	TOBB Ekonomi ve Teknoloji Üniversitesi **Bilgisayar Mühendisliği Bölümü** **Elektrik ve Elektronik Mühendisliği Bölümü**	19 Ağustos 2023 BİL 265/264/264L – Mantıksal Devre Tasarımı ve Laboratuvarı 2022 – 2023 Öğretim Yılı Yaz Dönemi Final

AÇIKLAMALAR:

1. <Projenizin bulunduğu dizin>\<Proje ismi>\<Proje ismi>.srcs\sources_1\new → dizininde yazdığınız ".v" uzantılı dosyaları bulabilirsiniz. Simülasyon dosyalarını ise aynı uzantıda .srcs'den sonra \sim_1 klasöründe bulabilirsiniz.
2. **1.talimata uyulmaması ve dosya isimlerinin yanlış yazılması durumlarında toplam puanınız üzerinden 20 puan kırılacaktır.**
3. **Dosya gönderimi için sorularda belirtilen ".v" uzantılı dosyalarınızı "isim_soyisim_numara_final" isimli bir klasöre attıktan sonra klasörü sıkıştırınız (.zip) ve sınav sırasında gözetmenin getireceği USB ya da Uzak'a yüklemeye hazır olacak şekilde bekleyiniz .**

Sınavda Göndermeniz Gereken .v dosyaları:(Gönderim yapmak istemediğiniz soruları eklemek zorunda değilsiniz.)

- faiz.v
- kimlik.v
- bavul.v
- odeme.v
- ucak.v
- havalimani.v
- havalimani_boru.v

1. [35 puan] Faiz hesabı

Oluşturacağınız modüle "faiz"(oluşacak dosya "faiz.v") ismini verin. İstenen modülü Verilog davranışsal modelleme kullanarak gerçekleştirin.

Her saat darbesinde, giriş olarak 4 bitlik bir enflasyon değeri gelmektedir. Bu değer yüzde olarak aylık enflasyonu göstermektedir. 3.saat darbesinden(3.aya karşılık gelmektedir) başlayarak her saat darbesinde, geçmiş 3 aylık periyodun kümülatif enflasyonunu hesaplayarak, bu enflasyon değerinden 2 birim yüksek olacak şekilde bir faiz değerini hesaplayarak dışarı veren bir modül tasarlayın. Modül, 3.saat darbesinden başlayarak her saat darbesinde faizi hesaplayarak çıkış olarak vermelidir. reset sinyali geldiğinde, saat ile senkron şekilde devrenin durumunu sıfırlamanız gerekmektedir.

Devrenin girişleri:

saat: 1 bitlik saat sinyali
reset: 1 bitlik reset sinyali (senkron)
enflasyon: 4 bitlik aylık enflasyon değeri

Devrenin çıkışı:

faiz: 6 bitlik faiz değeri

Örnek:

saat	**1**	**2**	**3**	**4**	**5**	**6**	**7**
enflasyon	12	10	6	5	9	15	..
kümülatif enflasyon	12	1.12x1.10 =1.23 => **23**	1.12x1.10x1.06 =1.30 => **30**	1.10x1.06x1.05 =1.22 => **22**	1.06x1.05x1.09 = 1.21 => **21**	1.05x1.09x1.15 = 1.31 => **31**	..
faiz	0	0	32	24	23	33	..

2. [75 puan] Havalimanı

[55 Puan] a) Bu kısımda 5 adet modül oluşturacaksınız. Modüllerinize "kimlik", "bavul", "odeme", "ucak" ve "havalimani" isimlerini verin(oluşacak dosyalar: kimlik.v,bavul.v,odeme.v,ucak.v,havalimani.v). Modüllerin birbiriyle ilişkisini ve giriş çıkışlarını şekilde görebilirsiniz.

1. **"kimlik" modülü:**

Havalimanının girişinde yapılan kimlik kontrolünün gerçekleştiği modüldür. Modül, her saat darbesinde giriş olarak *BIT* bitlik bir kimlik numarası ve 1 bitlik bir uyruk bilgisi (0:TC, 1:Yabancı) alır. TC uyruklu birine ait bir kimlik numarası gelmişse, bu numara hafızada saklanacak **"yerli.mem"** dosyasındaki 10 adet numaradan biri ise "1", diğer durumda ise "0" olarak gecerli çıkışını üretir. Benzer şekilde, yabancı birine ait bir kimlik numarası gelmişse, bu numara hafızada saklanacak **"yabanci.mem"** dosyasındaki 10 adet numaradan biri ise "1", diğer durumda ise "0" olarak gecerli çıkışını üretir. Modül, basla sinyali geldiğinde işleme başlar ve bir saat darbesinde kendi işlemini bitirir. (yerli.mem ve yabanci.mem dosyalarındaki veriler binary şekilde ve her satırda bir veri olacak şekilde tutulmaktadır, kendiniz için .mem dosyalarını oluşturabilirsiniz fakat .mem dosyalarını göndermeniz istenmiyor.)(İpucu: $readmemb("dosya.mem", bellek_reg); komutu binary bellek dosyasını okumayı sağlar.)

Devrenin parametre girişi:

BIT: kimlik numarası bit sayısını belirleyen parametre girişi (varsayılan değer: 6)

Devrenin girişleri:

saat: 1 bitlik saat sinyali

reset: 1 bitlik reset sinyali (senkron)

basla: 1 bitlik devrede işlemi başlatan giriş sinyali

kimlik_no: BIT bitlik kimlik numarası

uyruk: 1 bitlik uyruk bilgisi

Devrenin çıkışları:

gecerli: 1 bitlik geçerli/değil bilgisi

bitti: 1 bitlik devrenin çalışmasını bitirdiğini belirten çıkış sinyali

2. **"bavul" modülü:**

Bavulun ağırlığına göre ödenecek olan ücreti hesaplayan modüldür. Modül, her saat darbesinde giriş olarak 6 bitlik bir ağırlık bilgisi alır, 8 bitlik ücreti hesaplayarak çıkış olarak verir. Ödenecek ücreti hesaplarken, bu ağırlığın yanında uçaktaki toplam yükü de dikkate alır. Toplam yük 60'dan küçükse ücret 45 TL, 200'den büyükse ağırlığın karesinin 20'de biri ((Ağırlık^2)/20) olacaktır. (tam sayı bölmesi yapın, yani küsürat olmayacak.) Modül, basla sinyali geldiğinde işleme başlar ve bir saat darbesinde kendi işlemini bitirir.

Devrenin girişleri:

saat: 1 bitlik saat sinyali

reset: 1 bitlik reset sinyali (senkron)

basla: 1 bitlik devrede işlemi başlatan giriş sinyali

agirlik: 6 bitlik ağırlık girişi

Devrenin çıkışları:

ucret: 8 bitlik ücret çıkışı

bitti: 1 bitlik devrenin çalışmasını bitirdiğini belirten çıkış sinyali

Örnek:

Yolcu	1	2	3	4
Ağırlık	14	40	45	32
Toplam Yük	14	14+40=54	54+45=99	99+32=131
Ücret	45	45	101	51

3. **"odeme" modülü:**

Giriş olarak her saat darbesinde ödenecek ücret ve bakiye değerlerini alarak, bakiyenin ücrete eşit ya da büyük olduğu durumlarda onay çıkışı olarak "1", bakiyenin ücretten küçük olduğu durumda ise onay çıkışı olarak "0" üretir. Eğer onaylandıysa bakiyeden ücreti çıkartır ve kalan bakiye çıkışı olarak verir, yoksa kalan bakiye çıkışı olarak bakiyeyi verir. Modül, basla sinyali geldiğinde işleme başlar ve bir saat darbesinde kendi işlemini bitirir.

Devrenin girişleri:

saat: 1 bitlik saat sinyali
reset: 1 bitlik reset sinyali (senkron)
basla: 1 bitlik devrede işlemi başlatan giriş sinyali
ucret: 8 bitlik ücret girişi
bakiye: 9 bitlik bakiye girişi

Devrenin çıkışları:

onay: 1 bitlik onay çıkışı
k_bakiye: 9 bitlik kalan bakiye çıkışı
bitti: 1 bitlik devrenin çalışmasını bitirdiğini belirten çıkış sinyali

4. **"ucak" modülü:**

Her saat darbesinde giriş olarak 1 bitlik onaylanan yolcu ve geçerli kimlik değerini alır. 50 adet yolcu onaylanırsa (hem o_yolcu=1 hem g_kimlik=1 ise), uçak kalkar. Uçağın kalktığı durumda kalkis çıkışı "1", kalkmadığı durumda ise "0" olmalıdır. Uçak kalktıktan sonra da bu modül her saat darbesinde kalkis çıkışı olarak "1" üretmelidir. Modül, basla sinyali geldiğinde işleme başlar ve bir saat darbesinde kendi işlemini bitirir.

Devrenin girişleri:

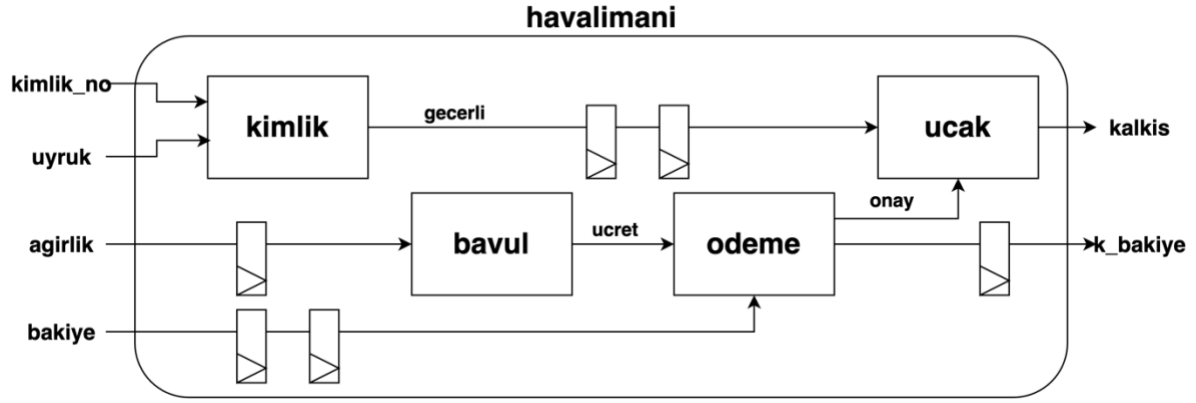
saat: 1 bitlik saat sinyali
reset: 1 bitlik reset sinyali (senkron)
basla: 1 bitlik devrede işlemi başlatan giriş sinyali
o_yolcu: 1 bitlik onaylanan yolcu girişi
g_kimlik: 1 bitlik geçerli kimlik girişi

Devrenin çıkışları:

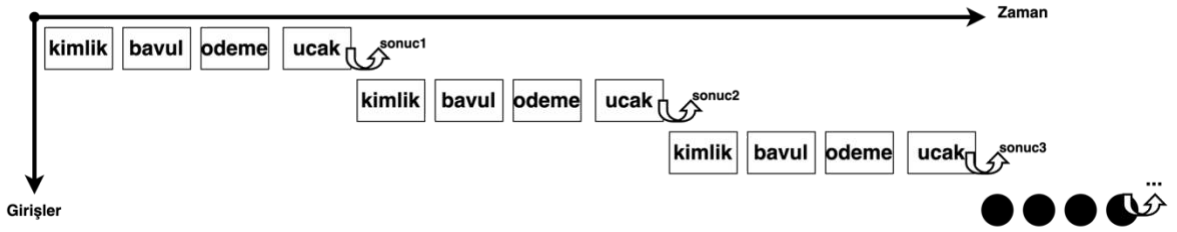
kalkis: 1 bitlik kalktı/kalkmadı çıkışı
bitti: 1 bitlik devrenin çalışmasını bitirdiğini belirten çıkış sinyali

5. **"havalimani" modülü:**

Aşağıda basitleştirilmiş şekilde görüleceği üzere, havalimani üst seviye bir modül olacak ve yazdığınız ilk 4 modülü kullanacaktır. Modüller; kimlik→bavul→odeme→ucak sırasıyla çalışacaktır ve **her saat vuruşunda sadece 1 tane** modül çalışmalıdır. Örneğin; ilk olarak kimlik modülü çalışacak, diğer tüm modüller bekleyecek, ikinci olarak bavul çalışacak diğer tüm modüller bekleyecek, … şeklinde. ucak modülünden sonra ise kimlik modülünden tekrar başlayarak çalışmaya bu döngüde devam edecektir. Bu şekilde, **her 4 çevrimde bir** girişlere uygun kalkis ve kalan bakiye çıkışları üretilecektir. Ayrıca, şekilde de göreceğiniz üzere, bu modülde aynı çevrimde gelen sinyallerin senkronizasyonu için; agirlik girişini bavul modülüne vermeden önce 1 çevrim, bakiye girişini odeme modülüne vermeden önce 2 çevrim, kimlik modülünün gecerli çıkışını ucak modülüne vermeden önce 2 çevrim, odeme modülünün k_bakiye çıkışını havalimani modülünün k_bakiye çıkışına vermeden önce 1 çevrim bekletmeniz gerekmektedir. reset sinyali geldiğinde, saat ile senkron şekilde, alt modüller de dahil bütün devrenin durumunu sıfırlamanız gerekmektedir.



Alt modüller zamana bağlı olarak aşağıda görüldüğü gibi çalışmalıdır. (0. çevrimde gelen girişin sonucu sonuc1→4. çevrimde, 4. çevrimde gelen girişin sonucu sonuc2→8.çevrimde, 8. çevrimde gelen girişin sonucu sonuc3→12. çevrimde çıkmalı.)



Devrenin parametre girişi:

BIT: kimlik numarası bit sayısını belirleyen parametre girişi (varsayılan değer: 6) (kimlik modülünün parametre girişine bağlanmalı)

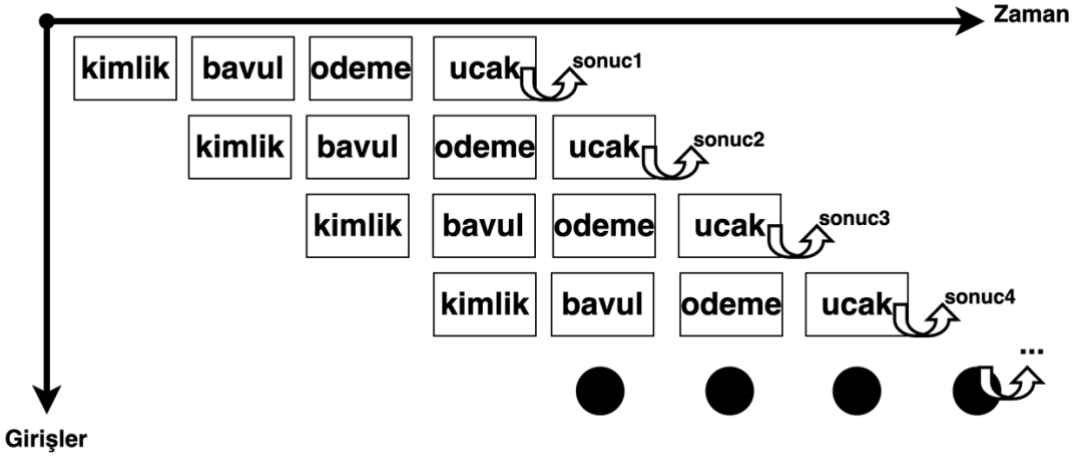
Devrenin girişleri:

saat: 1 bitlik saat sinyali
reset: 1 bitlik reset sinyali (senkron)
kimlik_no: BIT bitlik kimlik numarası
uyruk: 1 bitlik uyruk bilgisi
agirlik: 6 bitlik ağırlık girişi
bakiye: 9 bitlik bakiye girişi

Devrenin çıkışları:

kalkis: 1 bitlik kalktı/kalkmadı çıkışı
k_bakiye: 9 bitlik kalan bakiye çıkışı

[20 Puan] b) Bu kısımda a şıkkında yazdığınız ilk 4 modülü kullanarak havalimani modülünü **boru hattı** yöntemi kullanarak gerçekleyeceksiniz. Oluşturacağınız modüle "havalimani_boru"(oluşacak dosya "havalimani_boru.v") ismini verin. 4 aşamalı boru hattı; kimlik→bavul→odeme→ucak şeklinde çalışacaktır ve her saat vuruşunda bu farklı modüller **aynı anda** çalışabilmelidir. (Farklı girişlerin işlemlerini aynı anda yapabilmeliler.) **İlk 4 çevrimden sonra** sürekli olarak girişlere uygun kalkis ve kalan bakiye çıkışları üretilecektir. Alt modüller zamana bağlı olarak aşağıda görüldüğü gibi çalışmalıdır. (0. çevrimde gelen girişin sonucu sonuc1→4. çevrimde, 1. çevrimde gelen girişin sonucu sonuc2→5.çevrimde, 2. çevrimde gelen girişin sonucu sonuc3→6. çevrimde, 3. çevrimde gelen girişin sonucu sonuc4→7. çevrimde çıkmalı.)



havalimani_boru modülünün giriş ve çıkışları havalimani modülü ile birebir aynıdır. Aynı çevrimde gelen sinyallerin senkronizasyonu havalimanı modülünde olduğu gibi bu modülde de sağlanmalıdır.

Not: Parametrik istenilen modülleri parametrik yapamazsanız, BIT diye bir parametreyi hiç tanımlamayın ve kimlik_no girişlerinin bit sayısını 6 olarak ayarlayın. Buradan bir miktar puan kaybınız olacaktır ama modülünüzün çalışması çalışmamasından iyidir.