Fenerbahçe Üniversitesi

BLM 202 – Bilgisayar Mimarisi

LAB 5: System Verilog for Synthesis

Önemli Not: Aşamaları tamamladıkça, dersin hocası veya asistanı yanınıza çağırarak, tamamladığınız aşamayı gösterdikten sonra diğer aşamaya geçiniz.

LAB Hakkında:

System Verilog dili ile sentezlenebilir tasarım uygulamaları yapılacaktır..

Üniversitede 5 adet FPGA kartı online eğitim için sunucuya bağlanmıştır.

Sunucudaki FPGA'lere

- 10.18.0.136::3121
- 10.18.0.136::3122
- 10.18.0.136::3123
- 10.18.0.136::3124
- 10.18.0.136::3125

Adreslerinden erişilebilir. FPGA'lere erişim gösterilecektir.

Yapılacak tasarımlar online eğitim nedeniyle, FPGA'lerin üzerindeki butonlara basıp, LED'leri gözlemlemek mümkün değildir.

Dolayısıyla bunu dolaylı olarak mümkün kılmak için, yapılacak tasarımlarda Xilinx'in VIO (Virtual Input Output) IP'si de birlikte kullanılacaktır.

LAB'ın aşamaları ve puanlar:

Lab 0 – Başlangıç Tasarımı

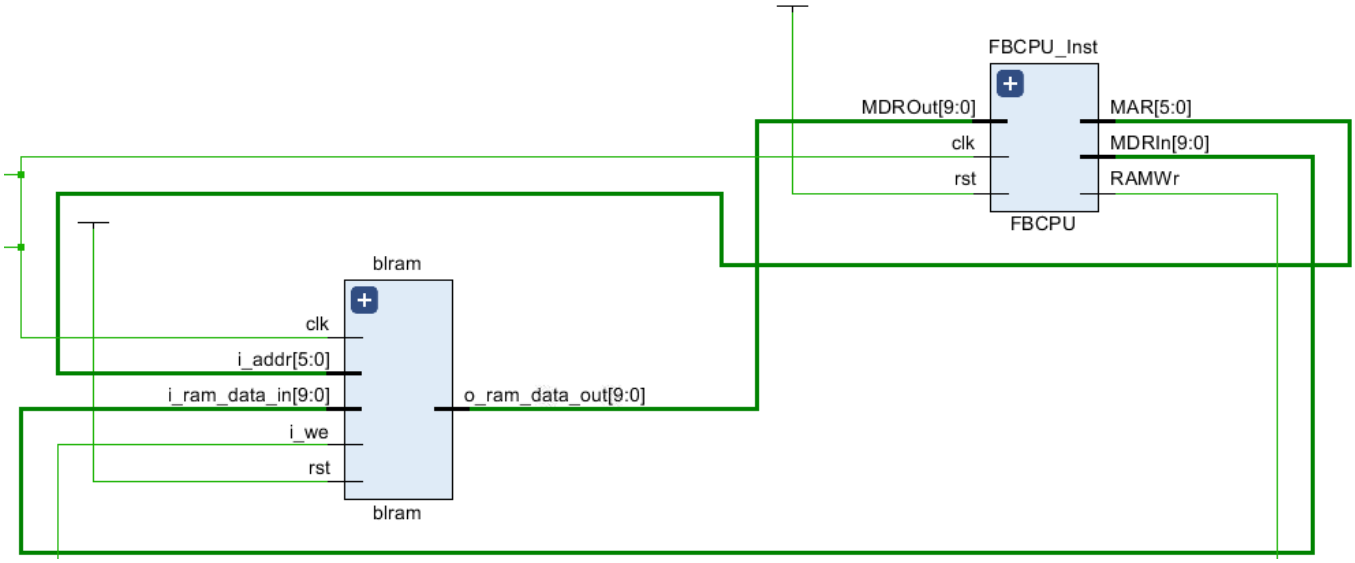
FB-CPU isminde bir işlemcinin Verilog dili ile RTL tasarımı verilmiştir.

Basit bir işlemcideki RAM, Kontrol Ünitesi ve Saklayıcıların bir arada çalışıp, makine dilindeki kod parçacıklarını nasıl yürütebildiği gözlemlenebilen bir testbench kodu vardır. Bu kod ile işlemcinin doğru çalışıp çalışmadığı 3 farklı test case girişi verilebilerek kontrol edilmektedir.

Başlangıç tasarımı rar arşivinin içerisinde 6 adet dosya bulunmaktadır. Bunlar;

- fbcpu_core.v: İşlemcinin tasarımını barındırır.
- tb_fbcpu.v: İşlemciyi test edecek olan komutları besler ve sonucun doğru olup olmadığını kontrol eden testbench tasarımıdır.
- memory.v: Komutlar ve verilerin tutulduğu, Block RAM olarak tasarlanmış bellektir.
- testCase1.v: Örnek yazılım başlığında verilen 1. Yazılımı içerir.
- testCase2.v: Örnek yazılım başlığında verilen 2. Yazılımı içerir.
- testCase3.v: Örnek yazılım başlığında verilen 3. Yazılımı içerir.

tb_fbcpu.v dosyasında verilen testbench tasarımı, fbcpu_core.v, memory ve testCase1-2-3 dosyalarını alt modül olarak kullanmaktadır. Şekil 1'de testbenchte modüllerin birbirlerine olan bağlantıları verilmektedir.



Şekil 1. Testbench Bağlantılar

Temel olarak 4 elemanı vardır.

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

FB-CPU'nun desteklediği operasyonlar Tablo 1'de verilmektedir.

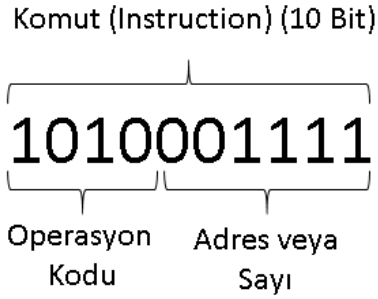
Tablo 1. FB-CPU ISA (Instruction Set Architecture)

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.	0010

	ACC = ACC +*(ADDR)	
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
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

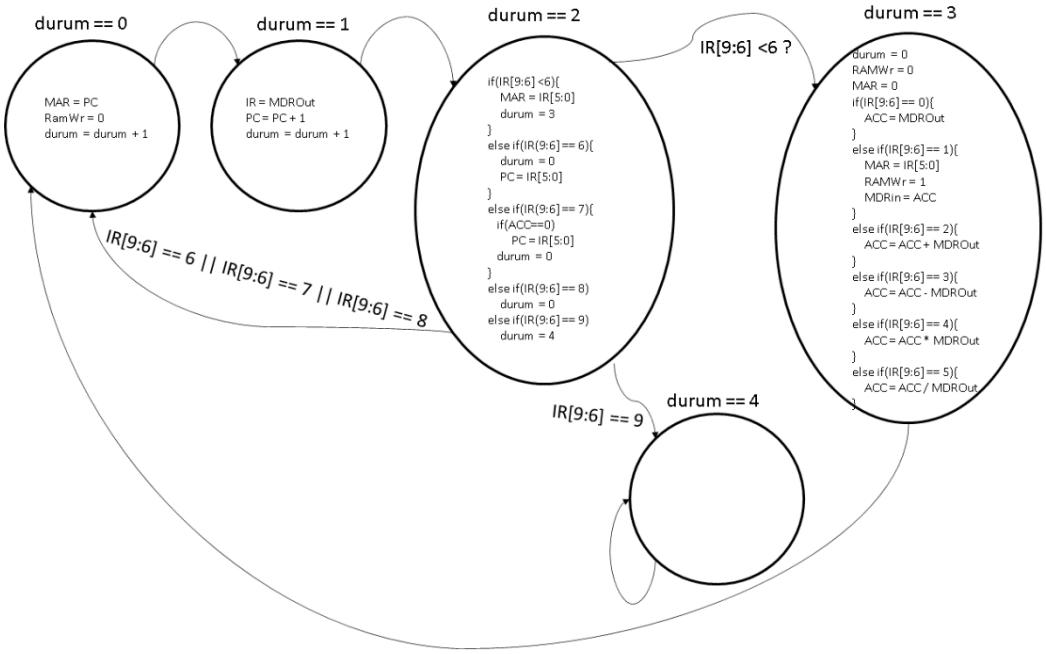
İşlemci 9 adet komutu desteklemektedir.

Şekil 3'te FB-CPU'nun 10 bitlik komutunun, operasyon ve adres için bitlerinin ayrılması gösterilmiştir.



Şekil 3. FB-CPU Örnek Komut Binary Gösterimi

FB-CPU'nun durum diyagramı olarak ifade edilmiş hali Şekil 4'te verilmektedir. İşlemcinin adım adım yapması gereken işler bir arada göstermektedir.



Şekil 5. FB-CPU Durum Makinası Gösterimi

Verilen başlangıç tasarım kullanılarak LAB 1-7 gerçekleştirilecektir. Her aşamada kodun sentezlenebilir olduğunu kontrol ediniz.

Lab 1 – TypeDef Kullanımı

Verilen başlangıç tasarımda bir durum makinası kullanılmıştır.

Bu durum makinası sabit sayılara göre hareket etmektedir. Okun okunabilirliğini arttırmak için;

0. durum'a FETCH_1
1. durum'a FETCH_2
2. durum'a DECODE
3. durum'a EXECUTE

İsimlerini enum kullanarak veriniz.

10 bit olarak tasarlanmış olan işlemcinin, 10 bitlik ifadesini WORD olarak typedef tanımlayınız.

6 bitlik bir tanım daha yaparak bu tanıma ADDR ismini veriniz ve değişkenleri bu tanımlara göre düzenleyiniz.

Tasarımda işlemcinin baktığı operasyon kodları vardır. Bu kodlar sabit sayılar olarak ifade edilmiştir. Sabit sayılar yerine enum ifadesi ile operasyon tablosunda verilen operasyon isimleri ile kodun okunabilirliğini arttıracak değişiklikleri yapınız.

Lab 2 – Always FF & Latch Blokları

Tasarımda verilog syntax'inden kalan;

always@(posedge clk)
always@(*)

blokları bulunmaktadır.

Bu blokları system verilog'a yeni eklenmiş olan

Always_ff
Always_latch

Blokları ile değiştiriniz.

Lab 3 – Fonksiyonlar

Tasarımda DECODE durumunda, gelen operasyon kodunun 6’dan küçük olmasına göre yani EXECUTE durumuna atlaması gerekip gerekmediği kontrol edilmektedir.

Bu kontrolü bir fonksiyona bağlayarak gerçekleştiriniz.

Fonksiyon argüman olarak operasyon kodunu alıp, geriye 6’dan küçük ise doğru, değil ise yanlış döndürmelidir.

Fonksiyonun döndürdüğü değere göre DECODE durumunda, atlama yapıp yapılmayacağına karar verilmelidir.

Lab 4 – Localparam

LAB 3’te tasarıma eklenen kontrol fonksiyonunun içinde sabit olarak 6 sayısından küçük olması kontrolü tasarlanmıştır.

Bu sabit 6 sayısını başka bir modülden güncellenememesi ancak kendi içinde tasarımcı istediğinde güncelleyebilmesi için localparam türünde tanımlayarak, fonksiyonu bu parametreden kontrol yapması için gerekli düzenlemeleri yapınız.

Lab 5 – Struct

Tasarıma yeni bir struct ekleyerek içeriğine;

6 bitlik data ve 4 bitlik opCode yazınız.

Bu struct IR saklayıcını ifade edecektir.

FETCH_2 durumunda kullanılan IR saklayıcının değerlerini doğrudan üzerine atmak yerine opCode ve data değişkenlerini, MDROut'u ayırarak atayınız.

Bu aşamadan sonra kodun içerisinde IR'nın son 4 bitine yanı opCode'a bakan kısımları, IR.opCode olarak;

İlk 6 bitine bakan yani data kısımlarını ise IR.data olarak güncelleyiniz.

Lab 6 – Package

Tasarımda kullanılan tüm fonksiyon, enum, typedef, struct ve parametre yapılarını bir package yapısı içerisine taşıyınız.

Bu package yapısına cpu_pack ismini veriniz.

cpu_pack'ı FBCPU modülünün bulunduğu system verilog dosyasından import ediniz.

Lab 7 – Interface

İşlemcinin belleğe giden sinyallerini gruplayarak bir interface hazırlayınız.

Bu interface değişikliği nedeniyle, tasarımın içinde bulunan sinyallerin'de isimlerinin önüne yeni tanımlanacak olan interface'nin ismi eklenecektir. Yani yeni tanımlanacak interface'nin ismi x ise x.clk olarak isimlendirme yapılması gerekecektir.

Testbecnh'in kullanılmaya devam edebilmesi için aynı değişiklikleri tb_fbcpu ve memory dosyalarında da gerçekleştiriniz.

Interface tanımına modport yapısı koyarak giriş ve çıkış tanımlamalarını hem tasarım dosyası olan işlemci için hemde testbench tarafı için tanımlayınız.

İşlemci tarafı için modprob tanımı ismi forCPU, testbench tarafı için ise forMemory olarak isimlendiriniz.