FB-CPU RTL Tasarımı

Bedirhan İleri

Fenerbahçe Üniversitesi

Bilgisayar Mühendisliği

İstanbul, Türkiye

e-mail: {bedirhan.ileri}@fbu.edu.tr,

**Özetçe—** 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.

**Anahtar Kelimeler —** FPGA, CPU

**Abstract—** In this project, RTL design of a processor named FB-CPU with Verilog language and various code snippets written on the computer with machine language will be written. At the end of the project, it will be observed how RAM, Control Unit and Registers in a simple processor can work together and execute code snippets in the machine language. FBCPU demo will be made on Basys3 FPGA development board to be used.

**Keywords —** FPGA, CPU.

# Giriş

FB-CPU isimli işlemcinin, tasarımda Von Neumann mimarisi baz alınacaktır ve durum makinesi kullanılacaktır. RAM, saklayıcılar, kontrol üniteleri ve ALU birlikte kullanılarak yazılan kod çalıştırılacak ve assembly dilindeki test kodlarıyla test edilecektir.

# Sistem Mimarisi

FB-CPU tasarlanırken Xilinx Vivado Design Suite; Verilog,

VHDL vb… donanım tasarım dillerini alarak, FPGA’e konfigüre

edilebilecek (Xilinx firması FPGA’leri için .bit uzantılı dosyalar)

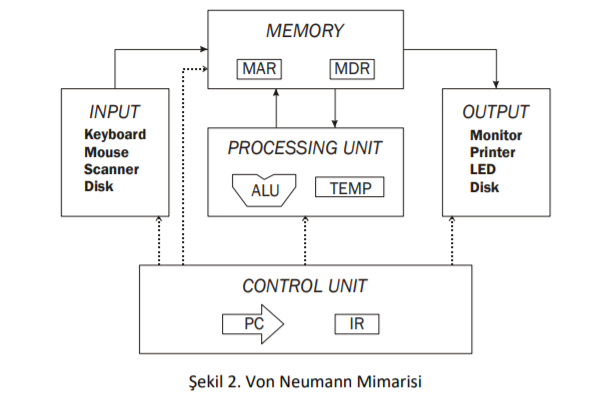
tasarım dosyasını oluşturur.

FB-CPU’nun mimarisini görselleştiren, veri akışının

gözlemlenebildiği “FB-CPU Simülatörü” test yazılımlarının nasıl

çalıştığını görmemize yardımcı olmuştur.

FB-CPU RTL tasarımı, Von Neumann mimarisindedir.



Temel olarak 4 elemanı vardır; saklayıcılar (Şekil 2’de

Processing Unit’in altındaki Temp değişkeni), bellek(RAM),

İşlem Ünitesi (ALU), Kontrol Ünitesi.

Von Neumann mimarisinde kullanılan ünitelerin görevleri:

Bellek, operasyon komutlarını ve değişkenleri tutmaktadır.

İşlemci Ünitesi, aritmetik ve mantık işlemlerini yapmaktadır.

Kontrol Ünitesi, komutların çözülmesi için gereklidir.

Saklayıcılar, çeşitli görevlerde kullanılan saklama alanları.

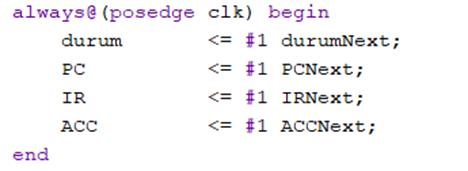
FB-CPU’nun tasarımda 4 adet saklayıcı bulunmaktadır.

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.



FB-CPU durum makinaları yöntemi ile gerçeklenecektir. Yani

bu işlemci durum ismindeki saklayıcının değerine göre 23 = 8

farklı durumda çalışan bir tasarımı olacaktır (İşlemcinin

desteklemesi istenen işlemlerin tamamı 8 farklı durumda

yapılabilmektedir). Diğer tüm saklayıcılar, durum saklayıcısının

değişimine göre çalışacaktır. Yani durumun değerine göre tüm

saklayıcıların giriş sinyalleri değişmektedir. 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.

Bellek (RAM, Random Access Memory); FB-CPU’nun

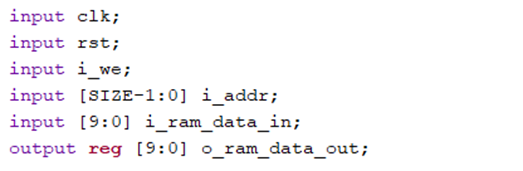
komutları okuyup, hesaplanan değerleri geri yazacağı bir Block

RAM mekanizması bulunmaktadır. Test kodunun instantiate

ettiği bellek memory.v dosyasında bulunmaktadır. RAM’e bağlı

4 saklayıcı, clock ve reset sinyali bulunmaktadır. RAM’e bağlı

saklayıcıların görevleri saklayıcılar bölümünde açıklanmıştır.



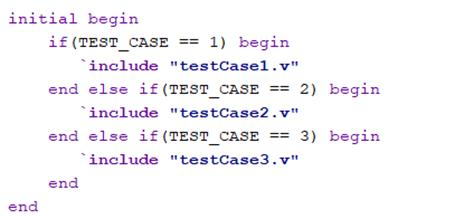
Aynı zamanda memory.v içerisinde bulunan include

testCase’ler, tb\_fbcpu.v dosyasında parametre ile çağırdığımız

test kodunun içeriğini alıp if bloğunun altına getiriyor. Bu

şekilde yapmamızın amacı daha sade ve anlaşılır olmasıdır

yoksa dosyanın içeriğini direkt kopyalasak da olurdu.



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.

Kontrol Ünitesi; Saklayıcılar, Aritmetik İşlem Ünitesi ve

RAM’e verilerin birbirleri arasında transferinden sorumludur.

İşlemci içi veri akışını yönetir.

Şekil 3’te FB-CPU’nun 10 bitlik komutunun, operasyon ve

adres için bitlerinin ayrılması gösterilmiştir.



İşlemcimizin tepe modülü olan, tasarımı barındıran

fbcpu\_core.v modülü vardır. İşlemcinin kendisini tanımlar.

memory.v isimli bir başka dosyada da RAM’in kendisini

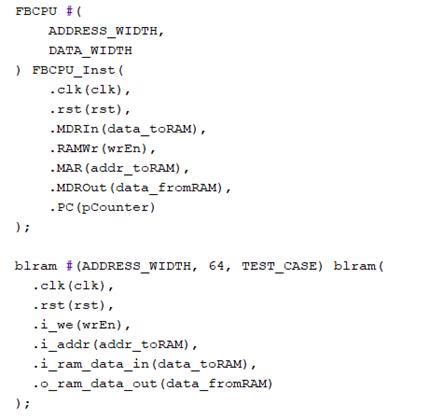
tanımlanır. Bununla birlikte bu iki dosyayı aynı anda test

edebilmek ve birbirine bağlamak için tb\_fbcpu.v isimli test

bench kodu vardır. Bu test bench kodunda fbcpu\_core ile block

RAM’i bulunduran memory.v instance edilmiştir. Burada ikisinin

sinyalleri birbirine bağlanmaktadır.



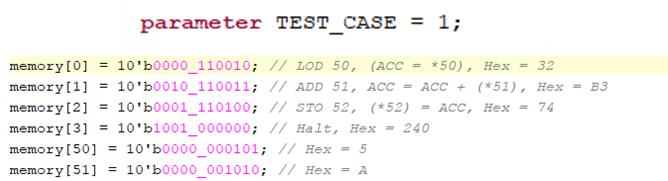
Aynı zamanda test bench kodunun içinde TEST\_CASE

isminde bir paramatremiz var. Bu test case hocanın bize

sunmuş olduğu 3 adet kodu ifade ediyor. İşlemcinin

okuyabilmesi için belleklere yazılmış olan programı test

etmemizi sağlayan test dosyalarıdır.



Örneğin, test bench kodundaki parametre ile TEST\_CASE 1’i

çağırırsak kodu çalıştırdığımızda assembly yani makine diliyle

yazılmış bu dosya çağırılmış olur. Bu dosyalardan hangisinin o

esnada test edileceğini parametreye test casenin değerini

vererek söylüyoruz.

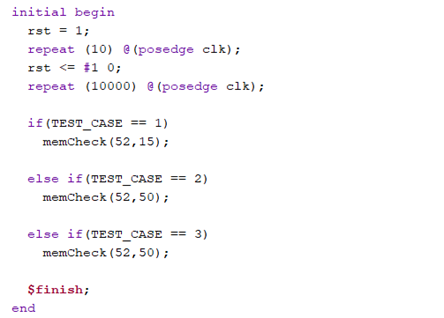
Test bench’imizin içinde aynı zamanda parametreden

girdiğimiz değere göre gerekli değerleri atayan bir if’li seçim

mekanizması bulunmaktadır. Örneğin test case 1’i denemek

istersek işlemci çalışana kadar 10.000 cycle beklememiz

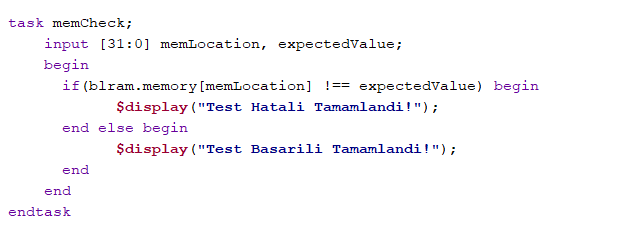
gerekmektedir.



10.000 cycle’ın sonunda 52. Adreste 15 olması gerekir. A ile

5’i toplanır. Diğer caselerde de farklı işlemlere aynı yaklaşım

uygulanır.



Bunların doğru olup olmadığını kontrol eden otomatik bir

yapı var. Simülasyonu başlattığımız zaman ekrana, testin hatalı

tamamlanıp tamamlanmadığını gösteren bir çıktı veriyor.

Fbucpu\_core.v’de önce saklayıcıları tanımladık sonra da

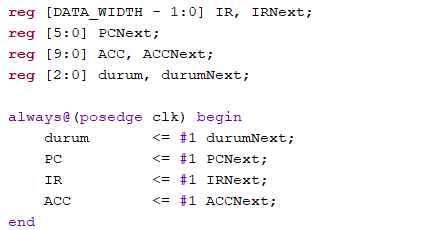
always@(posedge clk) bloğunun içerisinde saklayıcılarımızın

atamalarını gerçekleştirdik. #1 diyerek de bir cycle sonra

atamanın gerçekleşmesini sağladık. Yani yükselen kenar gelir

gelmez değil de gelmesinden bir nano saniye sonra çıktı

vermeye başlayacak şekilde tasarlandı.



Tasarımdaki bir diğer önemli nokta ise RAM’e giden

sinyalleri ayrıca bir saklayıcıdan geçirmiyoruz. RAM’e giderken

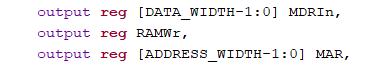
zaten içinde saklayıcı var. 2 defa saklayıcıdan geçirmiş gibi

oluyoruz bu yüzden de geri veri gelirken 2 cycle beklemek

zorunda kalıyoruz. Bu yüzden alt ifadelerde bunların nextli olan

ifadeleri değil direkt kendileri kullanılıyor. Yani bunlar

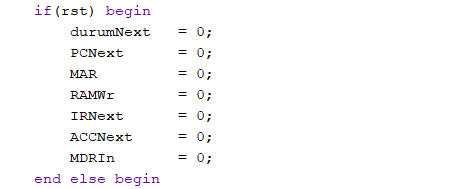
kombinasyonel devre olarak dışarı çıkıyorlar.



İlk reset anında bu durumlarla beraber beklemede. Rst biter

bitmez case’de durum saklayıcısının değerine bakacak şekilde

aktifleşir.



# Kullanılan Yazılım

Bu projede 3 tane test yazılımı kullanıldı. 1. test yazılımı, FB-CPU için bellekte 50 ve 51 adresteki iki sayının toplamını 52 no’lu adrese kaydeden uygulamadır. 50. adresteki değer LOD komutuyla ACC’nin içerisine alınıp ADD komutuyla 51. adresteki değer, ACC’nin içerisindeki değerle toplanır. Sonra STO komutuyla ACC’nin içerisindeki değer 52. adrese yazılıp Halt komutuyla işlem sonlandırılır. 2. test yazılımı, bellekte 50 ve 51. adresteki iki sayının çarpımını 52 nolu adrese kaydeden uygulamadır. LOD komutuyla 50. adresteki değeri ACC’nin içerisine alıp ADD komutuyla 51. adresteki değeri ACC’nin içerisindeki değer ile çarptıktan sonra yeniden ACC’nin içerisine yazar. STO komutuyla ACC’nin içerisindeki değeri 52. adrese yazıp Halt komutuyla işlemi sonlandırır. 3. test yazılımı, çarpma operasyonu kullanmadan 50. adresteki sayıyı 51. adresteki sayı defa toplayıp 52 nolu adrese yazan uygulamadır.

# Sonuçlar

Geliştirilen FB-CPU işlemcisi 9 adet komutu yerine getirip 4 adet işlem yapabilmektedir. Bu proje sayesinde basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalışıp makine dilindeki kod parçacıklarını nasıl yürütebildiğini gözlemledik. Simülasyon aracını kullanarak FB-CPU mimarisinde basit bir işlemcinin veri akışlarının nasıl olduğunu, çalışma prensiplerini, işlemci-bellek alışverişlerini deneyerek ve gözlemleyerek öğrenmiş olduk. Ayrıca durum makinesinin bir tasarım aracında donanım kodlarının nasıl ifade edildiğini ve gösterildiğini öğrendik.

##### Proje Ekibi

06/12/2001 tarihinde Üsküdar’da doğdum. Prof. Dr. Nabi Avcı Anadolu Lisesi mezunuyum.Fenerbahçe Üniversitesi’nde Bilgisayar Mühendisliği 3. sınıf öğrencisiyim.

##### Referans Dosyalar

<https://www.youtube.com/watch?v=0Gptzkg8UYw>

<https://github.com/DeveloperBedirhan/FB-CPU-RTL-Design>

##### Kaynaklar

1. *<http://www.levent.tc/files/courses/electronic_devices/project/BLM301_proje_spesifikasyonlari.pdf>*