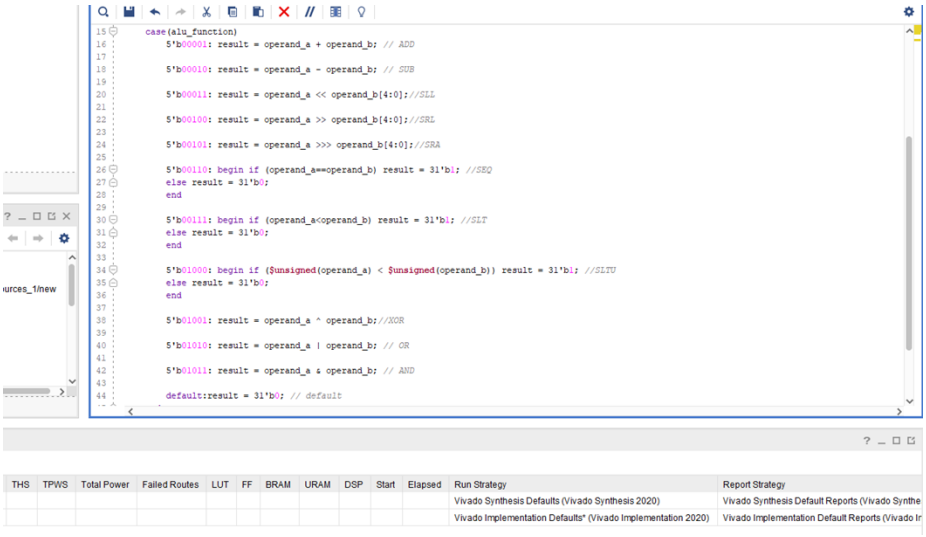
ALU, VEYA ve VE gibi basit toplama, çıkarma, çarpma, bölme ve mantık işlemlerini gerçekleştirir. Bellek, programın talimatlarını ve verilerini saklar.



ALU Tasarımı:

- 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 0 ise, ayrı bir çıkış olarak sonucun 0 olması durumunda 1 olan bir çıktı vardır.

```
module alu (

    input signed [31:0] operand_a,
    input signed [31:0] operand_b,
    input        [4:0]  alu_function,

    output logic [31:0] result,
    output              result_equal_zero
);

always_comb begin
    case(alu_function)
        5'b00001: result = operand_a + operand_b; // ADD

        5'b00010: result = operand_a - operand_b; // SUB
```

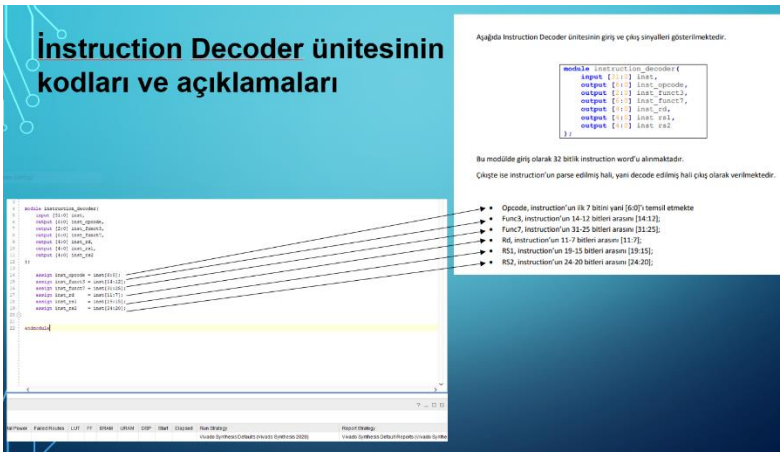


```
case(alu_function)
    5'b00001: result = operand_a + operand_b; // ADD

    5'b00010: result = operand_a - operand_b; // SUB

    5'b00011: result = operand_a << operand_b[4:0];//SLL

    5'b00100: result = operand_a >> operand_b[4:0];//SRL

    5'b00101: result = operand_a >>> operand_b[4:0];//SRA

    5'b00110: begin if (operand_a==operand_b) result = 31'b1; //SEQ
    else result = 31'b0;
    end

    5'b00111: begin if (operand_a<operand_b) result = 31'b1; //SLT
    else result = 31'b0;
    end

    5'b01000: begin if ($unsigned(operand_a) < $unsigned(operand_b)) result = 31'b1; //SLTU
    else result = 31'b0;
    end

    5'b01001: result = operand_a ^ operand_b;//XOR

    5'b01010: result = operand_a | operand_b; // OR

    5'b01011: result = operand_a & operand_b; // AND

    default:result = 31'b0; // default
```

THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Strategy
											Vivado Synthesis Defaults (Vivado Synthesis 2020)	Vivado Synthesis Default Reports (Vivado Synthe
											Vivado Implementation Defaults* (Vivado Implementation 2020)	Vivado Implementation Default Reports (Vivado Ir

Instruction_decoder Tasarımı:

- Bir işlemcinin talimat kod çözücüsü, bazen salt okunur bellek biçiminde, bazen de sıradan bir birleşimsel devre biçiminde olan bir birleşimsel devredir. Amacı, bir talimat kodunu, talimat için mikro kodun başladığı mikro bellekteki adrese çevirmektir.



Kullanılan Yazılım

Araçlar: Vivado Design Suite(Xilinx) , System Verilog dili

Aşağıda da görüldüğü üzere Pass yazısı çıktığı için testimiz başarıyla sonuçlanmıştır.Yani Kodlarımız doğru çalışmaktadır.

```
PC: 0040023c, Inst: 00200e93, Addr: 00000002, Rd-Dt: xxxxxxx
PC: 00400240, Inst: 01300e13, Addr: 00000013, Rd-Dt: xxxxxxx
PC: 00400244, Inst: 01d11463, Addr: 00000001, Rd-Dt: xxxxxxx
PC: 00400248, Inst: 01c01a63, Addr: 00000000, Rd-Dt: xxxxxxx
PC: 0040025c, Inst: ff000513, Addr: fffffff0, Rd-Dt: xxxxxxx
PC: 00400260, Inst: 00100593, Addr: 00000001, Rd-Dt: xxxxxxx
PC: 00400264, Inst: 00b52023, Addr: fffffff0, Rd-Dt: xxxxxxx
Pass
```

Şekil 2

SONUÇLAR

Projenin doğru çalıştığının test edilmesi için başlangıç test kodları ile test edilmiştir.

Eğer projenizde kod hatası veya herhangi bir çalışma hatası yok ise **Şekil 2** de kırmızı kutu içerisindeki bölgede **'Pass'** yazısını görücekseniz.

Bu proje kapsamında ALU'nun desteklediği işlemler ve operasyon kodlarını , ALU ve Instruction Decoder modüllerini , işlemcileri test etmeyi öğrendik.

PROJE EKİBİ

Mehmet Rauf Füzün

Okul numarası: 190301023

Doğum Yeri: Akhisar

raufozgecmis.pdf

Doğum Tarihi: 13.11.2001

Okuduğu Lise:Akhisar Anadolu İmam Hatip Lisesi

Ahmet Batuhan Yılmaz

Okul numarası: 190301008

Doğum Yeri: Sivas

Doğum Tarihi: 06.11.2001

Okuduğu Lise:Ankara Artı Destek Anadolu Lisesi

ahmetb_ozgecmis.pdf

Mert Meriç Karadeniz

Okul numarası: 190301013

Doğum yeri:İstanbul

Doğum Tarihi:17.09.2001

Okuduğu Lise:Mustafa Kemal Anadolu Lisesi

mertmericpdf.pdf

Hayat Zehra Demir

Okul numarası: 190301014

Doğum Yeri:Antalya

Doğum Tarihi:20.09.2001

Okuduğu Lise:Antalya İstek Koleji Lisesi

hayatzehra.pdf

Referans Dosyalar

http://www.levent.tc/courses/computer-architecture/syllabus-7

https://www.fpga4student.com/2017/06/Verilog-code-for-ALU.html