FENERBAHÇE ÜNİVERSİTESİ
Bilgisayar Mühendisliği Bölümü
İstanbul, Türkiye

FPGA KULLANARAK 9 KOMUTLUK İŞLEMCİ (FB-CPU) RTL TASARIMI

COMP-201 Mantıksal Sistem Tasarımı
2020-2021 Güz Dönemi

Proje Teslim Raporu

RECEP GEMALMAZ
200301501
recepgemalmaz@stu.fbu.edu.tr

ÖZET

Tarih boyunca insan oğlu kendi yapabildiklerinden daha fazla aritmetik işlem yapabilen makinalar geliştirmişlerdir. Bu geliştirilenler çok ilkel tasarımlarla başlasa da insan oğlunun gittikçe artan bilgi birikimiyle oldukça karmaşık tasarımlar da meydana çıkmaya başlamıştır. Son yıllarda farklı hesaplama teknikleri barındıran birçok bilgisayar mimarisi geliştirilmiştir. Bunlardan en çok kabul görenlerden biri de Von-Neumann mimarisidir. Bu çalışma, Von-Neumann mimarisi baz alınarak, FPGA (Field Programmable Gate Array) üzerinde tasarlanan bir mikroişlemci olan FBCPU' yu konu edinmektedir. Buna ek olarak işlemcinin desteklediği operasyonların RTL tasarımı yapılıp, test yazılımları kullanılarak test edilecektir. Proje sonunda basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalışıp, makine dilindeki kod parçacıklarını nasıl yürütebildiği gözlemlenecektir.

ABSTRACT

Throughout history, the son of man has developed machines that can perform more arithmetic operations than they can do themselves. Although these developments began with very primitive designs, quite complex designs began to emerge with the increasing knowledge of the human son. In recent years, many computer architectures have been developed that incorporate different computational techniques. One of the most widely accepted of these is the Von-Neumann architecture. This study focuses on FBCPU microprocessor designed on FPGA (Field Programmable Gate Array) based on the Von-Neumann architecture. In addition, the operations supported by the processor will be designed by RTL and tested using test software. At the end of the project, it will be observed how the RAM, Control Unit and storage in a simple processor can work together and execute machine-language code snippets.

1-GİRİŞ

1.1 Projenin Tanımı

Bu proje kapsamında FB-CPU isminde bir işlemcinin Verilog dili ile RTL tasarımı ve tasarlanan işlemci üzerinde makine dili ile yazılan çeşitli kod parçacıkları yazılacaktır. Proje sonunda basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalışıp, makine dilindeki kod parçacıklarını nasıl yürütebildiği gözlemlenecektir. Kullanılacak Basys3 FPGA geliştirme kartı üzerinde FBCPU demo'su yapılacaktır.
FB-CPU 9 adet komutu desteklemektedir ve tasarlanması istenen FBCPU RTL tasarımı, Von Neumann mimarisindedir. Bu komutlar test yazılımları kullanarak test edilecektir.

1.2 Projenin Amacı

Bu projenin amacı temel bir işlemcinin çalışma prensibini öğretmeyi amaçlamaktadır.
Proje sonunda basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalıştırıp, makine dilindeki kod parçacıklarını nasıl yürütebildiğini gözlemlemek ve makine dilinde yazılan 9 farklı operasyon kodu çalıştırabilen bir işlemci tasarımı geliştirmektir.

2-SİSTEM MİMARİSİ

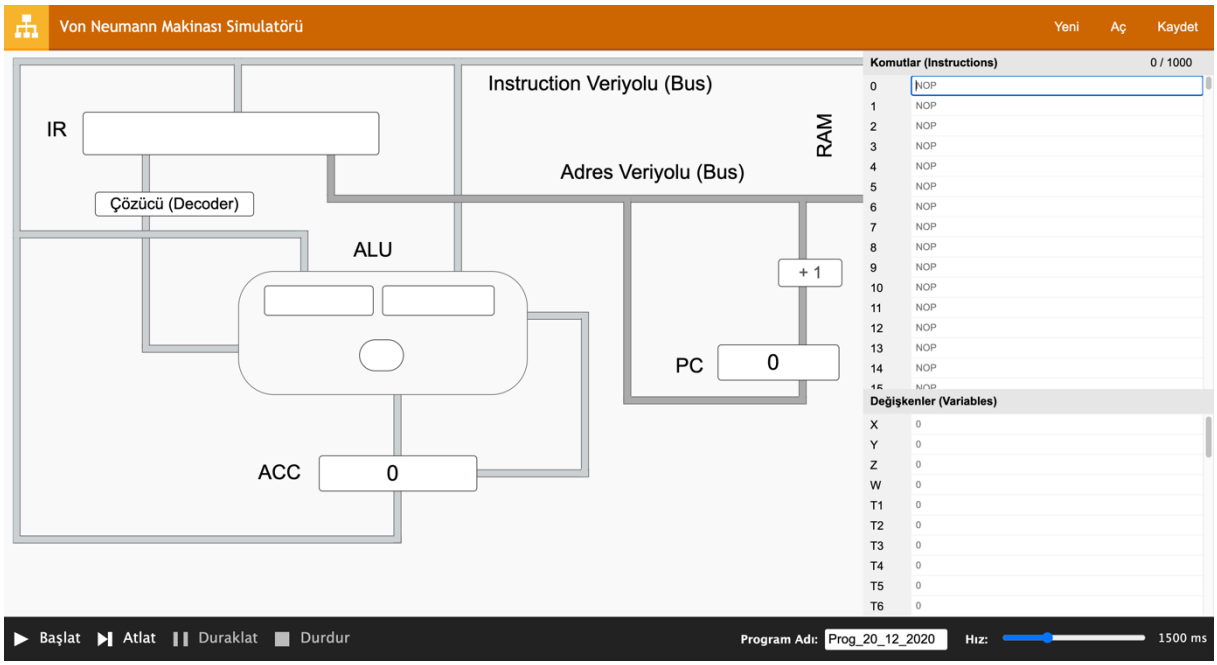
2.1-KULLANILAN ARAÇLAR

Proje kapsamında 2 araç kullanılmıştır.

- FBCPU Simulatörü
- Xilinx Vivado Design Suite

2.1.1 FBCPU Simulatörü:

FB-CPU'nun mimarisini görselleştiren, veri akışının gözlemlenebildiği "FBCPU Simulatörü" kullanılmıştır. Tasarımı gerçekleştiren ve veri akışlarını daha rahat bir şekilde gözlemleyebildiğimiz bir simülatördür.

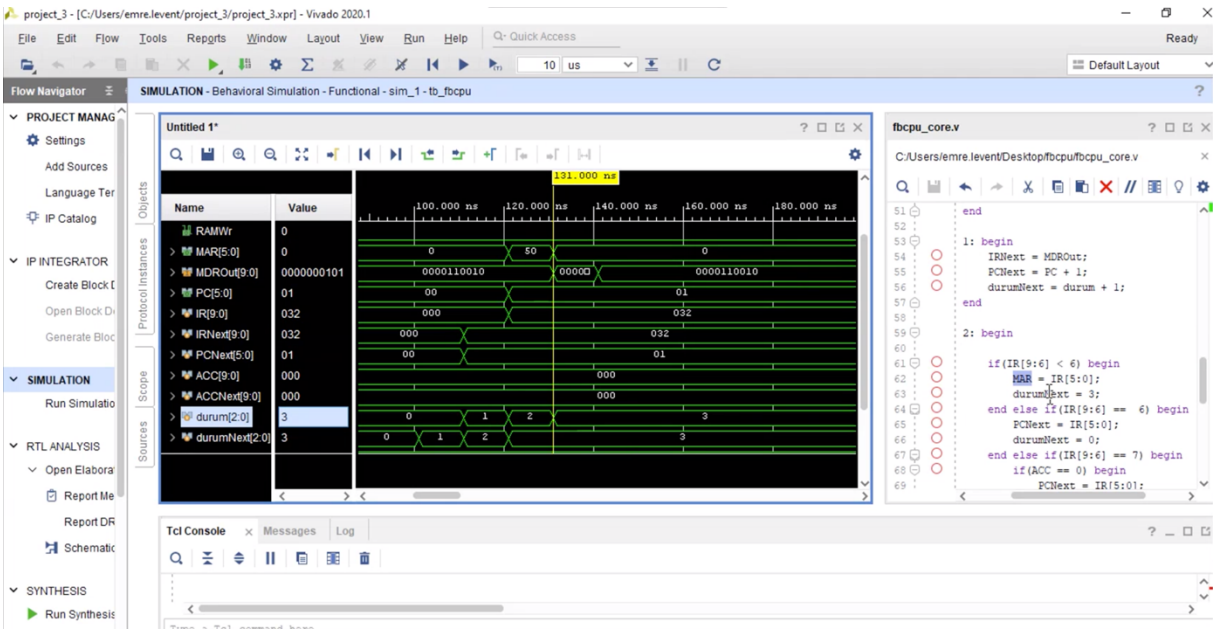


Şekil 1. FBCPU Simulatörü

Şekil 1'in sağ tarafında bulunan Komutlar ve Değişkenler belleğin (RAM) içerisinde bulunan sayılardır. Komutları 0-1'lar halinde yazmak yerine, assembly denen bir dil ile ifade ediliyor. Komutlar bölümüne, bu işlemci için geçerli komut seti ile komutlar yazılarak işlemcinin desteklediği 9 adet komutu çalıştırabiliriz.

2.1.2 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.

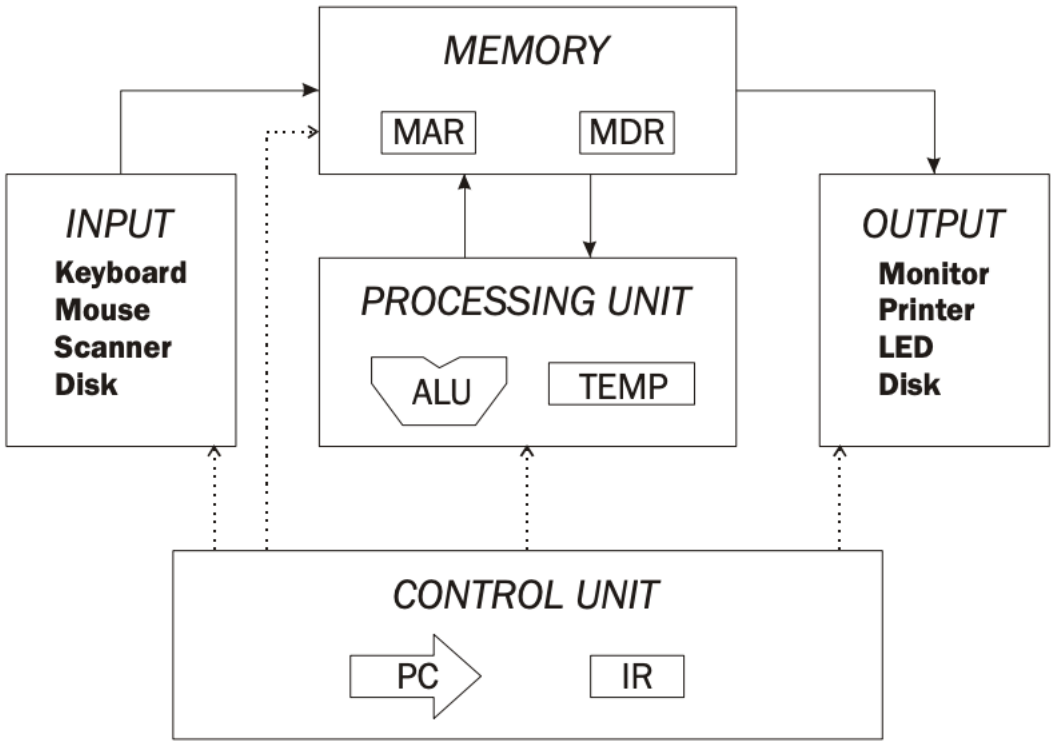


Şekil 2. Xilinx Vivado Design Suite

2.2 TASARIM

2.2.1 Von Neumann Mimarisi

Tasarlanması istenen FBCPU RTL tasarımı, Von Neumann mimarisindedir. Şekil 3'te Von Neumann Mimarisi verilmektedir.



Şekil 3. Von Neumann Mimarisi

Temel olarak 4 elemanı vardır.

• Saklayıcılar (Şekil 2'de Processing Unit'in altındaki Temp değişkeni)
• Bellek (RAM)
• İşlem Ünitesi (ALU)
• Kontrol Ünitesi

2.2.2 FB-CPU'nun Desteklediği Operasyonlar

FB-CPU'nun desteklediği operasyonlar Şekil 4'ün sol tarafında verilmektedir.

Komut Adı	Görevi	Operasyon Kodu
LOD ADDR	Yükleme (Load), Bellekteki verilen adresin içerisinden değeri alıp, ACC saklayıcısına yerleştirir. ACC = *(ADDR)	0000
STO ADDR	Kaydetme (Store), ACC'nin içerisindeki değeri alıp, bellekte verilen adrese yazar. *(ADDR) = ACC	0001
ADD ADDR	Bellekteki verilen adresteki değeri alır, ACC ile toplayıp, ACC'nin üzerine yazar. ACC = ACC +*(ADDR)	0010
SUB ADDR	Bellekteki verilen adresteki değeri alır, ACC ile çıkartıp, ACC'nin üzerine yazar. ACC = ACC - *(ADDR)	0011
MUL ADDR	Bellekteki verilen adresteki değeri alır, ACC ile çarpıp, ACC'nin üzerine yazar. ACC = ACC * (*(ADDR))	0100
DIV ADDR	Bellekteki verilen adresteki değeri alır, ACC ile bölüp, ACC'nin üzerine yazar. ACC = ACC / (*(ADDR))	0101
JMP SAYI	PC = Sayı olur.	0110
JMZ SAYI	ACC'ın değeri 0 ise, verilen sayı değerini PC'e atar, değilse işlem yapmaz.	0111
NOP	No Operation, hiçbir işlem yapılmaz.	1000
HLT	Uygulama durur	1001

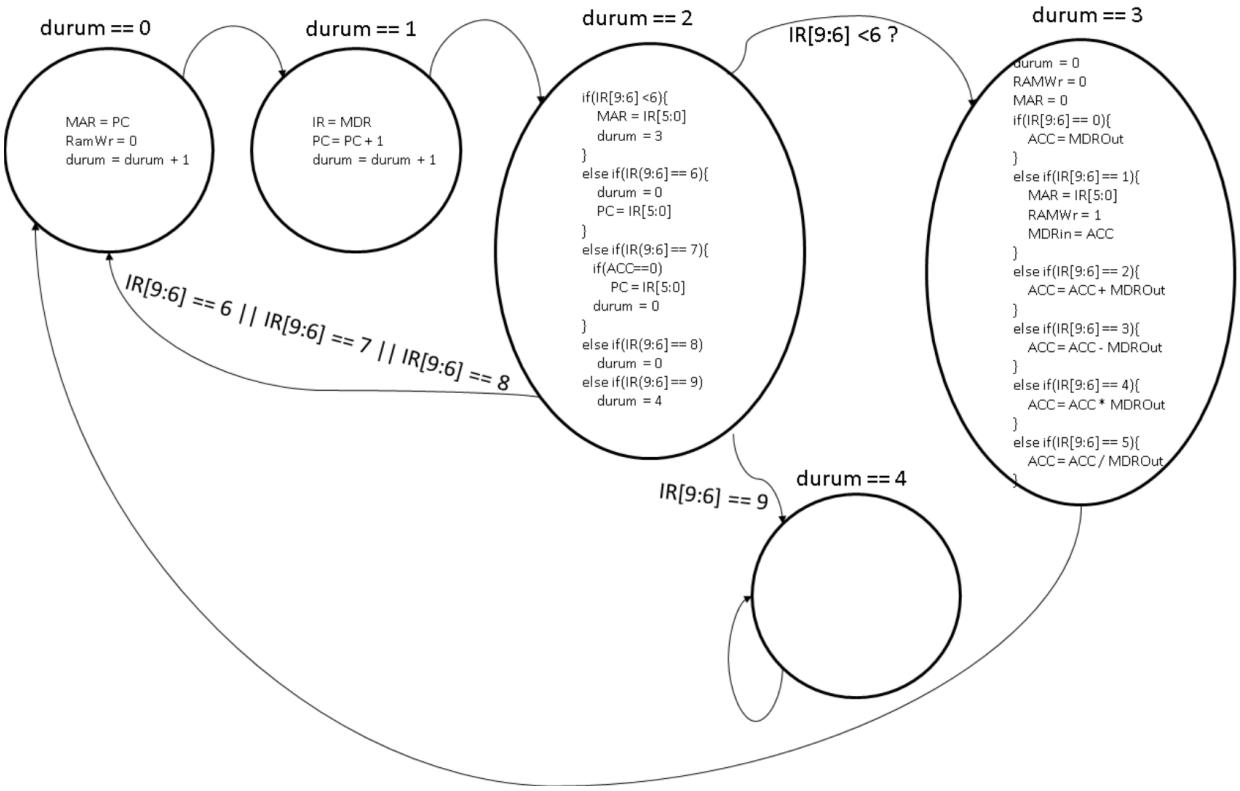
Şekil 4. FB-CPU ISA - FB-CPU Örnek Komut Binary Gösterimi

İşlemci 9 adet komutu desteklemektedir. Şekil 4'ün sağ tarafında FB-CPU'nun 10 bitlik komutunun, operasyon ve adres için bitlerinin ayrılması gösterilmiştir.

2.2.3 FB-CPU Durum Makinası Gösterimi

FB-CPU durum makinaları yöntemi ile gerçeklenmektedir. Yani bu işlemci durum ismindeki saklayıcının değerine göre 2^3 = 8 farklı durumda çalışan bir tasarımı olacaktır. FBCPU desteklemesi istenen işlemlerin tamamı 4 farklı durumda yapılabilmektedir. Diğer tüm saklayıcılar, durum saklayıcısının değişimine göre çalışacaktır. Yani durum'un değerine göre tüm saklayıcıların giriş sinyalleri değişmektedir. Diğer bir değiş ile, durum saklayıcısının değerine göre saklayıcıların üzerine başka başka sinyaller atanmaktadır ve sistemin ilerlemesi durum sinyaline bağlıdır.

Şekil 5'te FB-CPU durum makinası gösterimimi verilmiştir.



Şekil 5. FB-CPU Durum Makinası Gösterimi

Durum makinesinin çalışma prensibi;

Durum saklayıcısının aldığı değere göre o durumdaki komutların çalışmasıyla oluşrulmuştur.

Durum saklayıcısı = 0 ise:

- PC (Program Counter) saklayıcısındaki veri MAR(Memory Adress Register)'a besleniyor.
- Ram Write (RamWR) sinyali 0 besleniyor
- Durum saklayıcısı 1'e eşitleniyor.

Durum saklayıcısı = 1 ise:

- MDR (Memory Data Register)'daki veri IR (Instruction Register) saklayıcısına besleniyor.
- PC (Program Counter) saklayıcısındaki değer 1 artırılıyor.
- Durum saklayıcısı 2'ye eşitleniyor.

Durum saklayıcısı = 2 ise:

- IR'ın 6'dan 9'a kadar olan bitlerin değerinin 6'dan küçük olup olmadığına bakılıyor.
  - 6'dan küçük ise IR'ın 0'dan 5'e olan bitleri MAR'a besleniyor ve durum 3'e eşitleniyor.
- 6'dan 9'a kadar olan bitlerin değeri 6'dan küçük değil ise 6'ya eşit olup olmadığı kontrol ediliyor.
  - 6'ya eşit ise durum saklayıcısı 0'a eşitleniyor. IR'ın 0'dan 5'e olan bitleri MAR'a besleniyor.
- 6'dan 9'a kadar olan bitlerin değeri 7'ye eşit olup olmadığı kontrol ediliyor
  - 7'ye eşit ve ACC(Accumulator) de 0'a eşit ise IR'nin 0'dan 5'e kadar olan bitleri PC'a besleniyor ve durum 0'a eşitleniyor.
- 6'dan 9'a kadar olan bitlerin değeri 8'e eşit olup olmadığı kontrol ediliyor.
  - 8'e eşit ise durum 0'a eşitleniyor.
- 6'dan 9'a kadar olan bitlerin değeri 9'a eşit olup olmadığı kontrol ediliyor.
  - 9'a eşit ise durum 4'e eşitleniyor.

Durum saklayıcısı = 3 ise:

- Durum saklayıcısı 0'a eşitleniyor.
- RamWR sinyali 0'a eşitleniyor
- MAR değeri 0'a eşitleniyor.
- IR'ın 6'dan 9'a kadar olan bitlerin değerinin 0'a eşit olup olmadığı kontrol ediliyor.
  - 0'a eşit ise MDRout saklayıcısı ACC'ye eşitleniyor.
- IR'ın 6'dan 9'a kadar olan bitlerin değeri 1'e eşit olup olmadığı kontrol ediliyor.
  - 1'e eşit ise IR'ın 0'dan 5'e kadar olan bitleri MAR'a eşitleniyor.
  - RamWR sinyali 1'e eşitleniyor.
  - ACC değeri MDRin'e besleniyor.
- IR'ın 6'dan 9'a kadar olan bitlerin 2'ye eşit olup olmadığına kontrol ediliyor.
  - 2'ye eşitse ACC ve MDRout toplanıp ACC'e yazılıyor.
- IR'ın 6'dan 9'a kadar olan bitlerin 3'e eşit olup olmadığına kontrol ediliyor.
  - 3'e eşitse ACC ve MDRout çıkartılıp ACC'e yazılıyor.
- IR'ın 6'dan 9'a kadar olan bitlerin 4'e eşit olup olmadığına kontrol ediliyor.
  - 4'e eşitse ACC ve MDRout çarpılıp ACC'e yazılıyor.
- IR'ın 6'dan 9'a kadar olan bitlerin 5'e eşit olup olmadığına kontrol ediliyor.
  - 5'e eşitse ACC ve MDRout bölünüp ACC'e yazılıyor.

Durum saklayıcısı = 4 ise:

İşlemcideki 10 bitlik IR saklayıcısının ilk 4 biti 9 olduğu zaman ([9:6],1001), işlemci 4. duruma geçer. 4.durum halt işleminin yapıldığı durumdur. Bu durumda işlemci hiçbir işlem gerçekleştiremez ve bu durumdan çıkamaz. 4. Durumda tüm değerler durumun üstüne geri beslenmektedir. 4.durumda saklayıcıların değerleri:

- Durum = Durum
- PCNext = PC
- IRNext = IR
- ACC = ACC

2.2.4 SAKLAYICILAR

FBCPU'nun tasarımda 4 adet saklayıcı bulunmaktadır.
Bunlar;

- durum: Durum makinasında, hangi durumda olduğunu bilgisi tutulur.
- PC: RAM'deki hangi adresteki komutun çalıştığı bilgisi tutulur.
- IR: O anda çalışan komutun kendisi tutulur.
- ACC: Geçici saklama alanı.

Şekil 6'da FBCPU RTL tasarımında kullanılmış olan saklayıcılar gösterilmektedir.

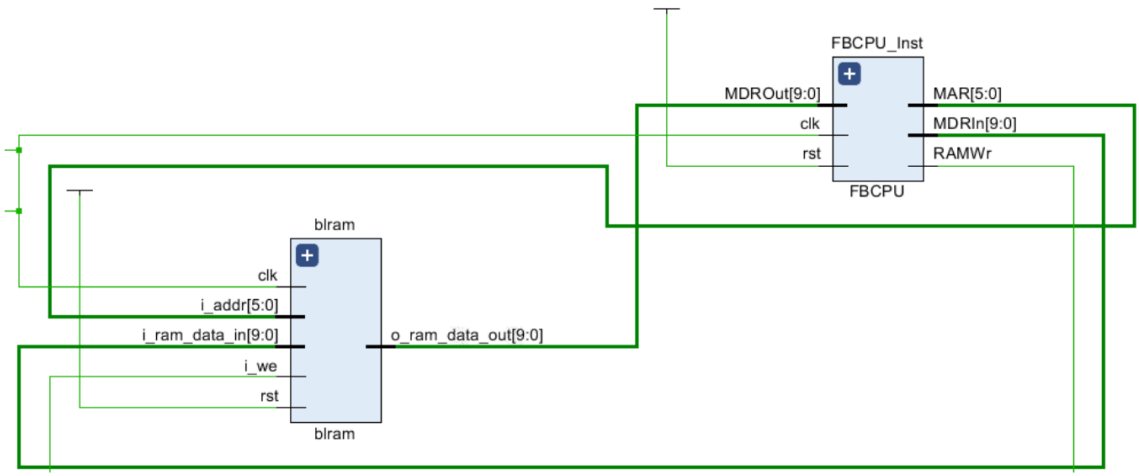
```
always@(posedge clk) begin
    durum       <= #1 durumNext;
    PC          <= #1 PCNext;
    IR          <= #1 IRNext;
    ACC         <= #1 ACCNext;
end
```

Şekil 6. FBCPU Saklayıcıları

Tasarımda giriş çıkış portlarına bağlı olan bellek sinyalleri aşağıda verilmektedir

- MAR (6 Bit): Memory Address Register isminde bir saklayıcıdır. Bu saklayıcı RAM'in adres girişine bağlanmıştır. RAM'in 2^6 lokasyonu olduğu için MAR 6 bitliktir. Saklayıcı RAM'in içerisindedir.
- MDRIn (10 Bit): Memory Data Register In, RAM'e bir veri yazılacağı zaman kullanılan saklayıcıdır. RAM'in bir lokasyonu 10 bitlik olmasından ötürü, saklayıcı 10 bittir. Saklayıcı RAM'in içerisindedir.

- RAMWr (1 Bit): RAM'e veri yazılacağı durumlarda aktif edilmektedir. 1 olmadığı durumlarda RAM'e veri yazılmaz. Saklayıcı RAM'in içerisindedir.
- MDROut (10 Bit): Memory Data Register, RAM'den veri okunacağı zaman kullanılan saklayıcıdır. RAM'in bir lokasyonu 10 bit olmasından dolayı, saklayıcı 10 bittir. Saklayıcı RAM'in içerisindedir.



Şekil 6. FBCPU ve Bellek arasındaki bağlantılar

2.2.5 Bellek (RAM, Random Access Memory)

FB-CPU'nun komutları okuyup, hesaplanan değerleri geri yazacağı bir Block RAM mekanizması bulunmaktadır. RAM'e bağlı 4 saklayıcı, clock ve reset sinyali bulunmaktadır. RAM'e bağlı saklayıcıların görevleri saklayıcılar bölümünde açıklanmıştır.

2.2.6 İşlem Ünitesi (ALU, Arithmetic Logic Unit)

Aritmetik işlemlerin gerçekleştirildiği bölümdür. FB-CPU'da 3 adet aritmetik işlem vardır. Bunlar toplama, çıkartma ve çarpma, gelen operasyon koduna göre işlemleri gerçekleştirip ACC saklayıcısına yazmaktadır.

2.2.7 Kontrol Ünitesi

Saklayıcılar, Aritmetik İşlem Ünitesi ve RAM'e verilerin birbirleri arasında transferinden sorumludurlar. İşlemci içi veri akışını yönetir.

3. KULLANILAN YAZILIMLAR

FB-CPU 10 bitlik komutunda, ilk 4 biti [9:6] operasyon kodunu, son 6 biti ise [5:0] adresi temsil etmektedir. Şekil 4'te verilmiş olan komutlar ile uygulamalar geliştirilmelidir. İşlemcinin doğru çalışıp çalışmadığı aşağıdaki kod parçacıkları ile test edilebilir.

3.1 Test Yazılım 1

```
0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32
1: 0010_110011 // ADD 51, ACC = ACC + (*51), Hex = B3
2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74
3: 1001_000000 // Halt, Hex = 240
50: 0000000101 // Hex = 5
51: 0000001010 // Hex = A
```

Şekil 7. Test Yazılım 1'in Komutları

Bellekte 50. ve 51. adresteki iki sayının toplamını 52 nolu adrese kaydeder. Adım adım aşağıdaki sırayla işlem yapılamaktadır.

- 50. Adresteki değeri LOD komutuyla (0000) ACC'nin içerisine alır.
- ADD komutuyla (0010) 51. Adresteki değeri ACC'nin içerisindeki değer ile toplar ve tekrardan ACC'nin içerisine yazar.
- STO komutuyla (0001) ACC'nin içerisinde ki değeri 52. adrese yazar.
- Halt komutuyla (1001) işlemi sonlandırır.

3.2 Test Yazılım 2

```
0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32
1: 0100_110011 // ADD 51, ACC = ACC * (*51), Hex = 133
2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74
3: 1001_000000 // Halt, Hex = 240
50: 0000000101 // Hex = 5
51: 0000001010 // Hex = A
```

Şekil 8. Test Yazılım 2'nin Komutları

Bellekte 50. ve 51. adresteki iki sayının çarpımını 52. Adrese yazar. Adım adım aşağıdaki sırayla işlem yapılamaktadır.

- LOD komutuyla (0000) 50. Adresteki değeri ACC'nin içerisine alır.
- ADD komutuyla (0100) 51. Adresteki değeri ACC'nin içerisindeki değer ile çarpıp tekrardan ACC'nin içerisine yazar.
- STO komutuyla (0001) ACC'nin içerisindeki değeri 52. adrese yazar.
- Halt komutuyla (1001) işlemi sonlandırır.

4. SONUÇLAR

- Basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalışıp, makine dilindeki kod parçacıklarını nasıl yürütebildiği gözlemledik.
- Geliştirilen FBU-CPU işlemcisi gerekli durum koşullarını sağladığında 9 adet komutu yerine getirdiğini ve 4 adet işlem yapabildiğini gözlemledik.
- Von Neumann mimarisinde inceleyerek bu yapı hakkında bilgi sahibi olduk.
- Xilinx Vivado Design Suite tasarım aracını kullanarak basit bir işlemcinin nasıl dananım tasarımı yapıldığını öğrenmiş olduk.
- Simülasyon aracını kullanarak FBCPU mimarisinde basit bir işlemcinin veri akışlarının nasıl olduğunu gözlemledik. Bir işlemcinin temel bileşenleri, çalışma prensiplerini, işlemci-bellek alışverişlerini deneyerek ve gözlemleyerek öğrenmiş olduk.
- Basys3 FPGA geliştirme kartının işlemciyle olan çalışma prensipleri ve benzerlikleri hakkında bilgi sahibi olduk.
- Durum makinesinin bir tasarım aracında donanım kodlarının nasıl ifade edildiğini ve gösterim şekillerini öğrendik.
- Durum makinesi kodlaması yapılırken kullanılması gereken saklayıcıların çalışma prensiplerini ve yine bu saklayıcıların bellek ve işlemci arasındaki veri alışverişinde neden ihtiyaç duyulduğunu öğrenmiş olduk.

5.PROJE EKİBİ

RECEP GEMALMAZ (200301501)
16.100.2000 tarihinde Kadıköy'de dünyaya geldim. Özel Sınav lisesinden mezun oldum. Şu anda Fenerbahçe Üniversitesi Bilgisayar Mühendisliği bölümünde lisans eğitimimi tamamlamaktayım.

6.REFERANS DOSYALAR

- Youtube Link:
  https://www.youtube.com/watch?v=vm8iiVlP2hs&t=11s
- Github Link:
  https://github.com/recepgemalmaz/fb_cpu_rtl_tasarim