ADM Kontrol Devresinin Gerçeklenmesi

BIL-204: Lojik Devreler II

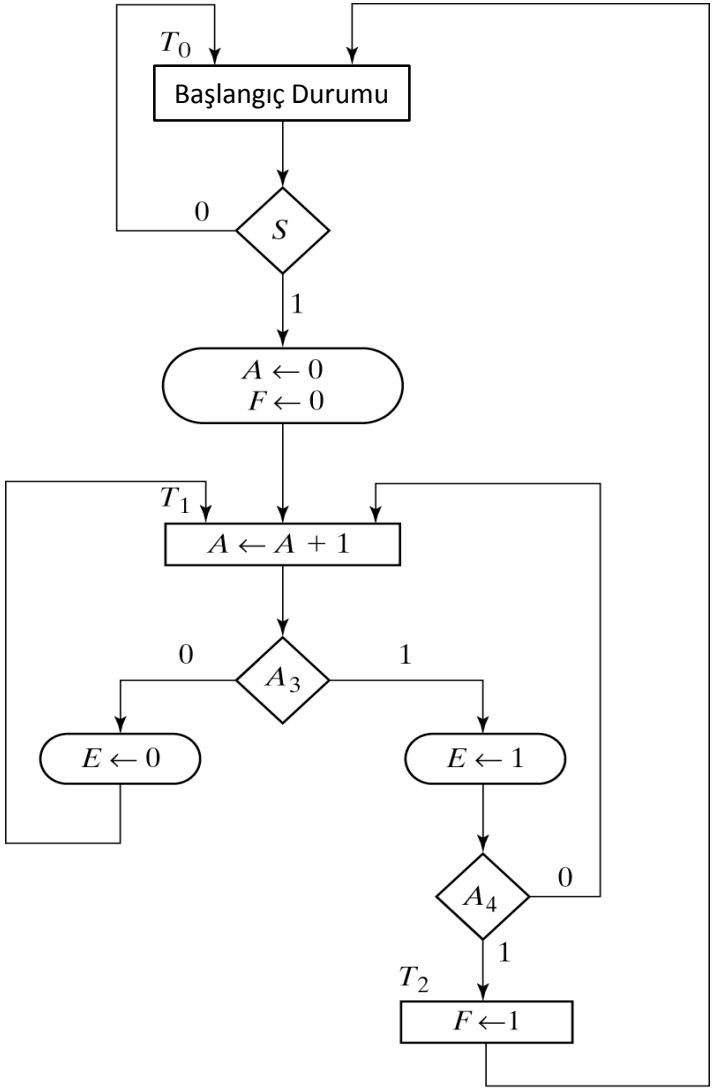
Dersi veren öğretim üyesi:
Yrd. Doç. Dr. Fatih Gökçe

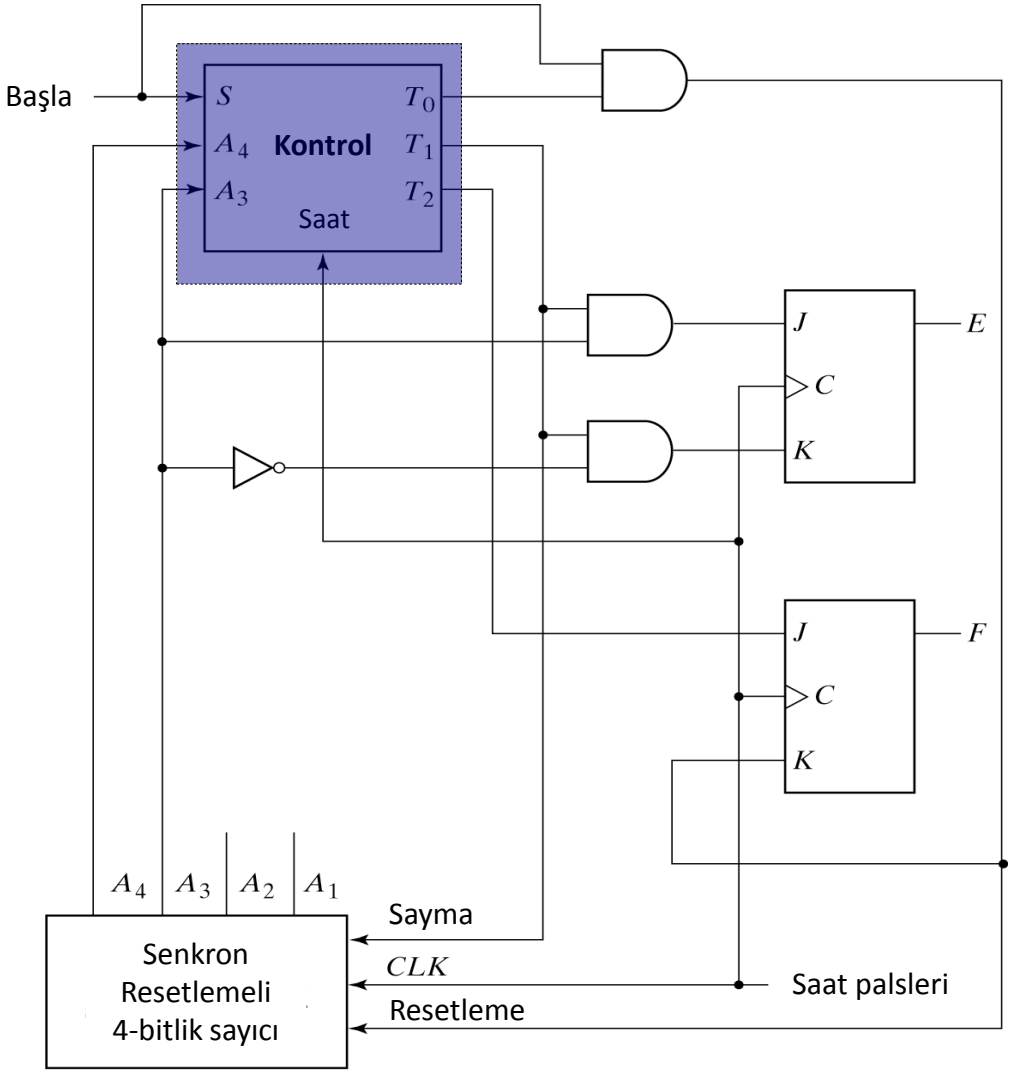
Süleyman Demirel Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Bu ders notu “M. Morris Mano, Digital Design, 2nd Edition” kitabının çevirisi olan “Seta Boğosyan, Metin Gökaşan, Salman Kurtulan, Sayısal Tasarım, Literatür Yayıncılık” kitabından ve METU-CENG-232-Logic Design ders notlarından faydalanılarak hazırlanmıştır.

Tasarım Örneği için ADM Diyagramı

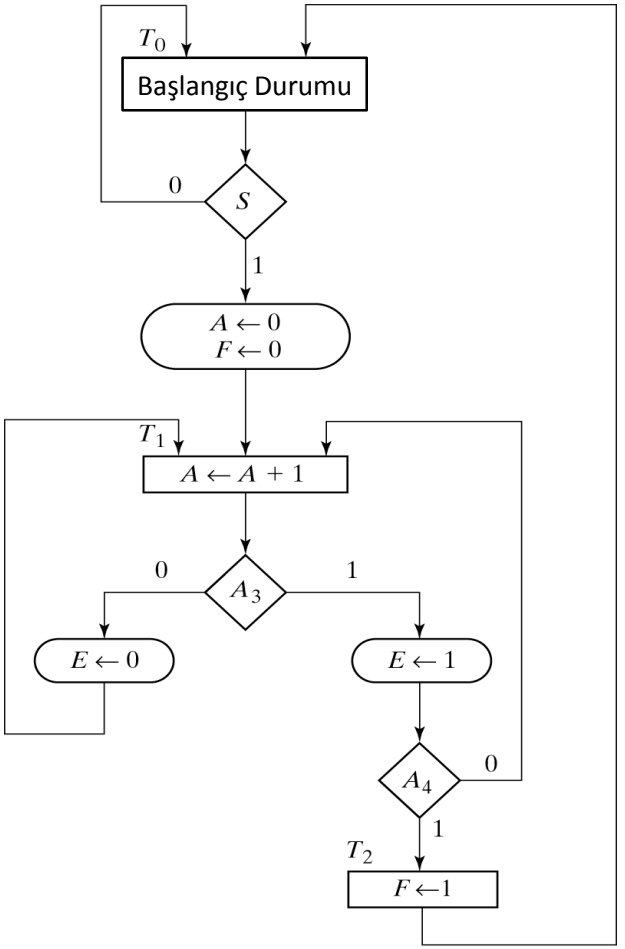
- Geçtiğimiz ders verdiğimiz sözel ifadelerden aşağıdaki ADM diyagramını elde etmiş ve veri işleyicisi kısmını tasarlamıştık. Bugün Kontrol Devresini farklı yaklaşımlarla gerçekleştireceğiz.





Durum Tablosu

Şimdiki Durum Sembolü	Şimdiki Durum		Girişler			Sonraki Durum		Çıkışlar		
	G_1	G_2	S	A_3	A_4	G_1	G_2	T_0	T_1	T_2
T_0	0	0	0	X	X	0	0	1	0	0
T_0	0	0	1	X	X	0	1	1	0	0
T_1	0	1	X	0	X	0	1	0	1	0
T_1	0	1	X	1	0	0	1	0	1	0
T_1	0	1	X	1	1	1	1	0	1	0
T_2	1	1	X	X	X	0	0	0	0	1



JK Flip-Floplu Lojik Diyagram

- Flip-Flop girişleri için uyarma tablosunun elde edilmesi ve ardışıl devrenin kombinasyonel devre kısmının basitleştirilmesi gerekir.
- Şimdiki durum ve Girişler toplam 5 sütün olduğu için 5 değişkenli diyagram kullanarak sadeleştirme gerekecektir.

Şimdiki Durum Sembolü	Şimdiki Durum		Girişler			Sonraki Durum		Çıkışlar		
	G_1	G_2	S	A_3	A_4	G_1	G_2	T_0	T_1	T_2
T_0	0	0	0	X	X	0	0	1	0	0
T_0	0	0	1	X	X	0	1	1	0	0
T_1	0	1	X	0	X	0	1	0	1	0
T_1	0	1	X	1	0	0	1	0	1	0
T_1	0	1	X	1	1	1	1	0	1	0
T_2	1	1	X	X	X	0	0	0	0	1

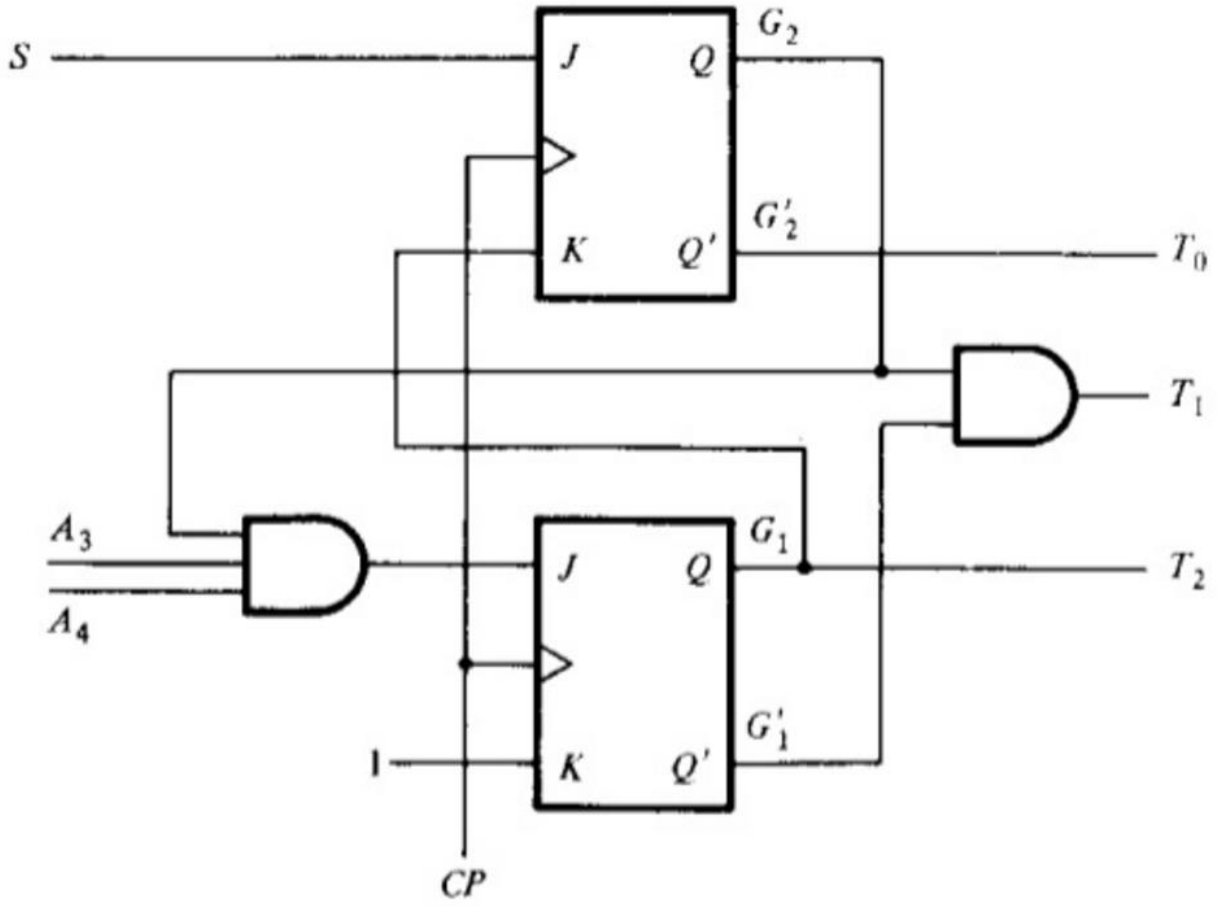
$$JG_1 = G_2A_3A_4$$
$$KG_1 = 1$$

$$JG_2 = S$$
$$KG_2 = G_1$$

$$T_0 = G_2'$$
$$T_1 = G_1'G_2$$
$$T_2 = G_1$$

JK Flip-Floplu Lojik Diyagram

- Flip-Flop girişleri için uyarma tablosunun elde edilmesi ve ardışıl devrenin kombinasyonel devre kısmının basitleştirilmesi gerekir.
- Şimdiki durum ve Girişler toplam 5 sütün olduğu için 5 değişkenli diyagram kullanarak sadeleştirme gerekecektir.



$$JG_1 = G_2A_3A_4$$
$$KG_1 = 1$$

$$JG_2 = S$$
$$KG_2 = G_1$$

$$T_0 = G_2'$$
$$T_1 = G_1'G_2$$
$$T_2 = G_1$$

D Flip-Flopları ve Kod Çözücü

- D Flip-flop kullanılması durumunda uyarma tablosuna gerek olmaz; sonraki durum doğrudan flip-flop girişleri olarak kullanılabilir.

Şimdiki Durum Sembolü	Şimdiki Durum		Girişler			Sonraki Durum		Çıkışlar		
	G_1	G_2	S	A_3	A_4	G_1	G_2	T_0	T_1	T_2
T_0	0	0	0	X	X	0	0	1	0	0
T_0	0	0	1	X	X	0	1	1	0	0
T_1	0	1	X	0	X	0	1	0	1	0
T_1	0	1	X	1	0	0	1	0	1	0
T_1	0	1	X	1	1	1	1	0	1	0
T_2	1	1	X	X	X	0	0	0	0	1

$$DG_1 = G_1' G_2 A_3 A_4$$
$$DG_2 = G_1' G_2' S + G_1' G_2$$

D Flip-Flopları ve Kod Çözücü

- T_0, T_1 ve T_2 çıkışlarını elde etmek için flip-flop çıkışlarına bir kod çözücü bağlanacaktır. Bu durumda flip-flop çıkışlarını mevcut durum koşulları şeklinde kullanmak yerine, kod çözücü çıkışlarını kullanabiliriz.

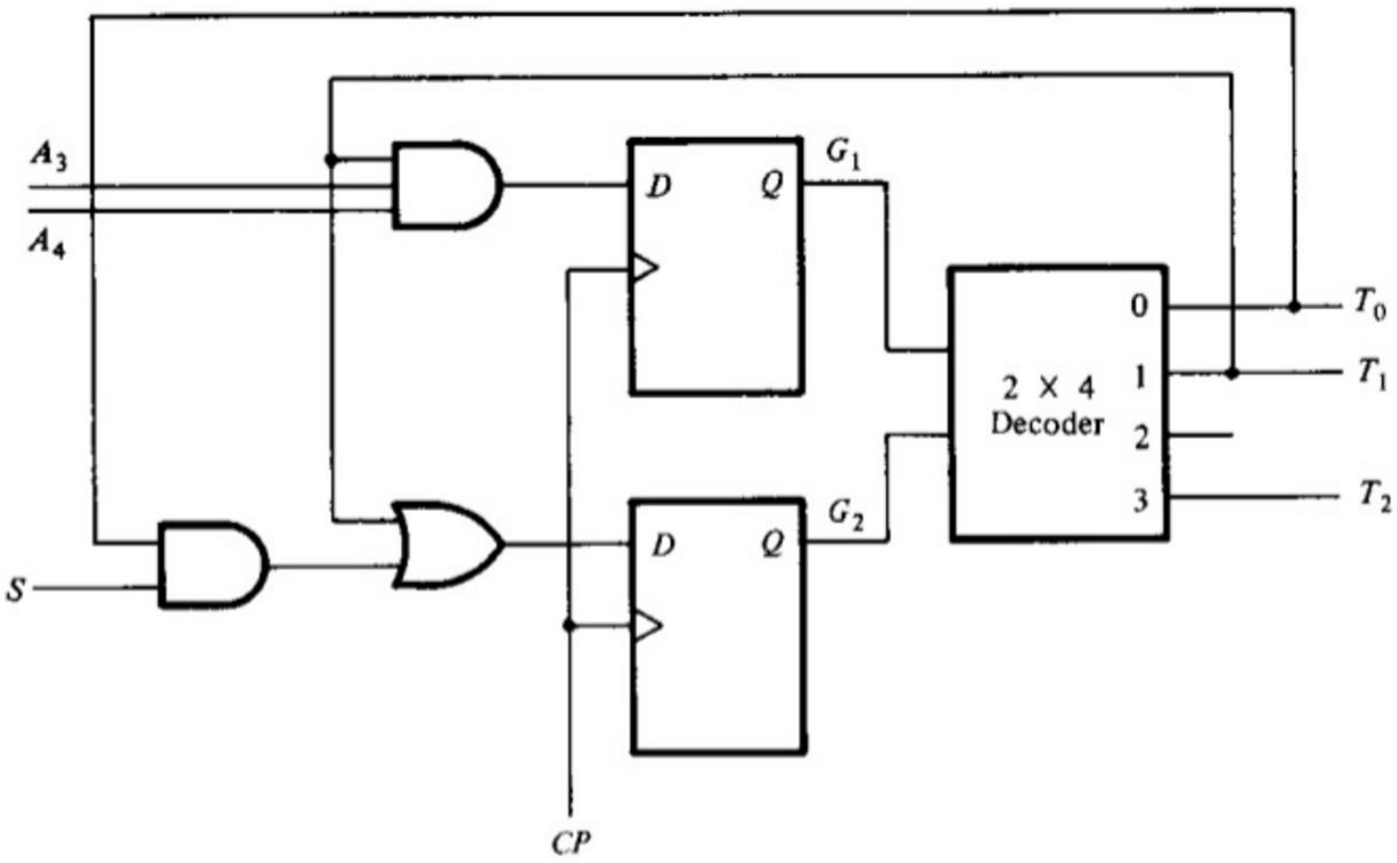
Şimdiki Durum Sembolü	Şimdiki Durum		Girişler			Sonraki Durum		Çıkışlar		
	G_1	G_2	S	A_3	A_4	G_1	G_2	T_0	T_1	T_2
T_0	0	0	0	X	X	0	0	1	0	0
T_0	0	0	1	X	X	0	1	1	0	0
T_1	0	1	X	0	X	0	1	0	1	0
T_1	0	1	X	1	0	0	1	0	1	0
T_1	0	1	X	1	1	1	1	0	1	0
T_2	1	1	X	X	X	0	0	0	0	1

$$DG_1 = G_1' G_2 A_3 A_4$$
$$DG_2 = G_1' G_2' S + G_1' G_2$$

$$DG_1 = T_1 A_3 A_4$$
$$DG_2 = T_0 S + T_1$$

D Flip-Flopları ve Kod Çözücü (Decoder)

- T_0, T_1 ve T_2 çıkışlarını elde etmek için flip-flop çıkışlarına bir kod çözücü bağlanacaktır. Bu durumda flip-flop çıkışlarını mevcut durum koşulları şeklinde kullanmak yerine, kod çözücü çıkışlarını kullanabiliriz.

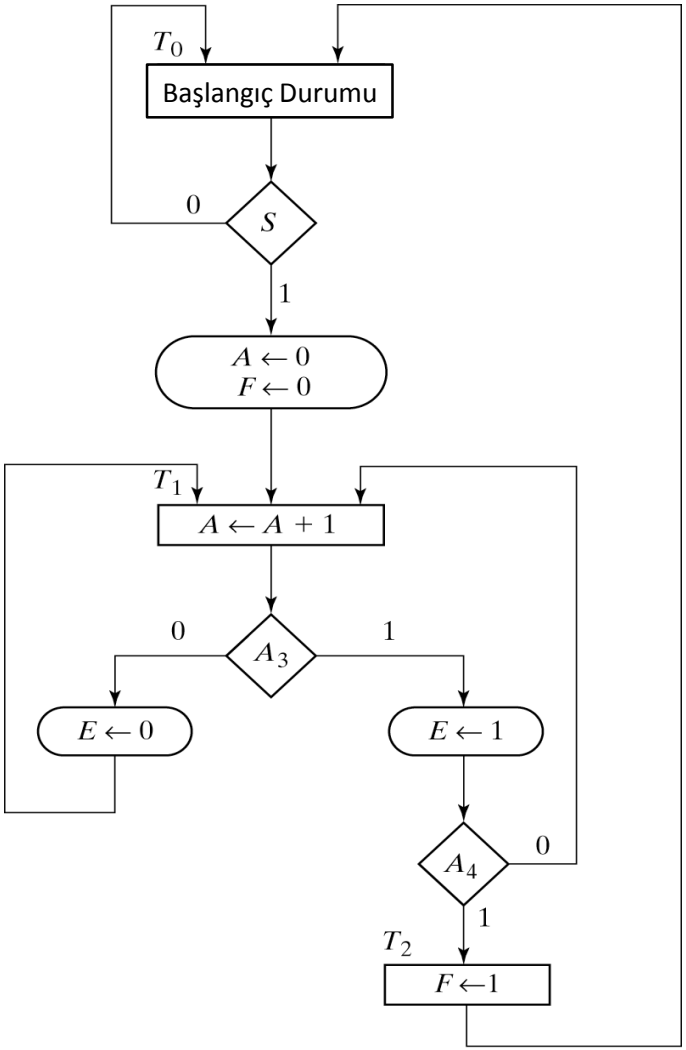


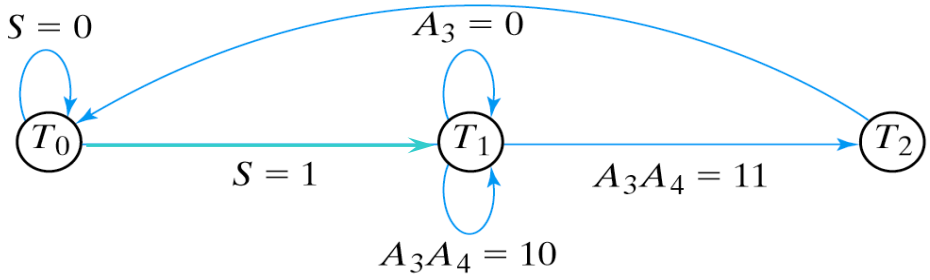
$$DG_1 = T_1 A_3 A_4$$

$$DG_2 = T_0 S + T_1$$

Durum Başına Bir Flip-Flop

- Durum veya uyarma tablosuna gerek duymadan, doğrudan durum diyagramından yararlanılır.

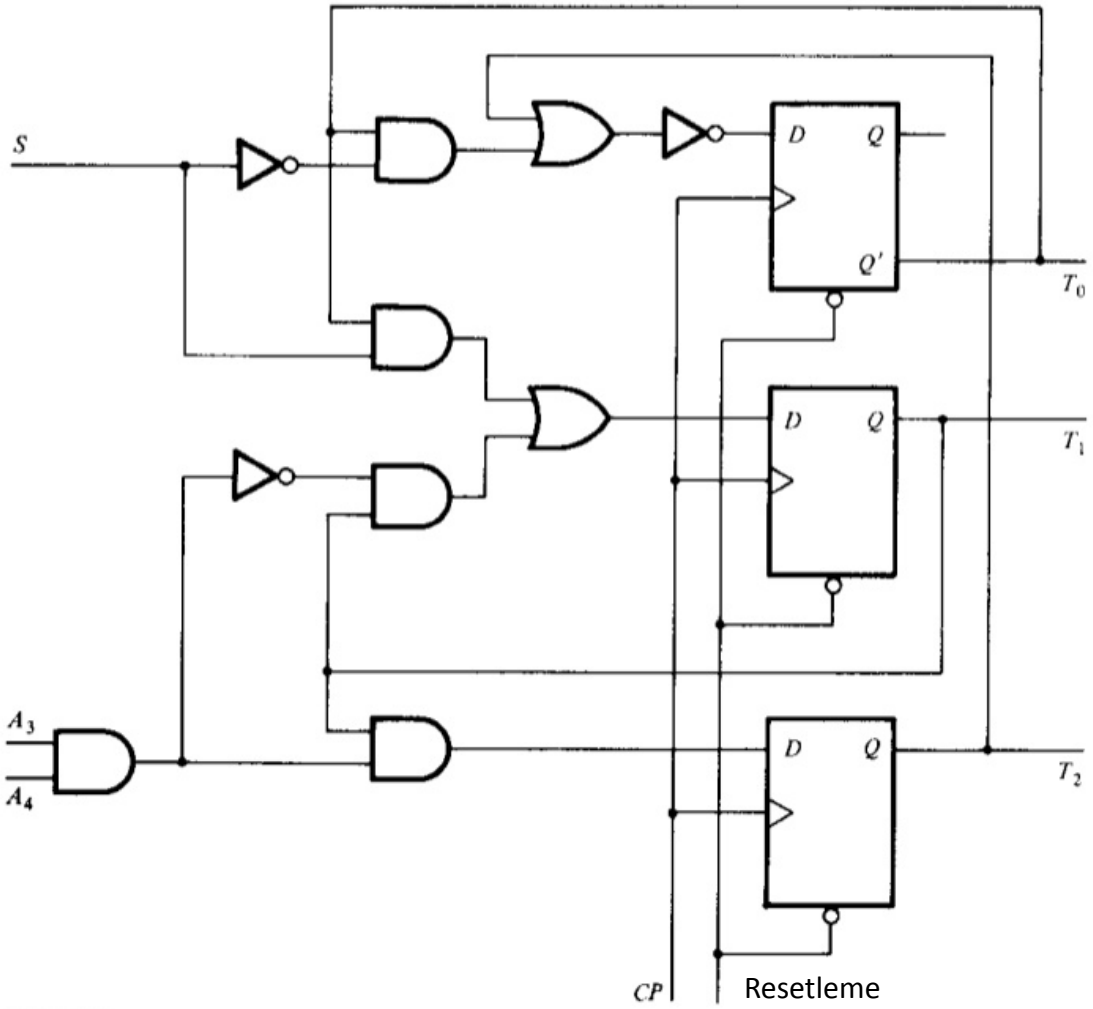




$$DT_0 = T_2 + S'T_0$$
$$DT_1 = ST_0 + A_3'T_1 + A_3A_4'T_1$$
$$= ST_0 + (A_3A_4)'T_1$$
$$DT_2 = A_3A_4T_1$$

Durum Başına Bir Flip-Flop

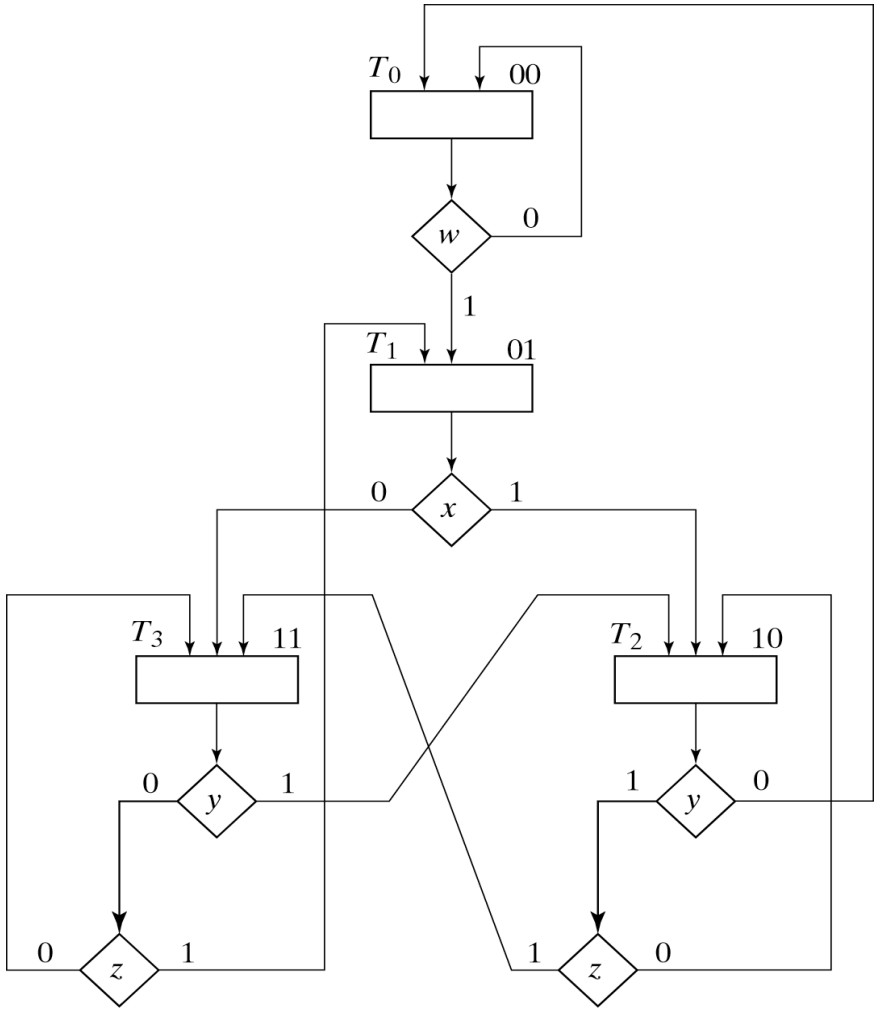
- Durum veya uyarma tablosuna gerek duymadan, doğrudan durum diyagramından yararlanılır.



$$DT_0 = T_2 + S'T_0$$
$$DT_1 = ST_0 + A_3'T_1 + A_3A_4'T_1$$
$$= ST_0 + (A_3A_4)'T_1$$
$$DT_2 = A_3A_4T_1$$

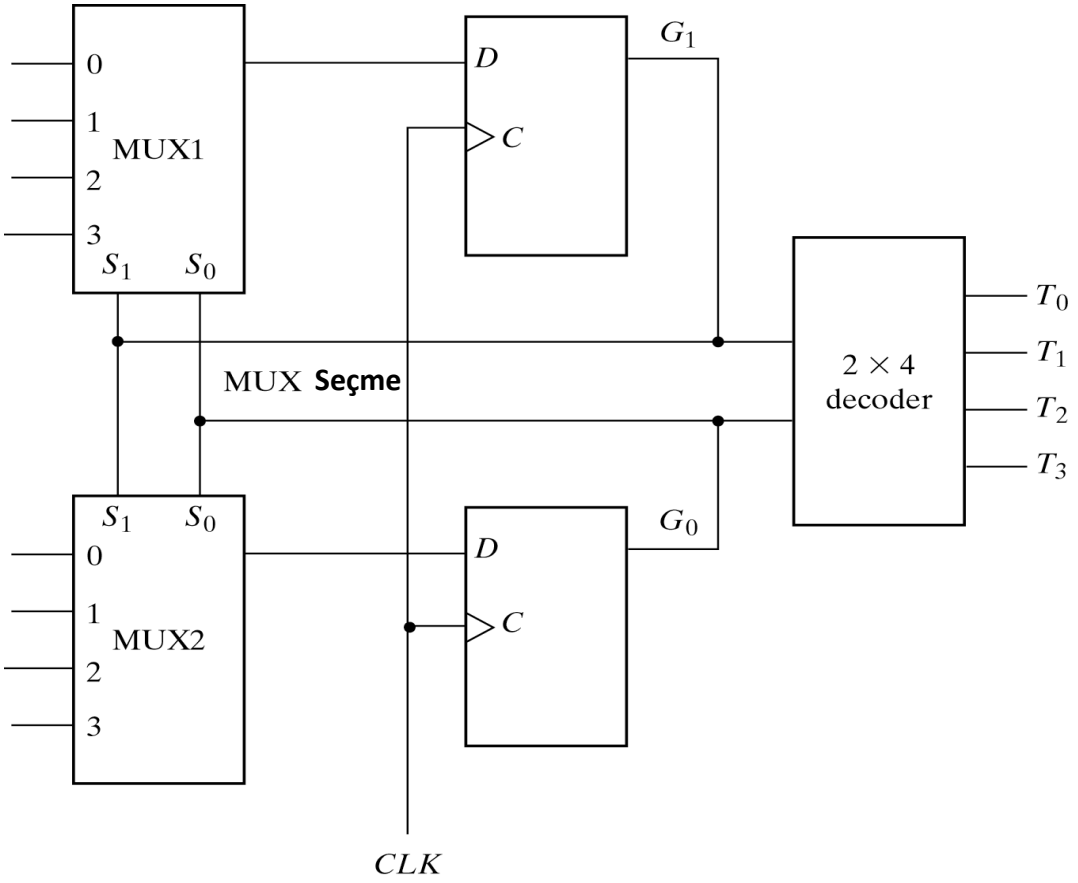
Veri Seçicilerle (Multiplexer, MUX) Tasarım

- Bu kısımda sadece kontrol devresiyle ilgileneceğimiz için durum kutuları boş bırakılmıştır.
- 4 kontrol girişi: w, x, y, z
- 4 Durum: T_0-T_3 →2 flip-flop.



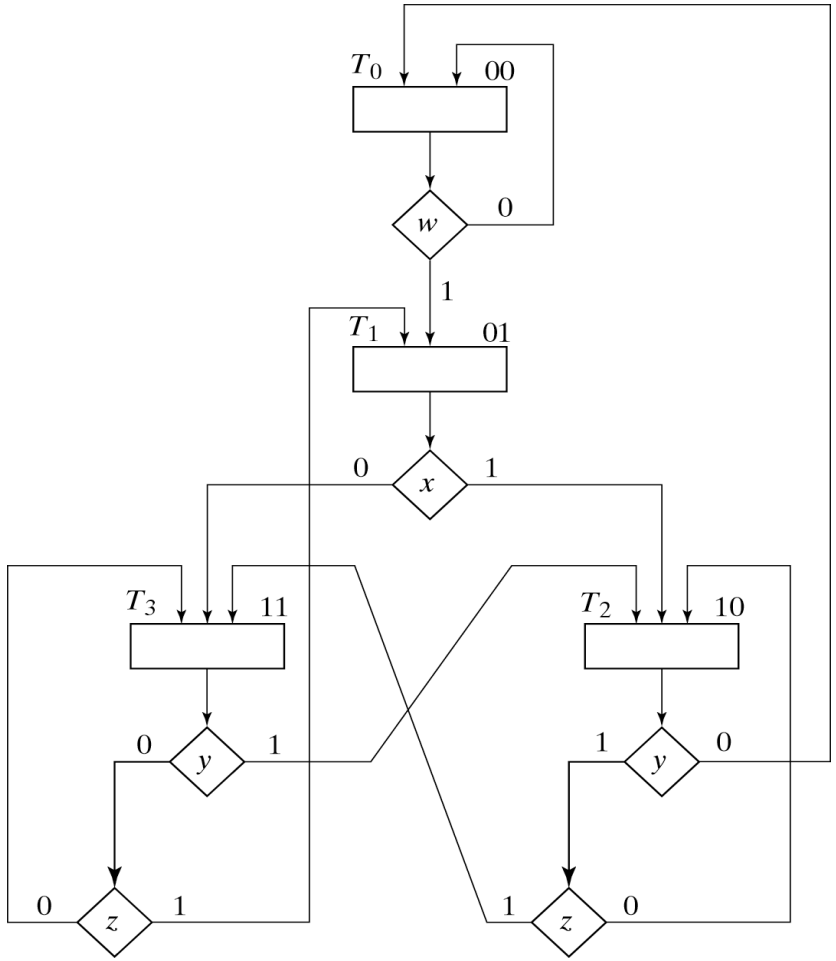
Veri Seçicilerle (Multiplexer, MUX) Tasarım

- 2 D flip-flop durumu kodlar.
- Durum, kod çözücü ile çözülerek T_0-T_3 elde edilir.
- Şimdiki durum bir sonraki durumu seçmeyi sağlar.
- Peki, MUX girişlerini nasıl belirleyeceğiz?



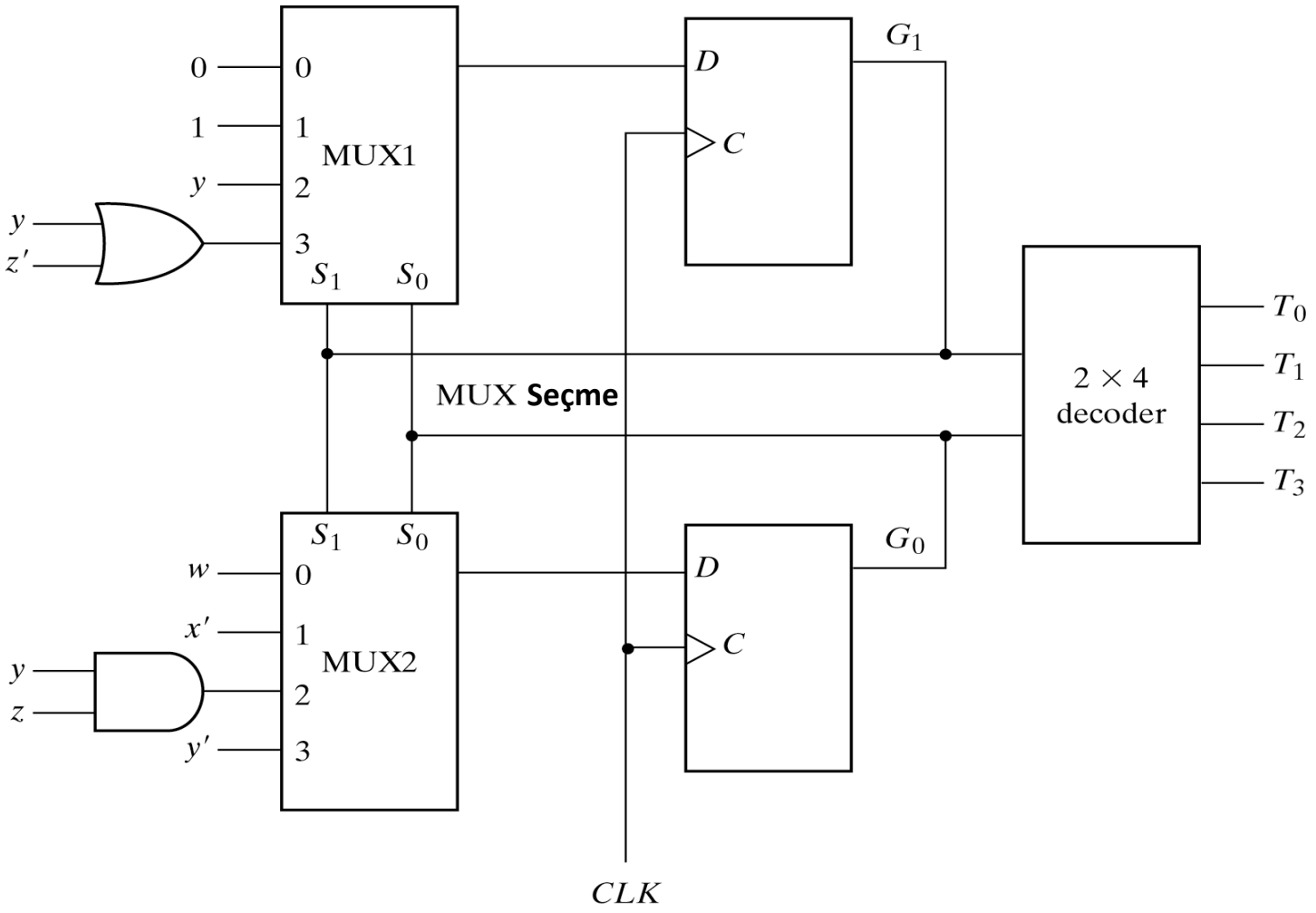
Veri Seçicilerle (Multiplexer, MUX) Tasarım

Şimdiki Durum		Sonraki Durum		Giriş Koşulları	Veri Seçici Girişleri	
G_1	G_2	G_1	G_2		MUX1	MUX2
0	0	0	0	w′	0	w
0	0	0	1	w		
0	1	1	0	x	1	x′
0	1	1	1	x′		
1	0	0	0	y′	yz′+yz = y	yz
1	0	1	0	yz′		
1	0	1	1	yz		
1	1	0	1	y′z	y+y′z′ = y+z′	y′z+y′z′ = y′
1	1	1	0	y		
1	1	1	1	y′z′		



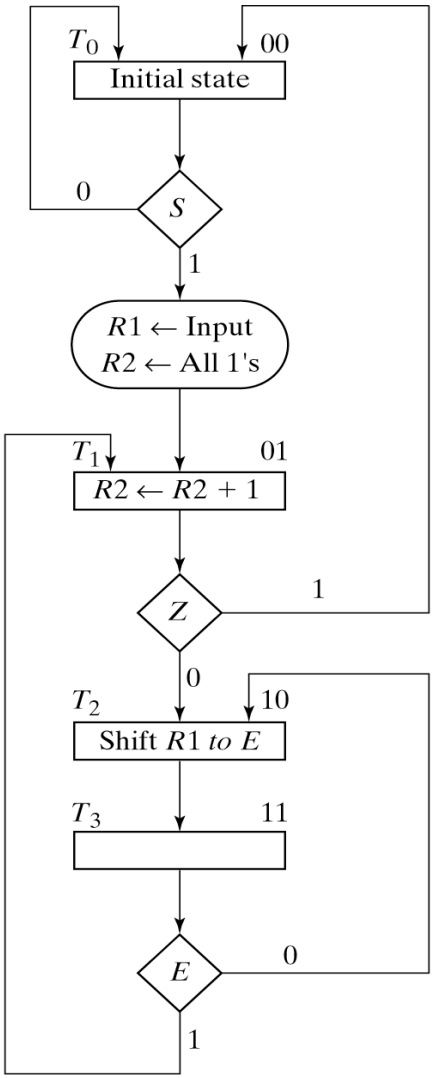
Veri Seçicilerle (Multiplexer, MUX) Tasarım

Veri Seçici Girişleri	
MUX1	MUX2
0	w
1	x′
yz′+yz = y	yz
y+y′z′ = y+z′	y′z+y′z′ = y′



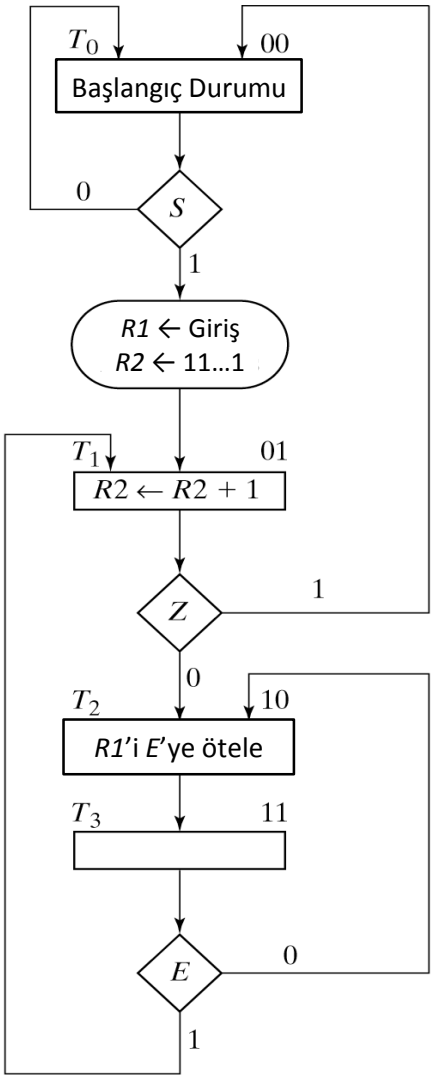
Tasarım Örneği: 1'lerin sayısı

- Sistem 2 register (R1 ve R2) ile bir flip-floptan (E) oluşur.
- Sistem R1'e yüklenen ikili sayıdaki 1'lerin sayısını sayar ve bu sayıyı R2'de tutar.
- R1'den E flip-flobuna 1 bit kaydır.
- Eğer E == 1 ise R2++
- Eğer Z == 1 (Yani R1 == 0 ise) ise dur.
- İlk durumda R2 tamamıyla 1'lere set edilmiştir. Niçin?

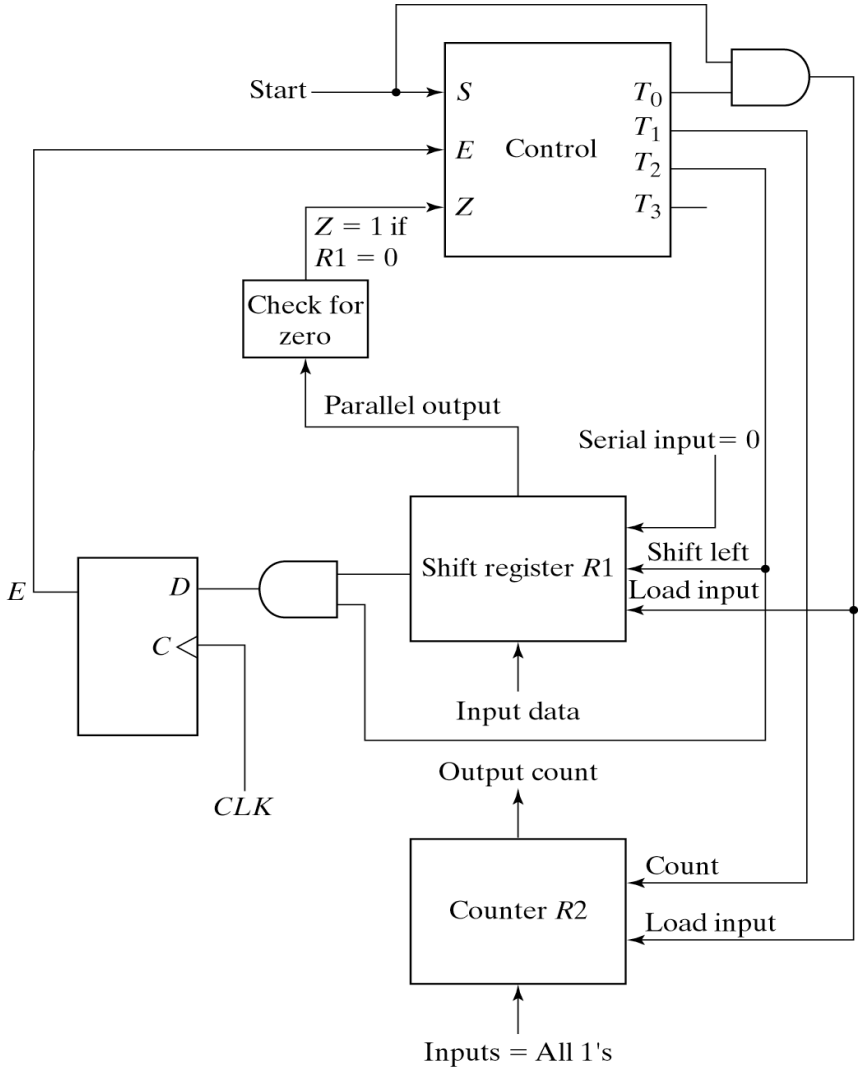


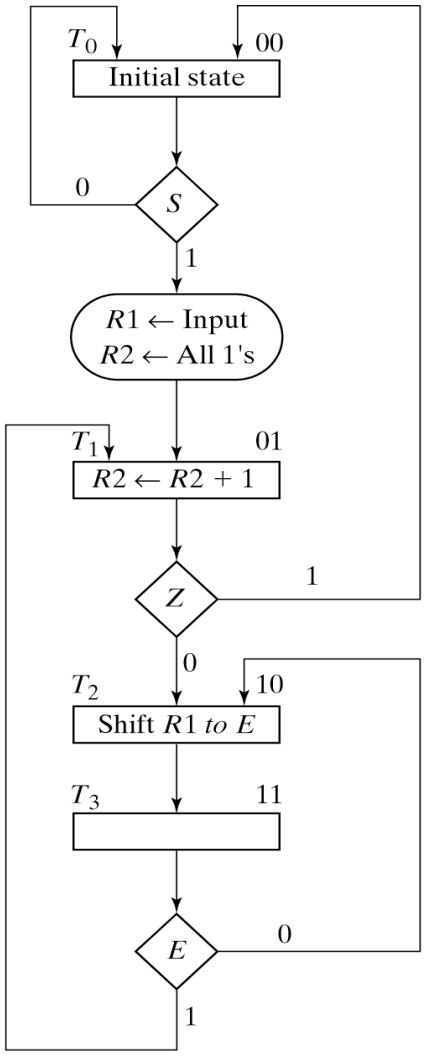
Tasarım Örneği: 1'lerin sayısı (Bu slaytta şekildeki ifadeler Türkçeleştirilmiştir.)

- Sistem 2 register (R1 ve R2) ile bir flip-floptan (E) oluşur.
- Sistem R1'e yüklenen ikili sayıdaki 1'lerin sayısını sayar ve bu sayıyı R2'de tutar.
- R1'den E flip-flobuna 1 bit kaydır.
- Eğer E == 1 ise R2++
- Eğer Z == 1 (Yani R1 == 0 ise) ise dur.
- İlk durumda R2 tamamıyla 1'lere set edilmiştir. Niçin?



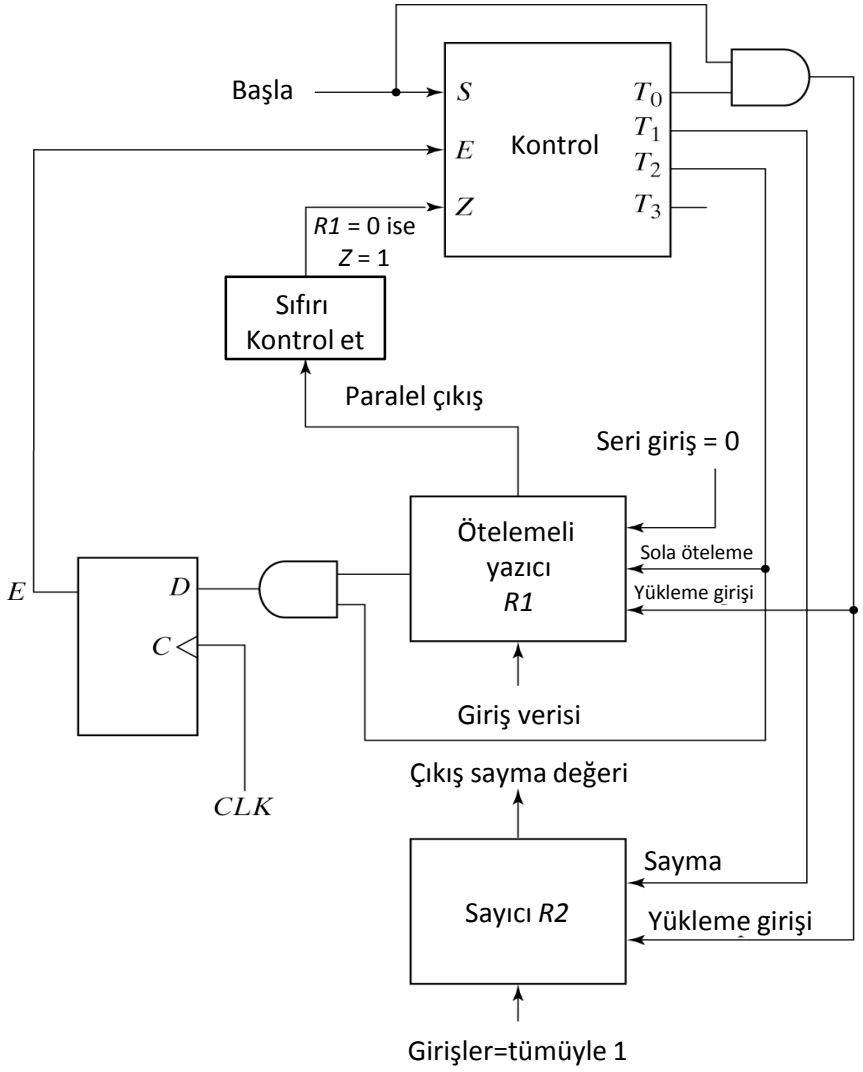
Tasarım Örneği: 1'lerin sayısı
Veri işleyici

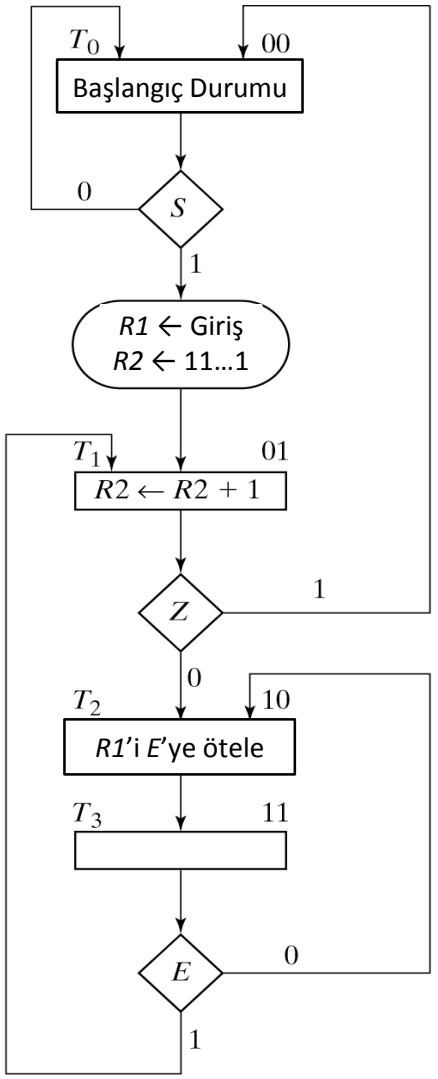


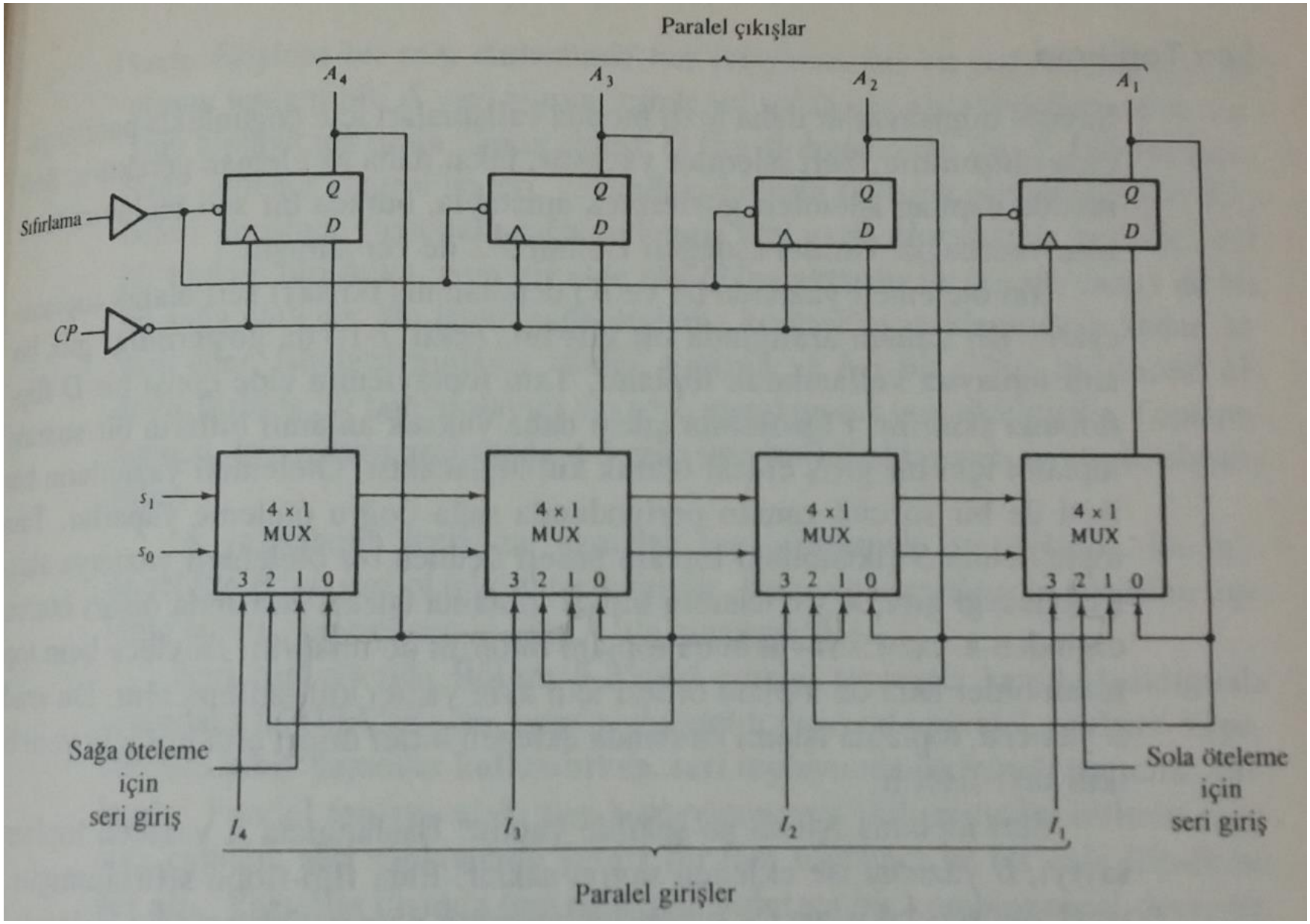


Tasarım Örneği: 1'lerin sayısı

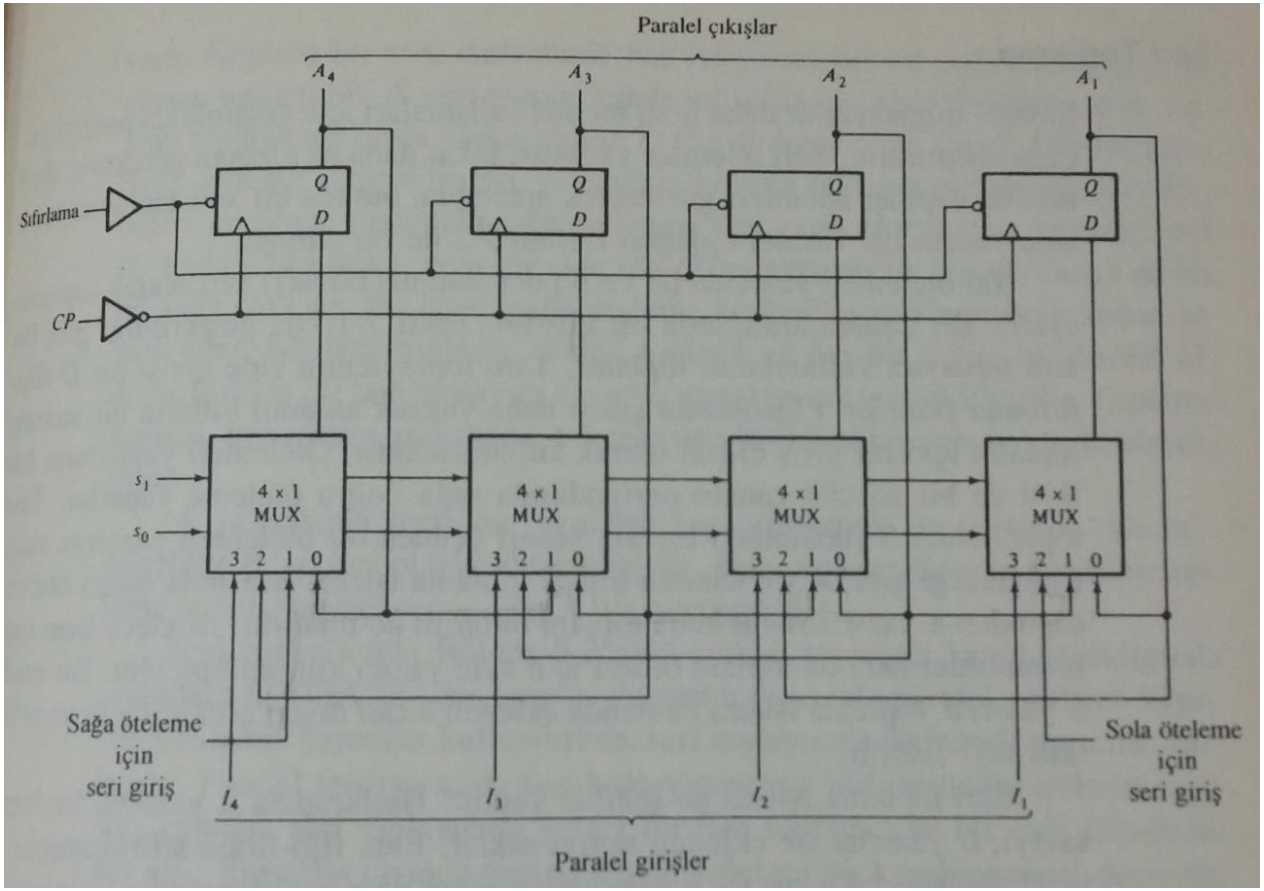
Veri işleyici (Bu slaytta şekillerdeki ifadeler Türkçeleştirilmiştir.)





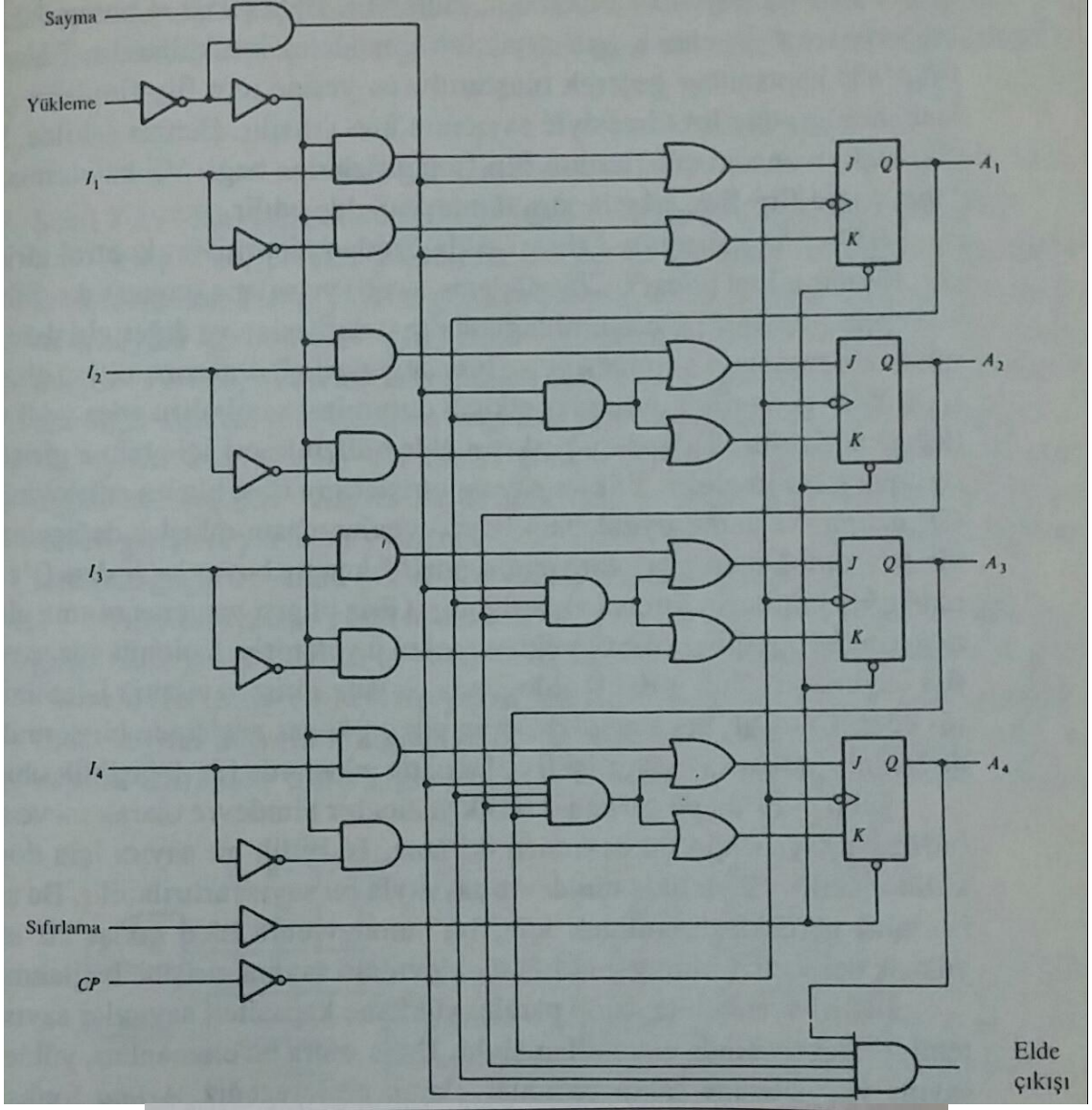


4 Bitlik Paralel Yüklemeli İki Yönlü Ötelemeli Kaydedici



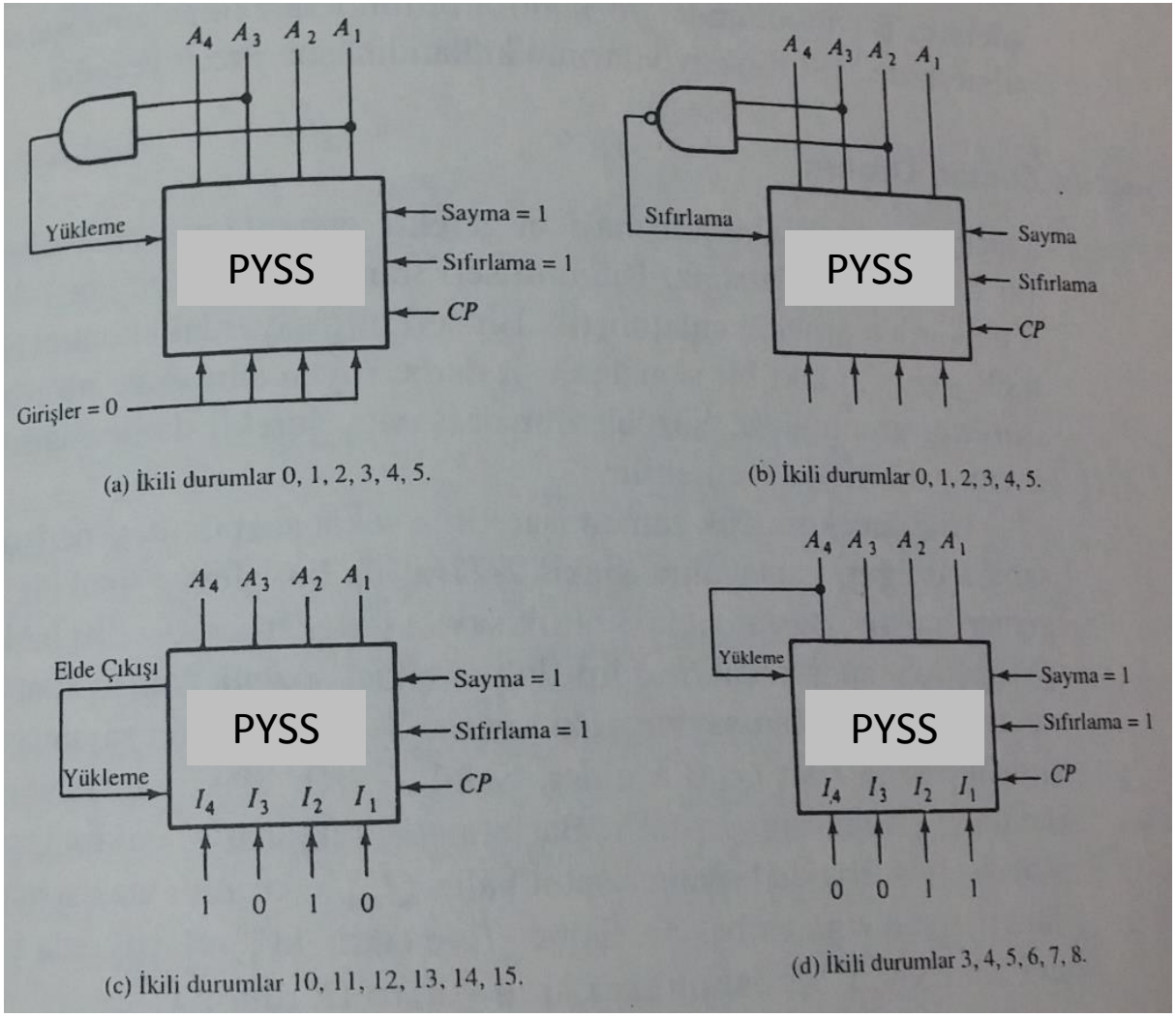
Mod Kontrol		
s_1	s_0	Yazıcının İşlevi
0	0	Değişim yok
0	1	Sağa öteleme
1	0	Sola öteleme
1	1	Paralel yükleme

4 Bitlik Paralel Yüklemeli İki Yönlü Ötelemeli Kaydedici



Sıfırlama	*CP*	Yükleme	Sayma	Fonksiyon
0	*X*	*X*	*X*	Sıfırlanır (0)
1	*X*	0	0	Değişim yok
1	↑	1	*X*	Girişler yüklenir
1	↑	0	1	Sonraki ikili durumu sayar

4 Bitlik Paralel Yüklemeli Senkron Sayıcı (PYSS)

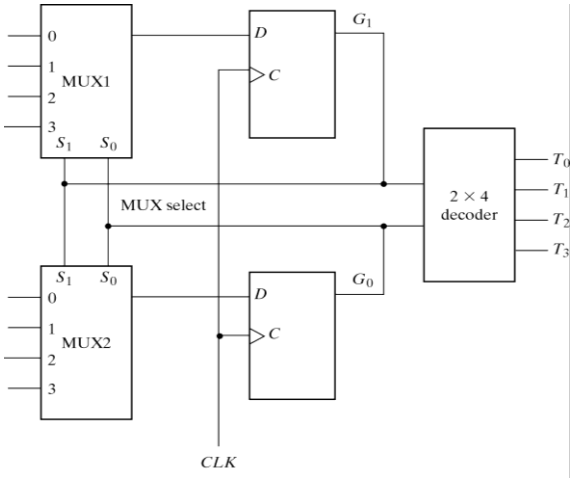


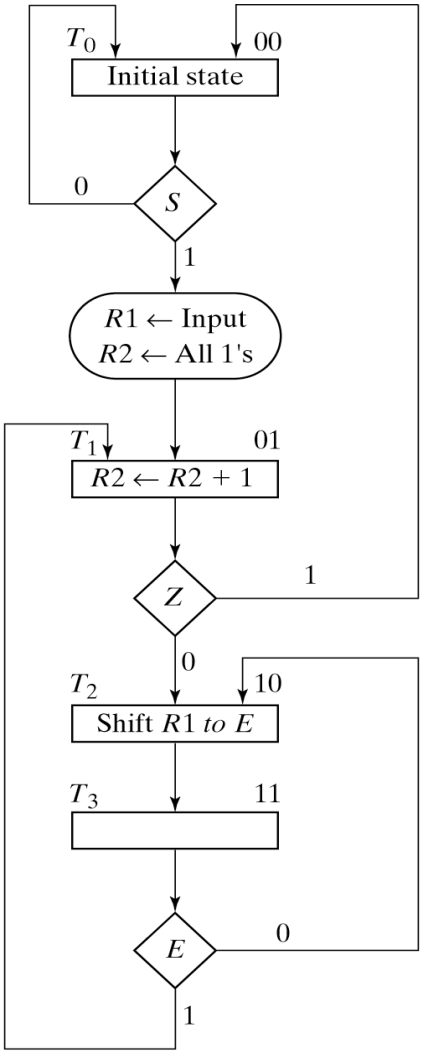
Sıfırlama	*CP*	Yükleme	Sayma	Fonksiyon
0	*X*	*X*	*X*	Sıfırlanır (0)
1	*X*	0	0	Değişim yok
1	↑	1	*X*	Girişler yüklenir
1	↑	0	1	Sonraki ikili durumu sayar

Paralel Yüklemeli Senkron Sayıcı (PYSS) kullanarak tasarımlar

Tasarım Örneği: 1'lerin sayısı Veri Seçici Girişleri

Şimdiki Durum		Sonraki durum		Giriş Koşulları	Veri Seçici Girişleri	
G1	G2	G1	G2		MUX1	MUX2
0	0	0	0	S'	0	S
0	0	0	1	S		
0	1	0	0	Z	Z'	0
0	1	1	0	Z'		
1	0	1	1	Yok	1	1
1	1	0	1	E	E'	E
1	1	1	0	E'		





Tasarım Örneği: 1'lerin sayısı
Veri Seçici ile Kontrol Lojiği Tasarımı