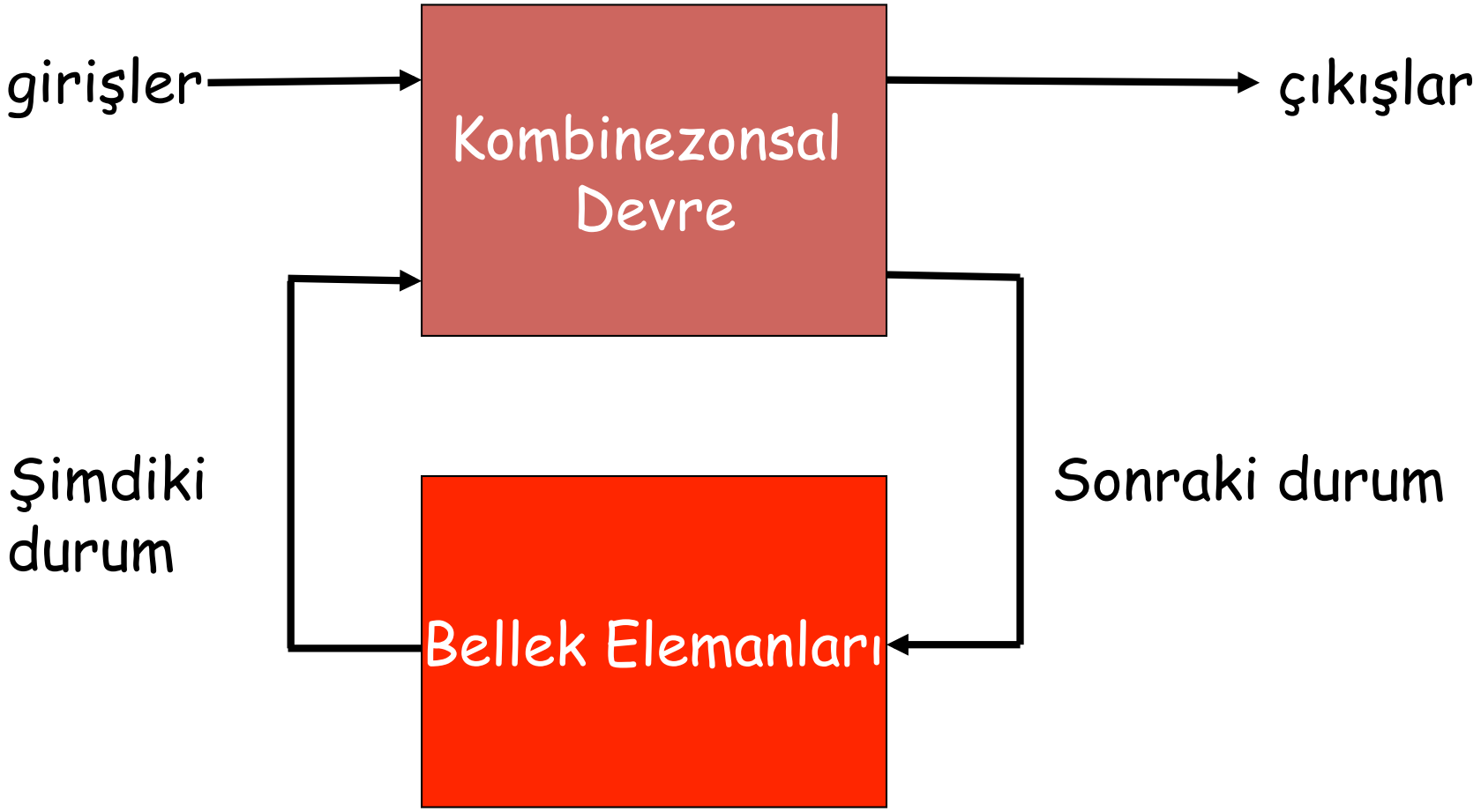
Ardışıl Devre Modeli



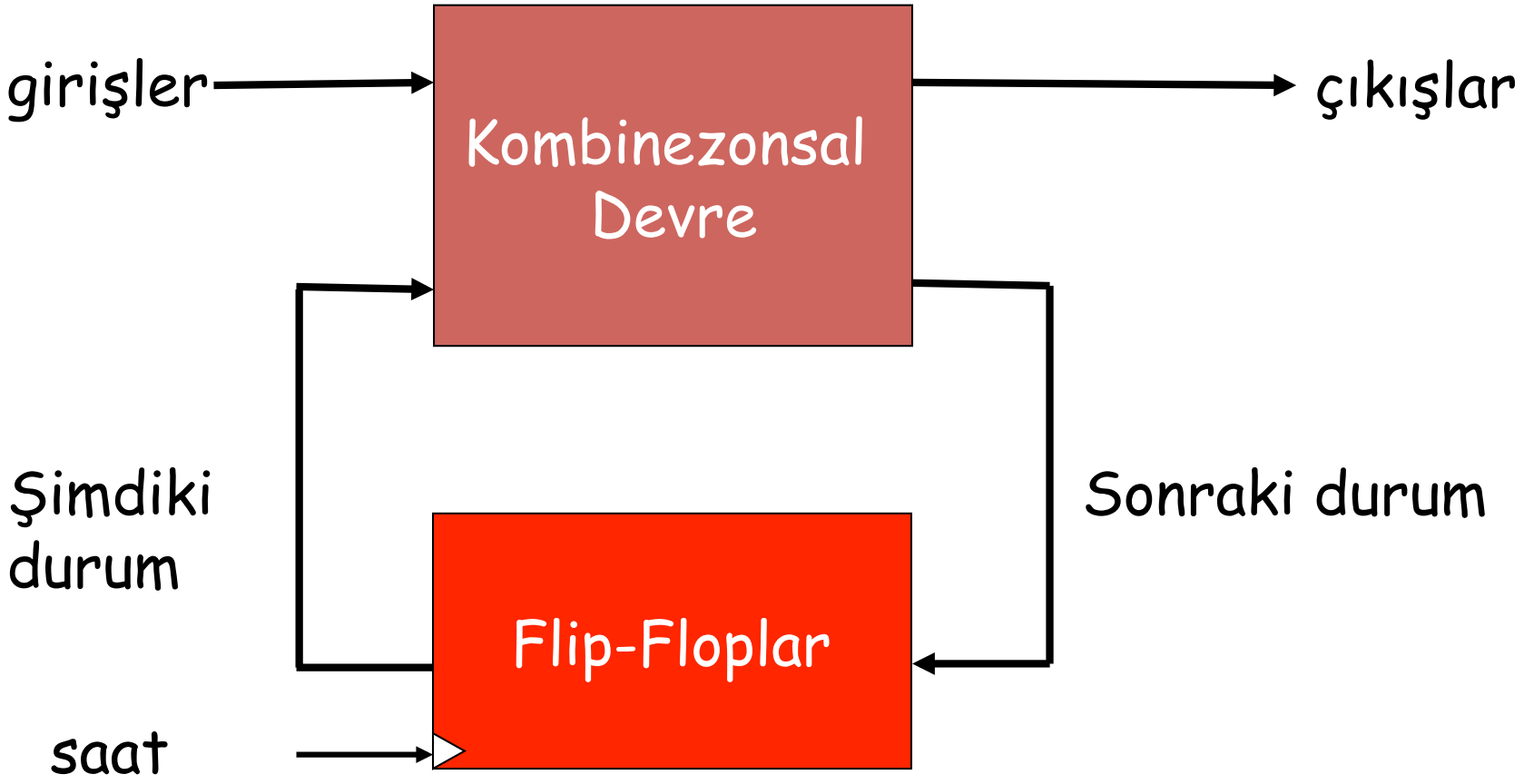
Şimdiki durum daha önceki girişlere bağlıdır.

Senkron Ardışıl Devreler

- İşaretler bellek elemanlarını zamanın ayrık anlarında etkilerler.
- Ayrık anlar senkronizasyon gerektirir.
- Senkronizasyon ortak bir saat ile sağlanır.
- “Saat üreteci” periyodik darbe dizisi üreten bir devredir.
- Bellek elemanlarının durumu her saat darbesinde güncellenir.

Senkron Ardışıl Devreler

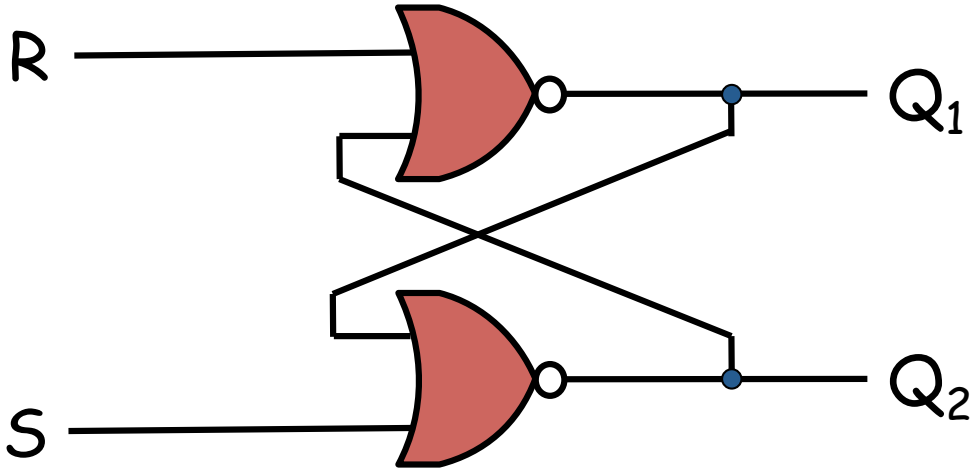
- Bellek elemanları 1-bitlik bilgi saklayabilen flip-flop lardır.



Latch ler

- Temel bellek elemanları
- Bir latch ikili durumunu sonsuza kadar koruyabilen bir bellek elemanıdır.
- Latch ler asenkron devrelerdir ve çalışmak için saat işaretine ihtiyaçları yoktur.
- Bu sebeple senkron ardışıl devrelerde doğrudan kullanılmazlar.
- Flip-flop ları elde etmek için kullanılırlar.

SR-Latch



S	R	Q_1	Q_2
0	0	0	1
0	0	1	0
0	1	0	1
1	0	1	0
1	1	X	X

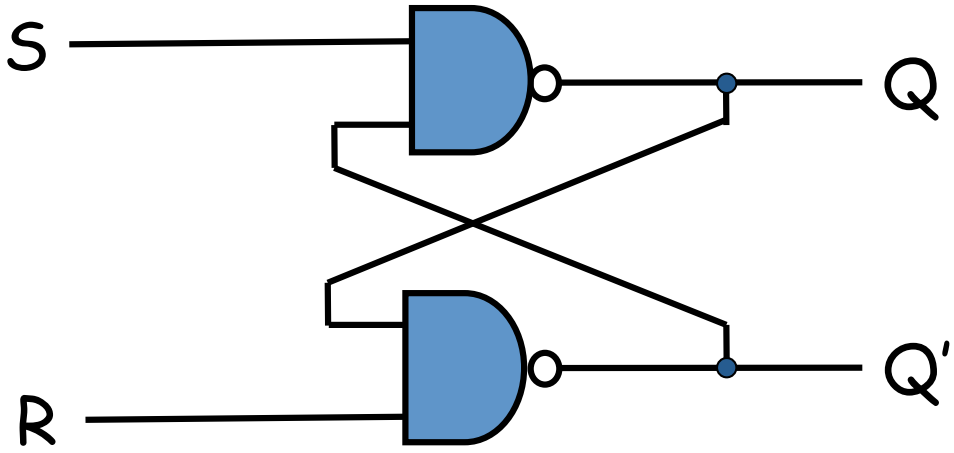
$Q_1 = (R + Q_2)' = R' Q_2'$

$Q_2 = (S + Q_1)' = S' Q_1'$

$Q_1 = Q_2'$

Tanımsız

SR-Latch



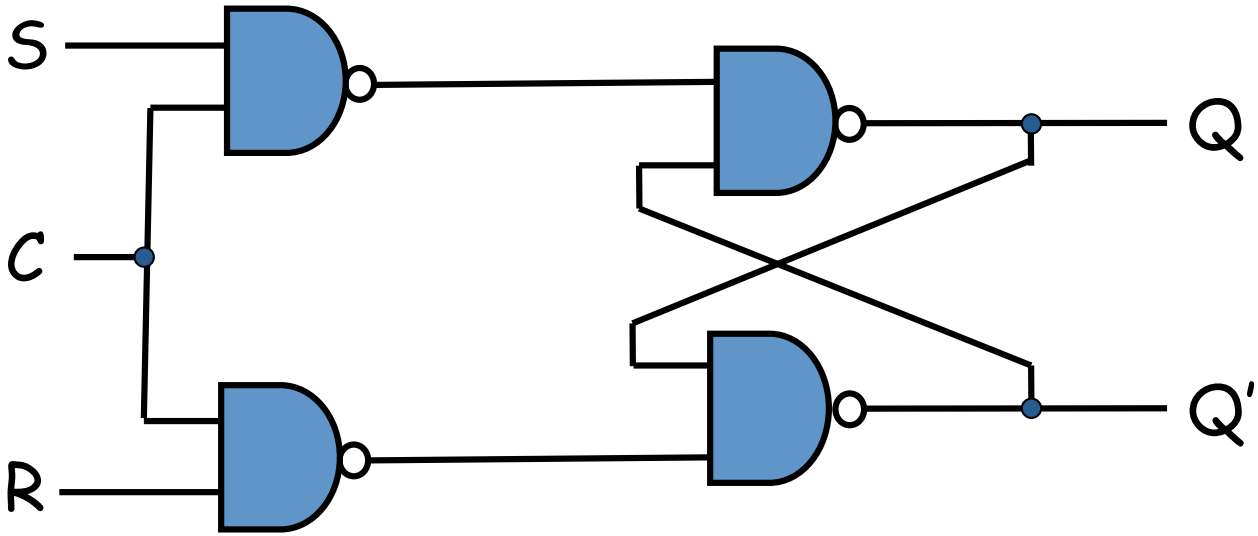
$Q = (S\ Q')' = S' + Q$

$Q' = (R\ Q)' = R' + Q'$

S	R	Q	Q'
0	0	x	x
0	1	1	0
1	0	0	1
1	1	1	0
1	1	0	1

Tanımsız

Kontrol Girişli SR-Latch



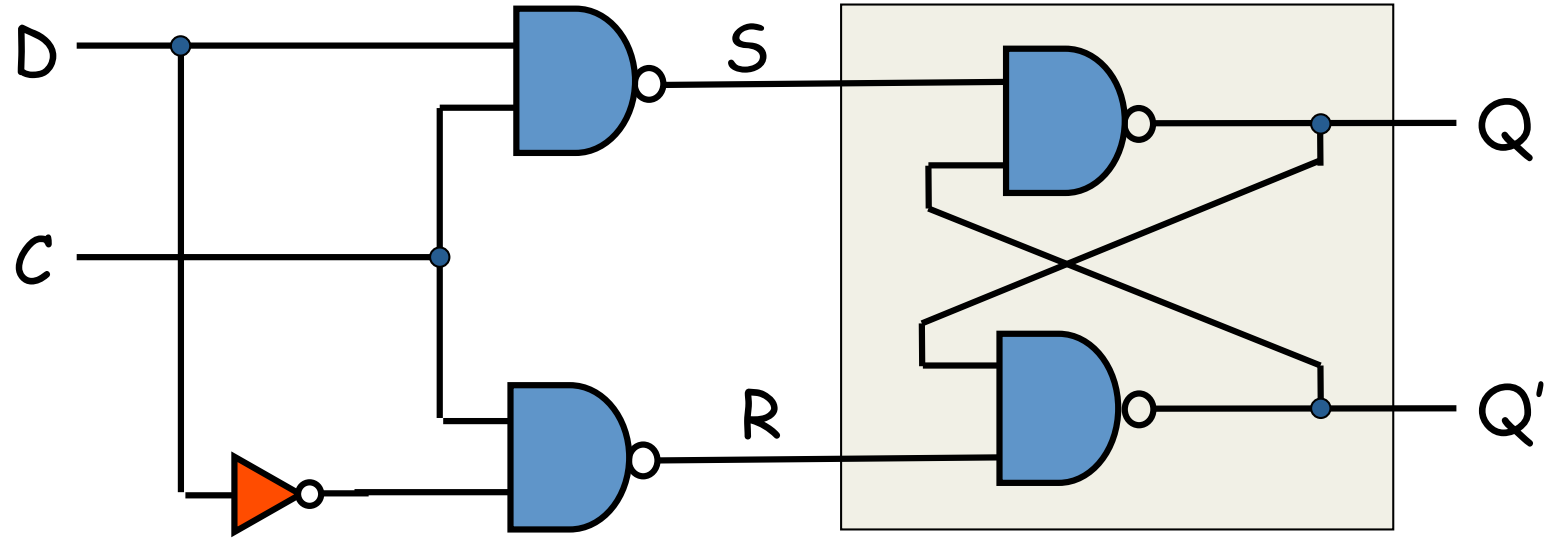
$Q = ((S\,C)'Q')' = SC + Q$

$Q' = ((R\,C)'Q)' = RC + Q'$

C	S	R	Q	Q'
0	X	X	Değişme yok	
1	0	0	Değişme yok	
1	0	1	Q = 0	Reset durumu
1	1	0	Q = 1	Set durumu
1	1	1	Tanımsız	

D-Latch

- Tanımsız hal devrede kararsızlığa sebep olabileceği için SR latch ler sık kullanılmaz.
- Çözüm: D-latch ler

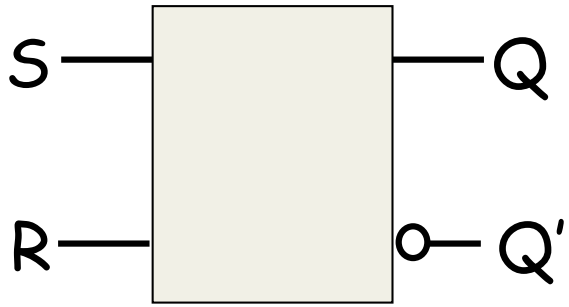


Bu devre S ve R girişlerinin her zaman birbirlerinin tümleyeni olmasını sağlar.

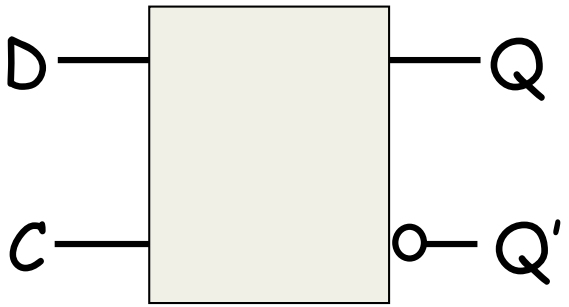
D-Latch

C	D	Q'nun sonraki durumu
0	X	Değişim yok
1	0	Q = 0; reset durumu
1	1	Q = 1; set durumu

- C=1 iken D girişi örneklenir.



SR-latch



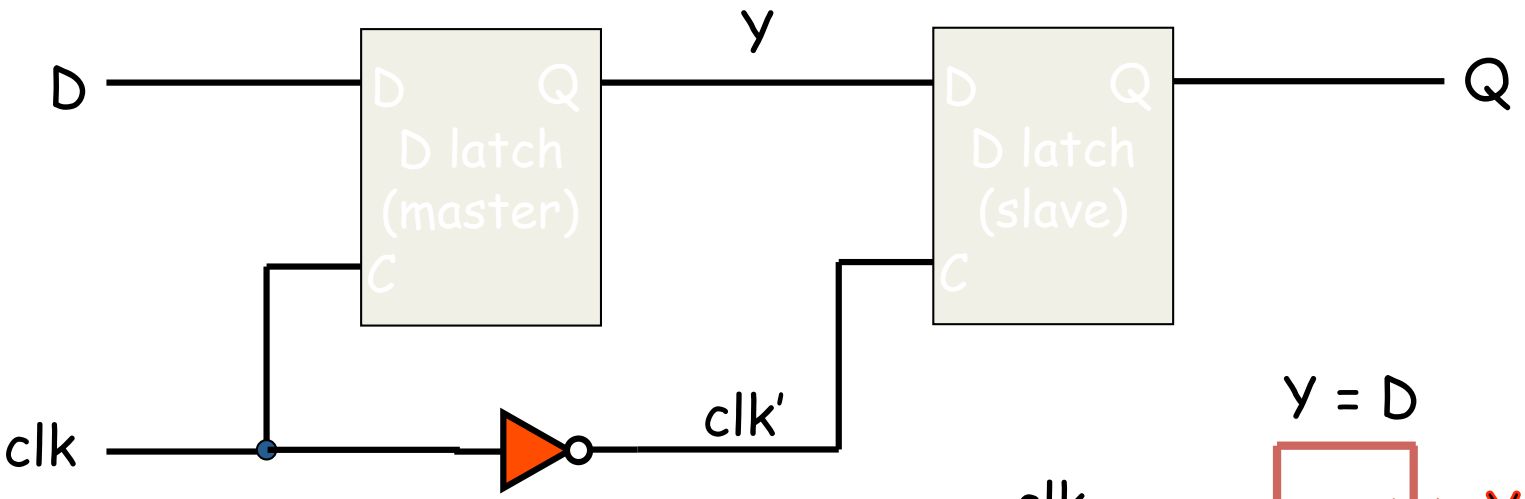
D-latch

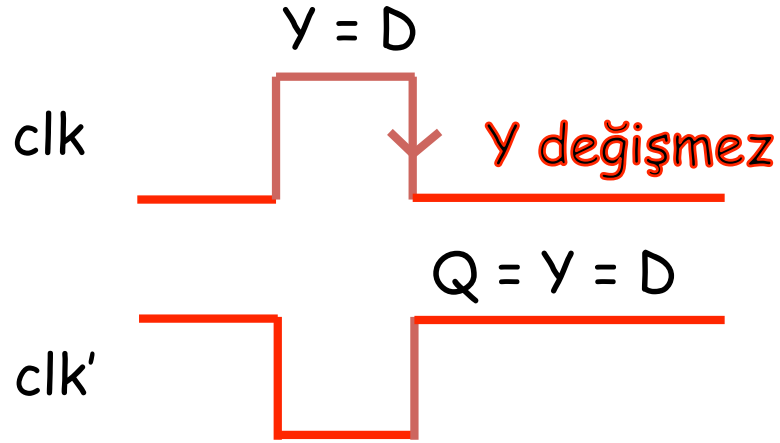
Saklama elemanı olarak D-Latch

- D–latch ler geçici saklama için kullanılabilirler.
- C = 1 olduğu sürece D-latch girişi çıkışa aktarılır.
- C = 0 olduğu sürece bilgi korunur.
- Latch ler seviye tetiklemeli olarak adlandırılır.
  - C lojik-1 seviyesinde kaldığı sürece veri girişindeki değişim durumu ve latch çıkışını değiştirir.
- Bellek elemanlarının durumları senkron olarak değişmeli.
- Düşen veya yükselen kenar tetiklemeli bellek elemanlarına flip-flop lar denir.

Kenar Tetiklemeli D Flip-Flop

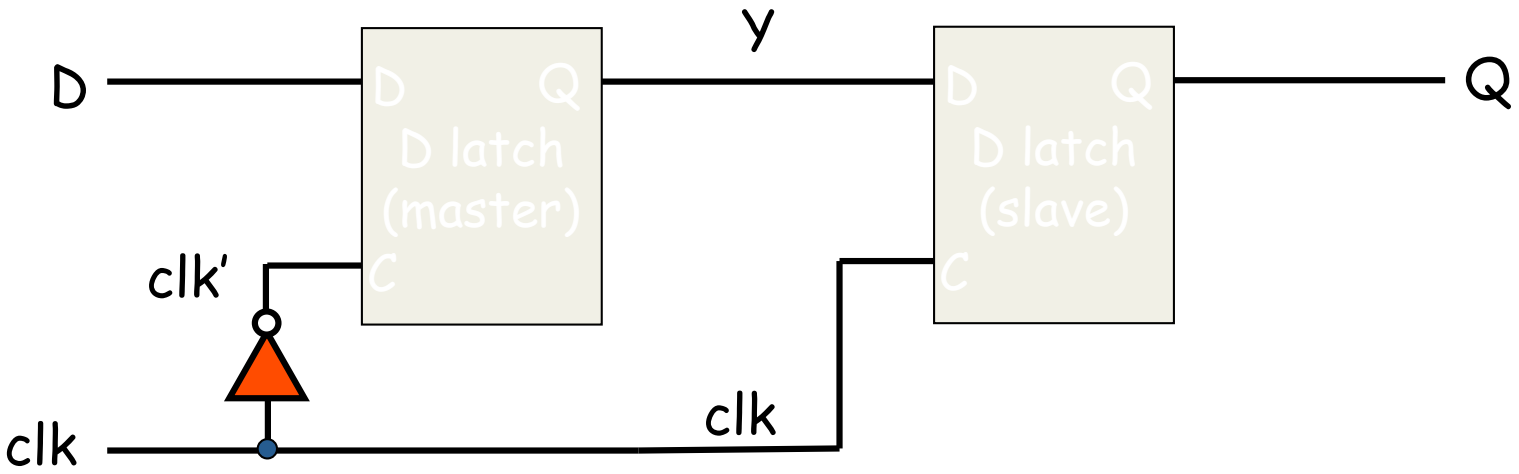
- Kenar tetiklemeli D flip-flop iki D latch kullanılarak yapılabilir.

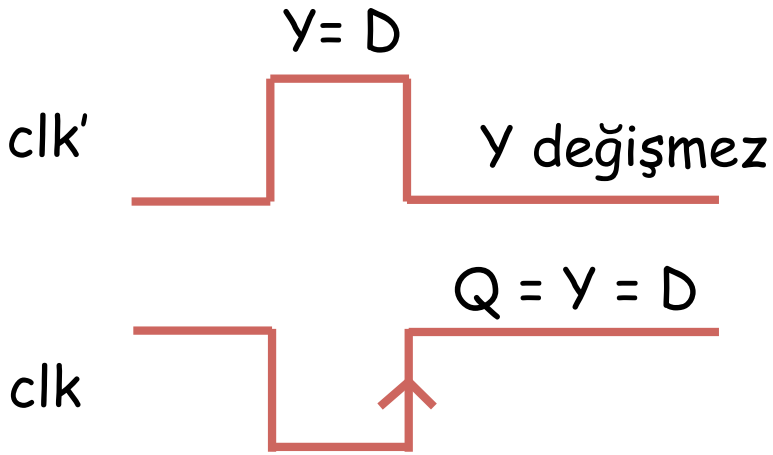




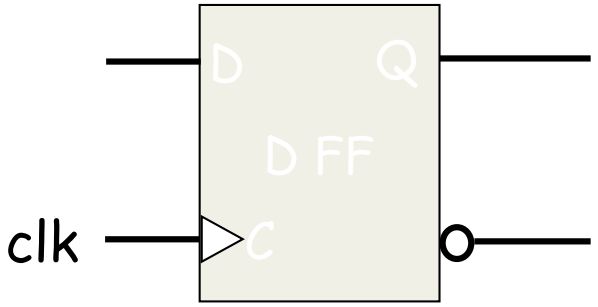
Düşen kenar tetiklemeli
D flip-flop

Yükselen Kenar Tetiklemeli D Flip-Flop

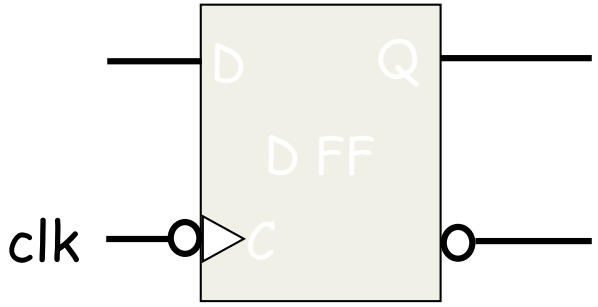




D Flip-Flop Sembolleri



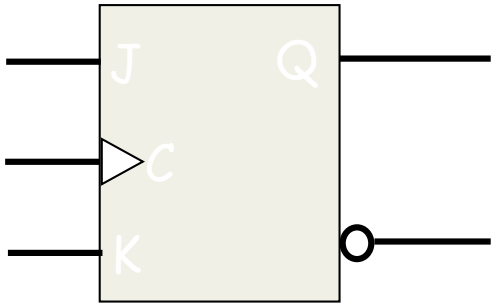
Yükselen kenar tetiklemeli
D Flip-Flop



Düşen kenar tetiklemeli
D Flip-Flop

- Karakteristik denklem
  - Q(t+1) = D

JK Flip-Flop lar

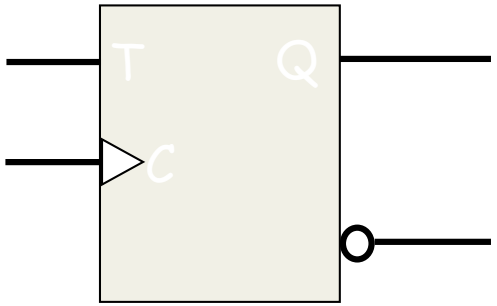


J	K	Q(t+1)	Sonraki durum
0	0	Q(t)	Değişim yok
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Tümleyen

Karakteristik Tablo

- Karakteristik denklem
  - $Q(t+1) = JQ'(t) + K'Q(t)$

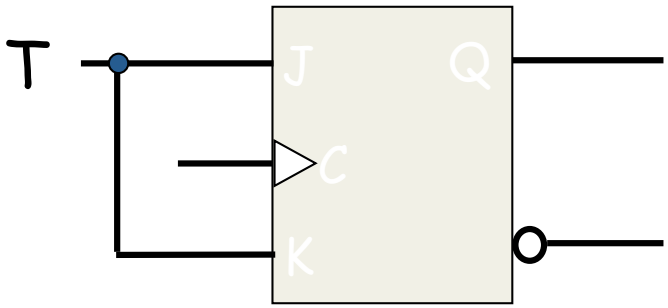
T (Toggle) Flip-Flop

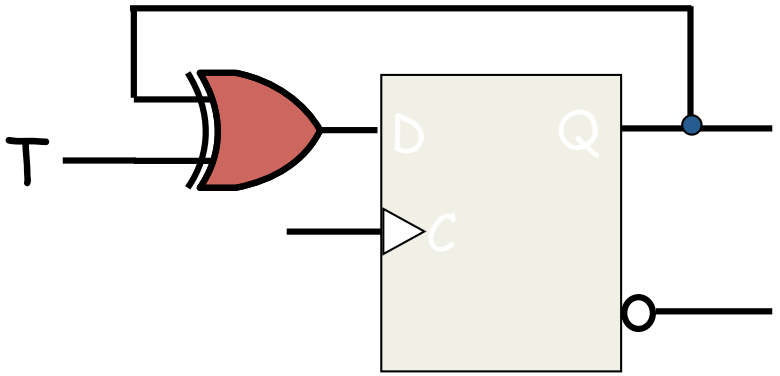


T	Q(t+1)	next state
0	Q(t)	no change
1	Q'(t)	Complement

Karakteristik Tablo

- Karakteristik denklem
  - $Q(t+1) = T \oplus Q = TQ' + T'Q$



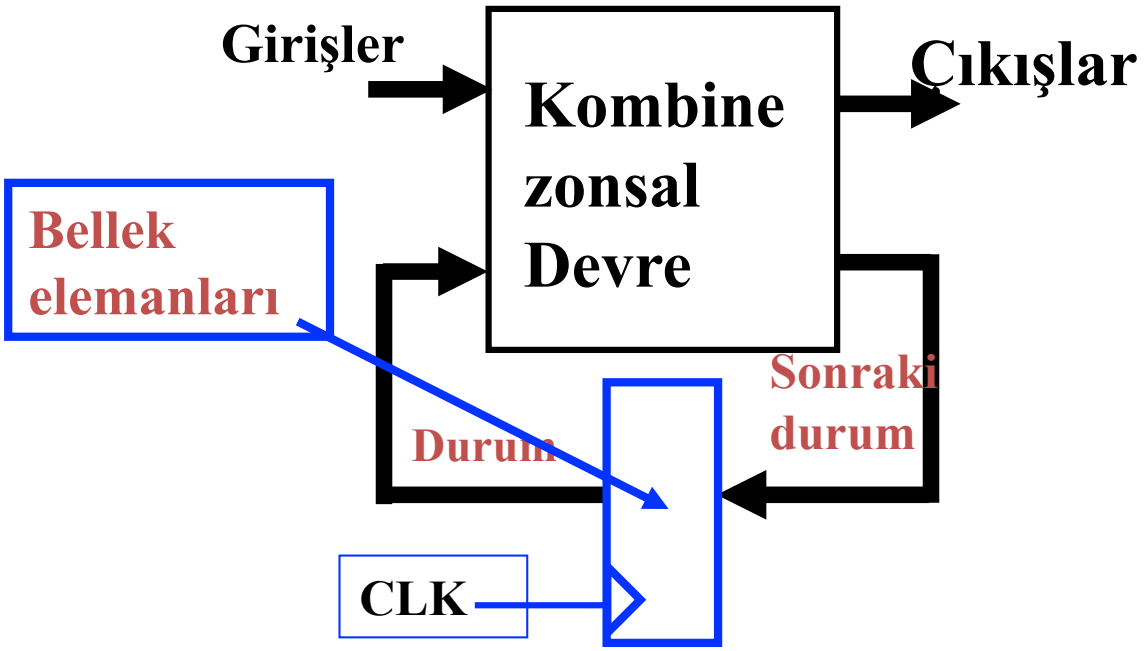


Senkron Ardışıl Devrelerin Analizi

- Amaç:
  - Senkron ardışıl devrelerin davranışını bulmak.
  - "Davranış"
    - Girişler
    - Çıkışlar
    - Flip-flop ların durumları kullanılarak elde edilir.
  - Çıkış ve sonraki durumun Boole fonksiyonlarını bulmak.
    - çıkış ve durum denklemleri
    - (durum) tablosu
    - (durum) diyagramı

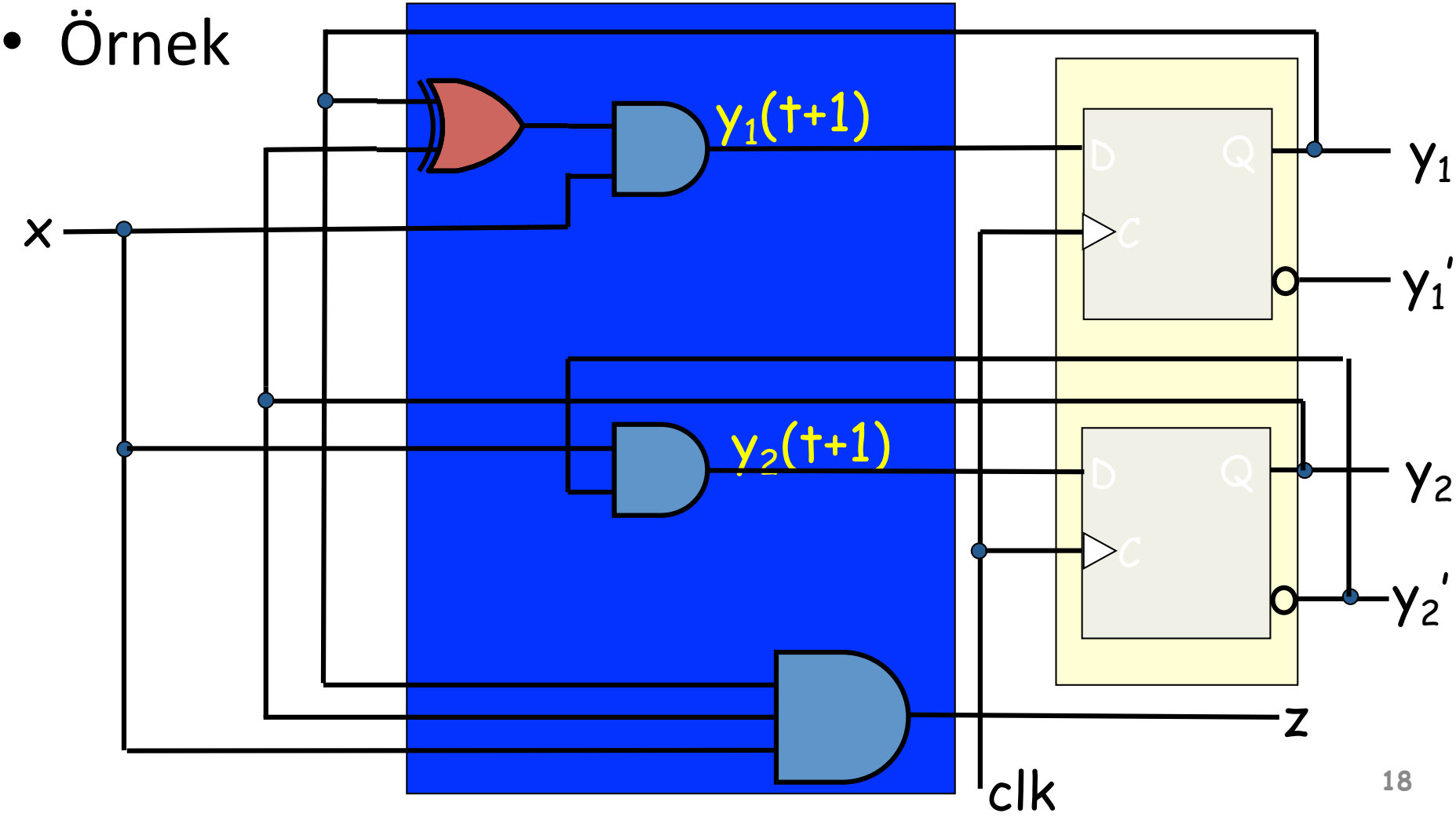
Senkron Ardışıl Devrelerin Analizi

- t anındaki şimdiki durum flip-flop dizisinde saklanır.
- (t+1) anındaki sonraki durum durum ve girişlerin oluşturduğu bir Boole fonksiyonu.
- t anındaki çıkışlar şimdiki durumlar ve bazen de girişlere bağlı Boole fonksiyonları.



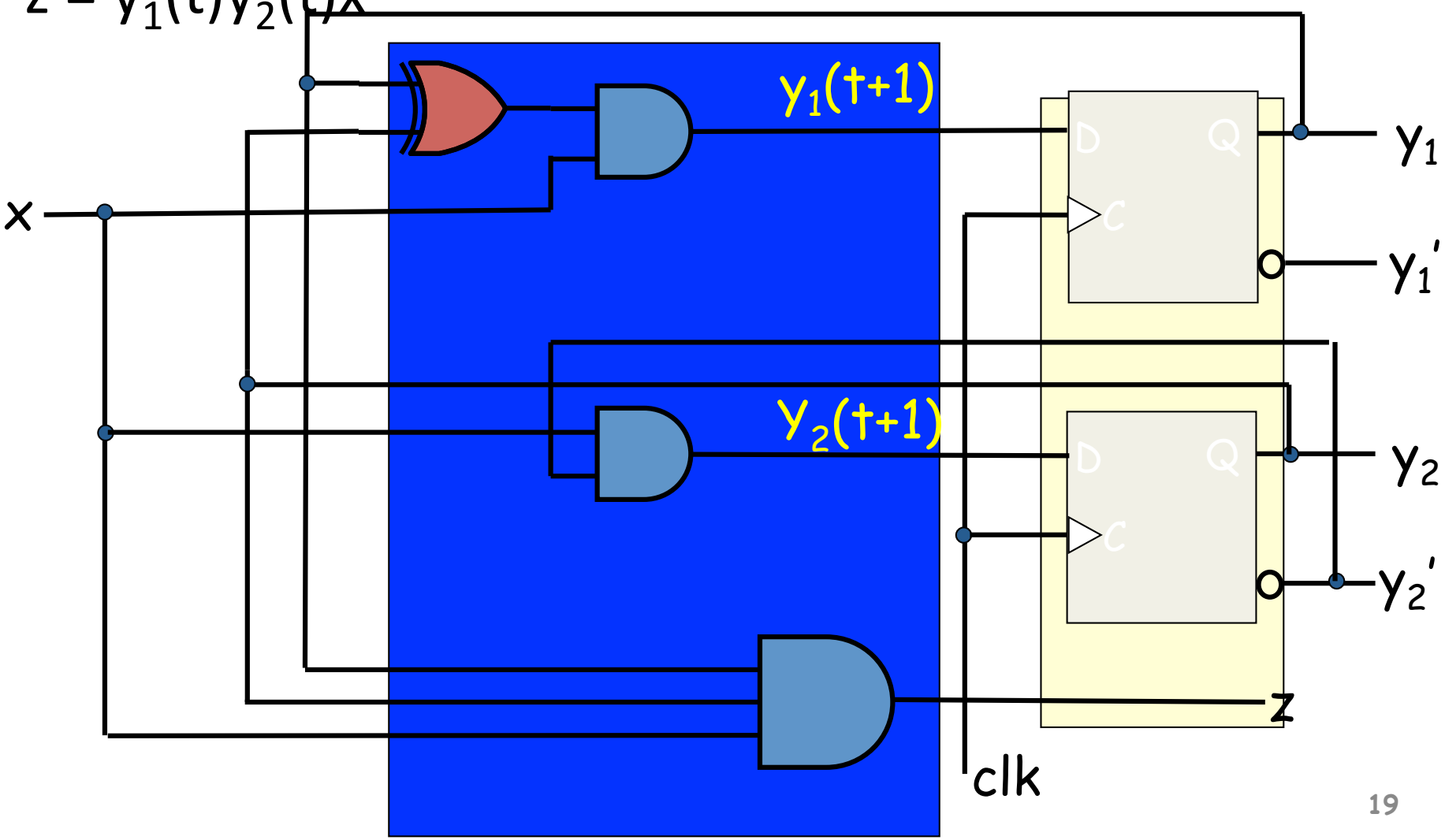
Durum Denklemleri

- Aynı zamanda geçiş denklemleri de denir.
  - Sonraki durumu şimdiki durum ve girişlerin bir fonksiyonu olarak verir.
- Örnek



Çıkış ve Durum Denklemleri

- $y_1(t+1) = (y_1(t) \oplus y_2(t))\, x$
- $y_2(t+1) = x\, y_2(t)'$
- $z = y_1(t) y_2(t) x$



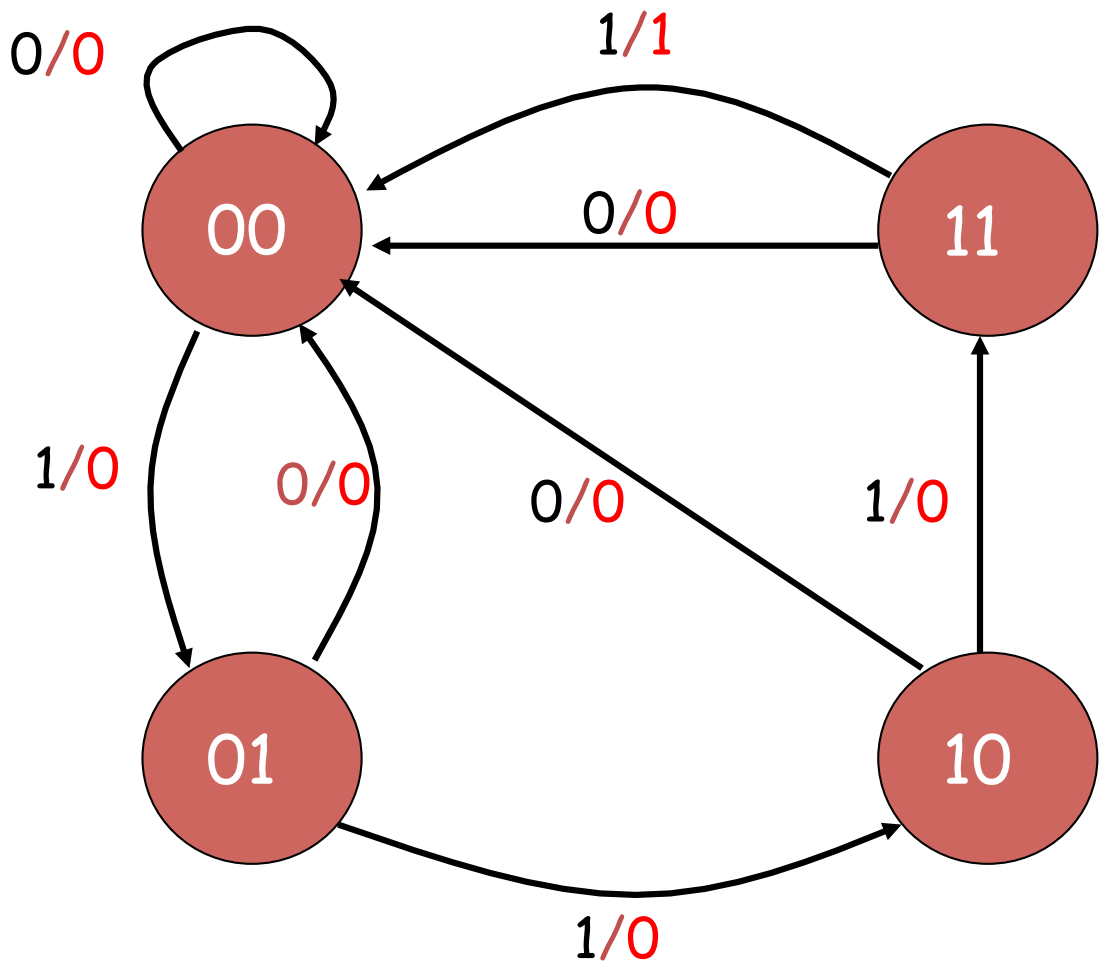
Örnek: Durum (Geçiş) Tablosu

$y_1(t+1) = ?$ $y_2(t+1) = ?$ $z = ?$

Şimdiki Durum		Giriş	Sonraki Durum		çıkış
y_1	y_2	x	Y_1	Y_2	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

m FF ve n girişi olan senkron ardışıl bir devrenin durum tablosunda 2^{m+n} satır vardır.

Örnek: Durum Diyagramı



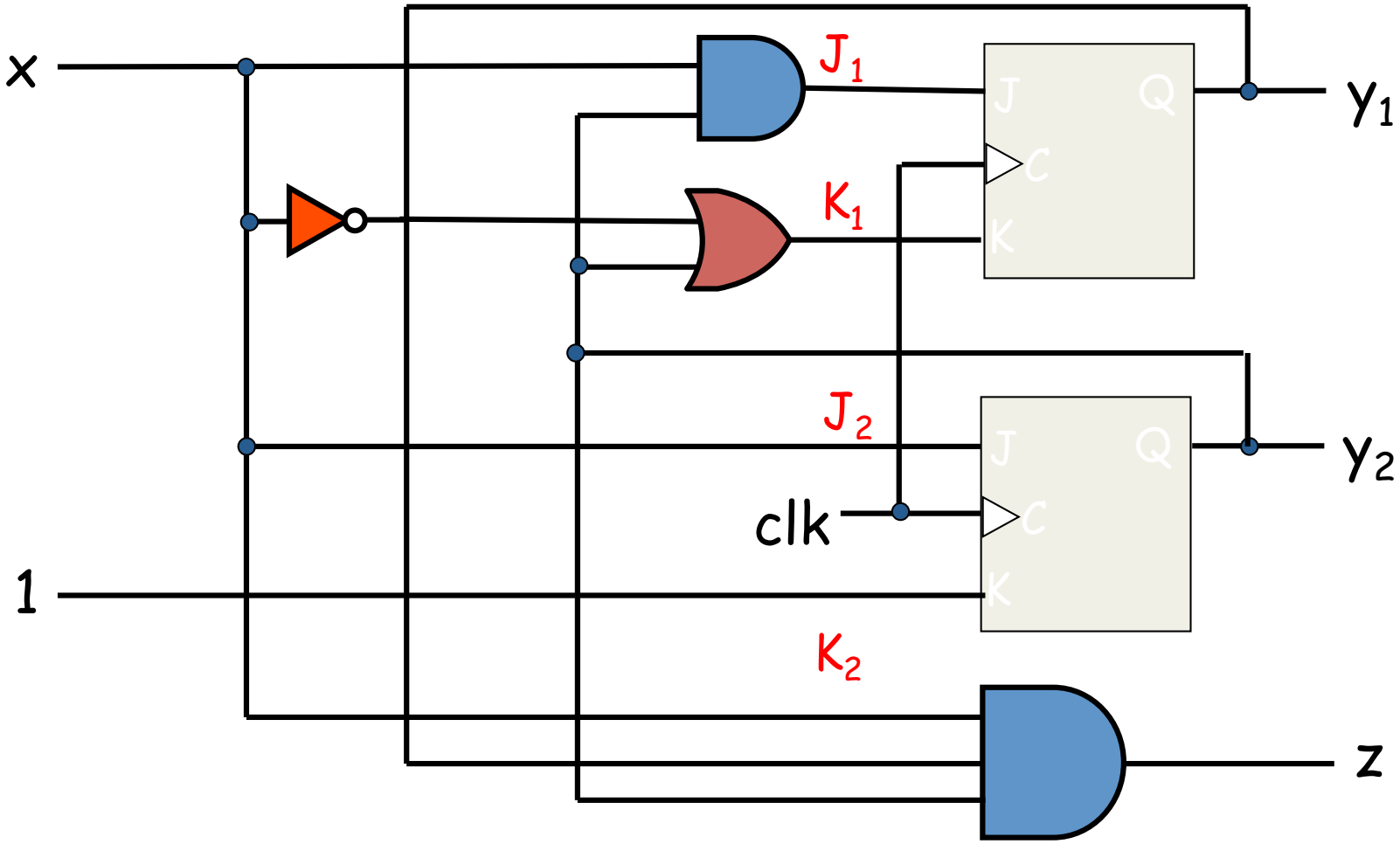
Şimdiki Durum		Giriş	Sonraki Durum		Çıkış
y_1	y_2	x	Y_1	Y_2	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

Durum diyagramı ile durum tablosu aynı bilgiyi verir.

JK tipi Flip-Flop lar ile Analiz

- D tipi flip-flop da durum denklemi flip-flop un giriş denklemi ile aynı
  - Q(t+1) = D
- JK tipi flip-flop larda bu iki denklem farklı
  - Amacımız durum denklemlerini bulmak.
  - Yöntem
    1. Flip-flop giriş denklemlerinin bulunması
    2. Her giriş denkleminin doğruluk tablosu oluşturulması
    3. Flip-flop ların karakteristik tablosu kullanılarak durum tablosundaki sonraki durum değerlerinin belirlenmesi

Örnek: JK tipi Flip-Flop lar ile Analiz



- Flip-flop input equations
  - $J_1 = xy_2$ ve $K_1 = x' + y_2$
  - $J_2 = x$ ve $K_2 = 1$

Örnek: JK tipi Flip-Flop lar ile Analiz

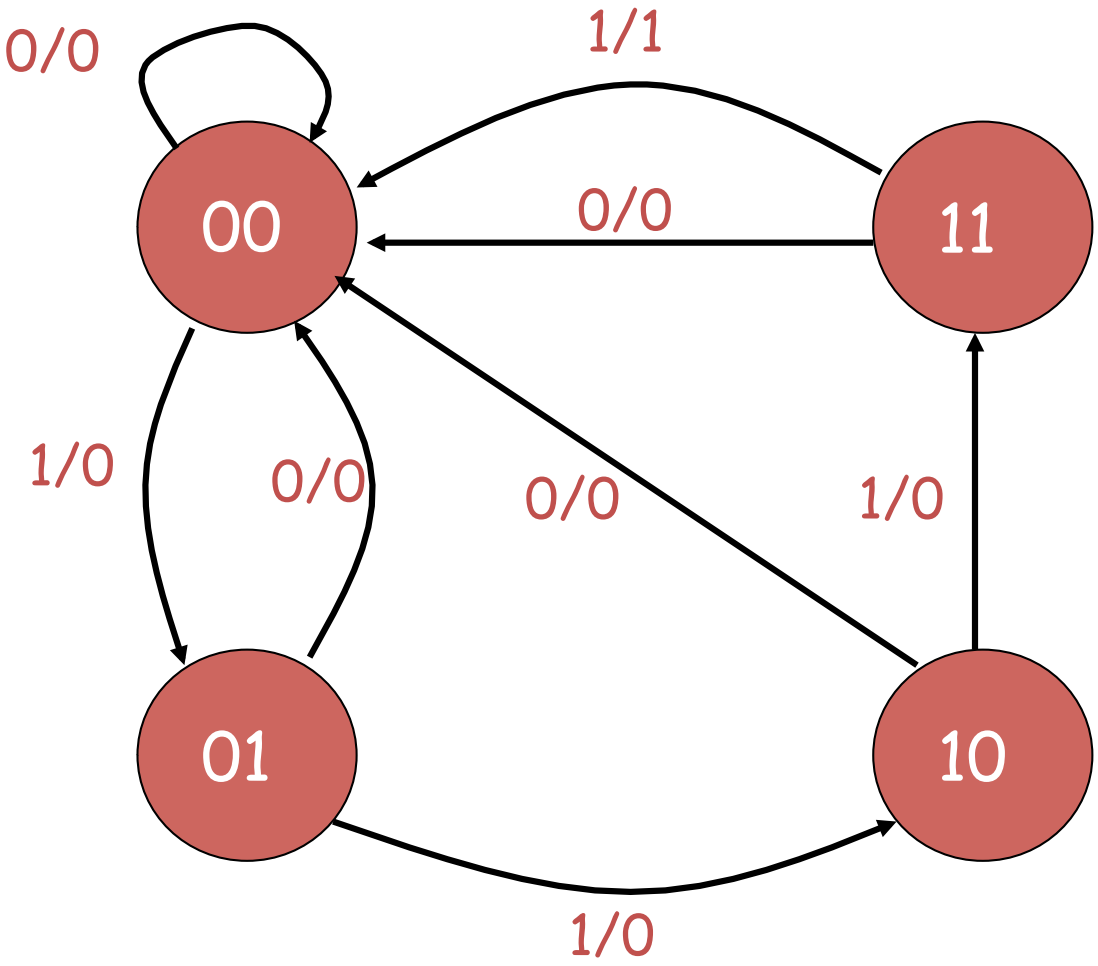
- $J_1 = xy_2$ ve $K_1 = x' + y_2$
- $J_2 = x$ ve $K_2 = 1$

Şimdiki Durum		Giriş	Sonraki Durum		FF girişleri			
y_1	y_2	x	Y_1	Y_2	J_1	K_1	J_2	K_2
0	0	0	0	0	0	1	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	0	0	1	0	1
0	1	1	1	0	1	1	1	1
1	0	0	0	0	0	1	0	1
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	1	0	1
1	1	1	0	0	1	1	1	1

Örnek: JK tipi Flip-Flop lar ile Analiz

- Karakteristik Denklemler
  - $Y_1 = J_1 y_1' + K_1' y_1$
  - $Y_2 = J_2 y_2' + K_2' y_2$
- Flip-flop giriş denklemleri
  - $J_1 = x y_2$ ve $K_1 = x' + y_2$
  - $J_2 = x$ ve $K_2 = 1$
- Durum denklemleri
  - $Y_1 = x y_2 y_1' + (x' + y_2)' y_1 = x y_2 y_1' + x y_2' y_1 = x(y_2 \oplus y_1)$
  - $Y_2 = x y_2' + 1' y_2 = x y_2'$

Durum Diyagramı



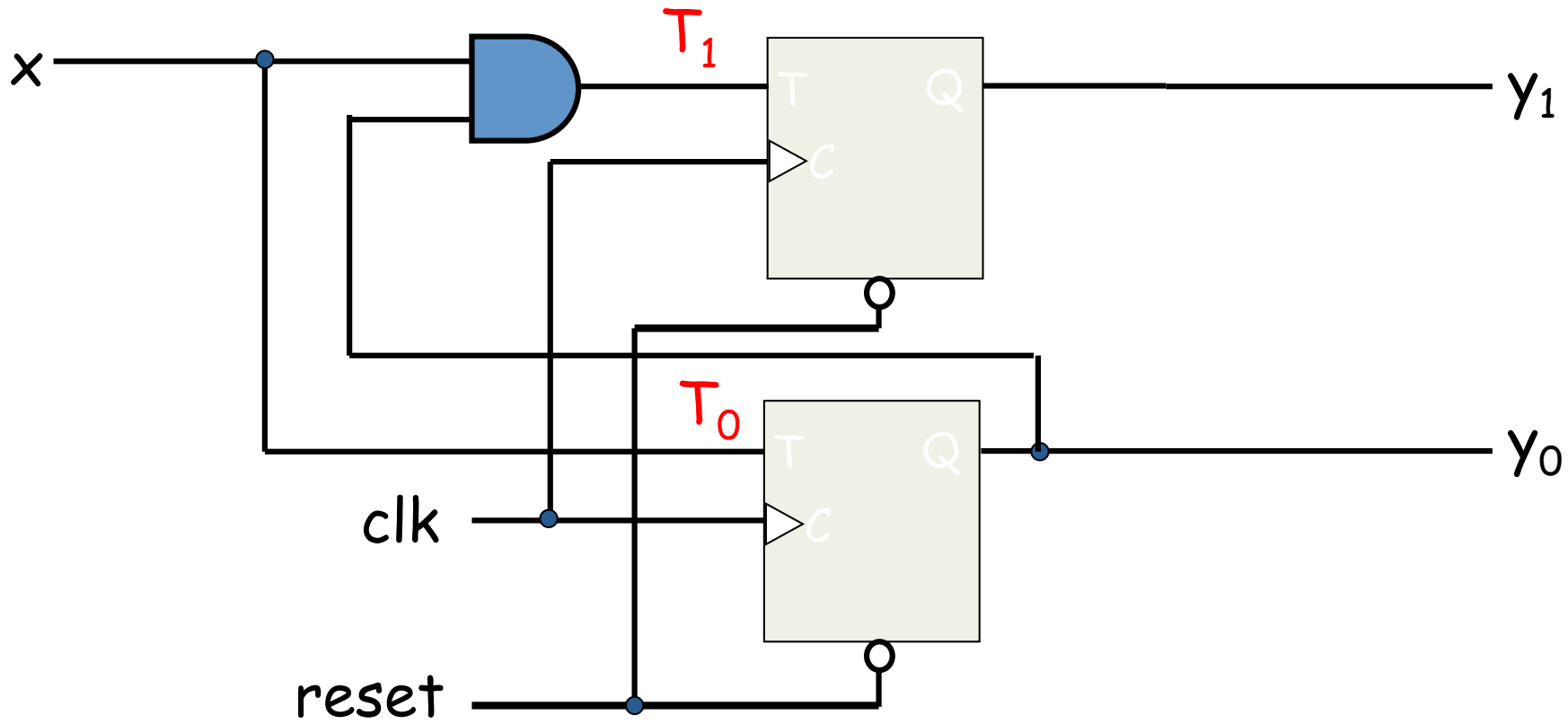
Şimdiki Durum		Giriş	Sonraki Durum		Çıkış
y_1	y_2	x	Y_1	Y_2	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

Devre ne yapıyor?

T tipi Flip-Flop lar ile Analiz

- Yöntem aynı
- Örnek

$T_1 = xy_0$
$T_2 = x$



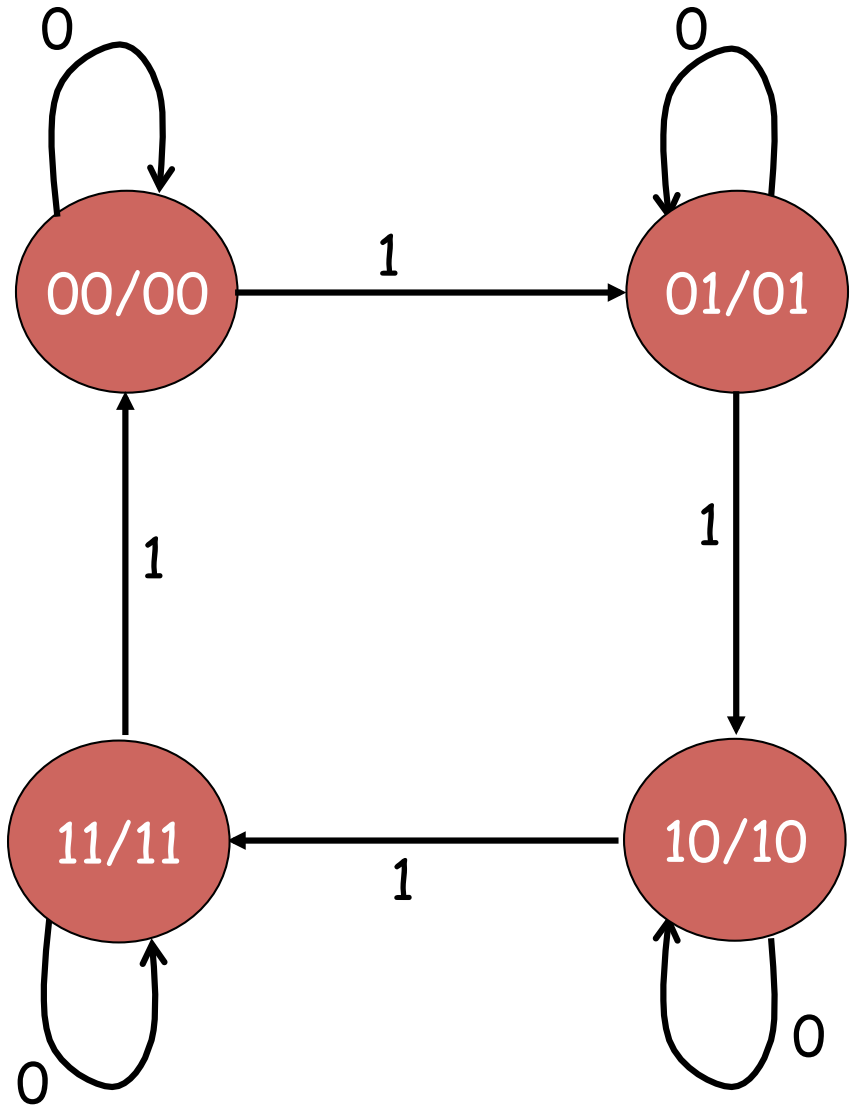
Örnek: T tipi Flip-Flop lar ile Analiz

- Karakteristik Denklemleri
  - $Y_0 = T_0 \oplus y_0$
  - $Y_1 = T_1 \oplus y_1$
- Flip-flop giriş denklemleri
  - $T_1 = x\, y_0$
  - $T_0 = x$
- Durum denklemleri
  - $Y_0 = x \oplus y_0$
  - $Y_1 = x\, y_0 \oplus y_1$

Durum Tablosu ve Diyagramı

- $Y_0 = x \oplus y_0$
- $Y_1 = x\, y_0 \oplus y_1$

Şimdiki Durum		Giriş	Sonraki Durum		Çıkış	
y_1	y_0	x	Y_1	Y_0	y_1	y_0
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	1	1	0
1	1	0	1	1	1	1
1	1	1	0	0	1	1

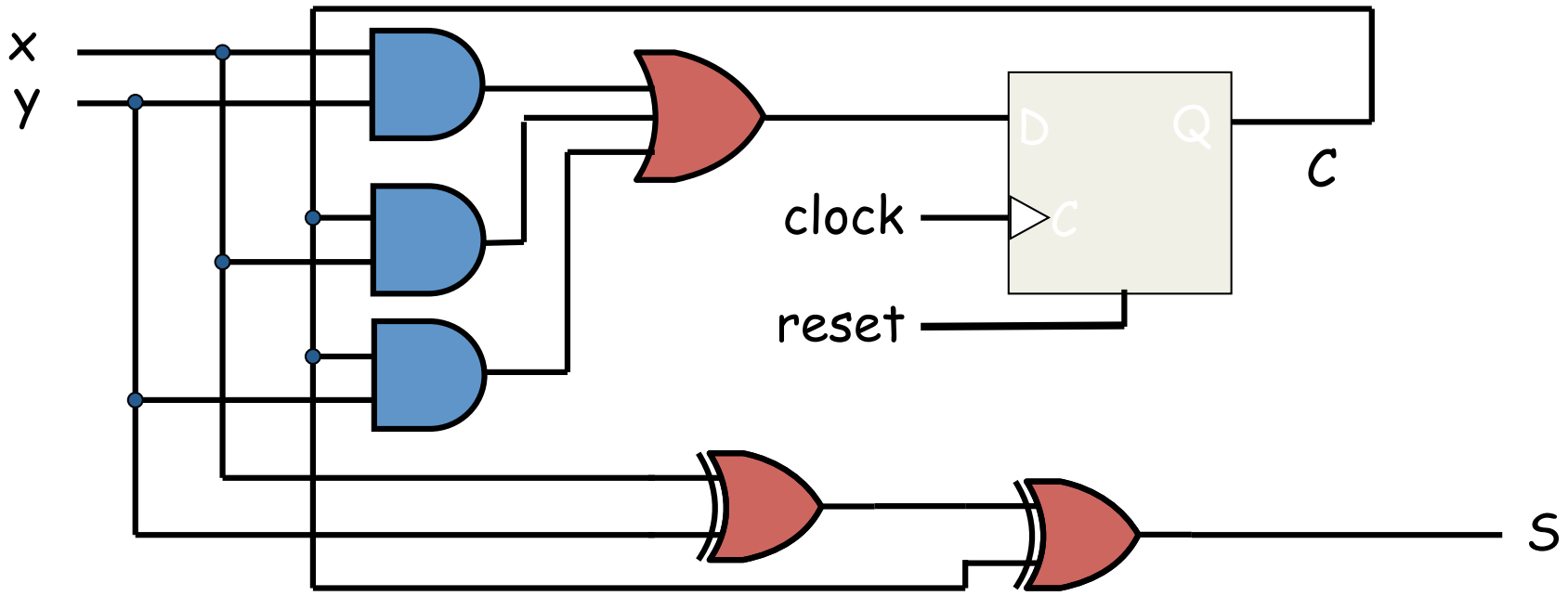


Moore ve Mealy Modelleri

- Senkron ardışıl devreler veya senkron makinalar aynı zamanda sonlu durum makinaları (*Finite State Machines* (FSMs)) olarak adlandırılırlar.
- İki tip model vardır:

- **Moore Modeli**
  - **E. F. Moore tarafından ortaya atılmıştır.**
  - **Çıkışlar SADECE durumlara bağlıdır.**
  - **Çıkışlar durum diyagramında durumların üzerinde gösterilir.**
- **Mealy Modeli**
  - **G. Mealy tarafından ortaya atılmıştır.**
  - **Çıkışlar girişlere VE durumlara bağlıdır.**
  - **Çıkışlar durum diyagramında durum geçiş çizgilerinin üzerinde gösterilir.**

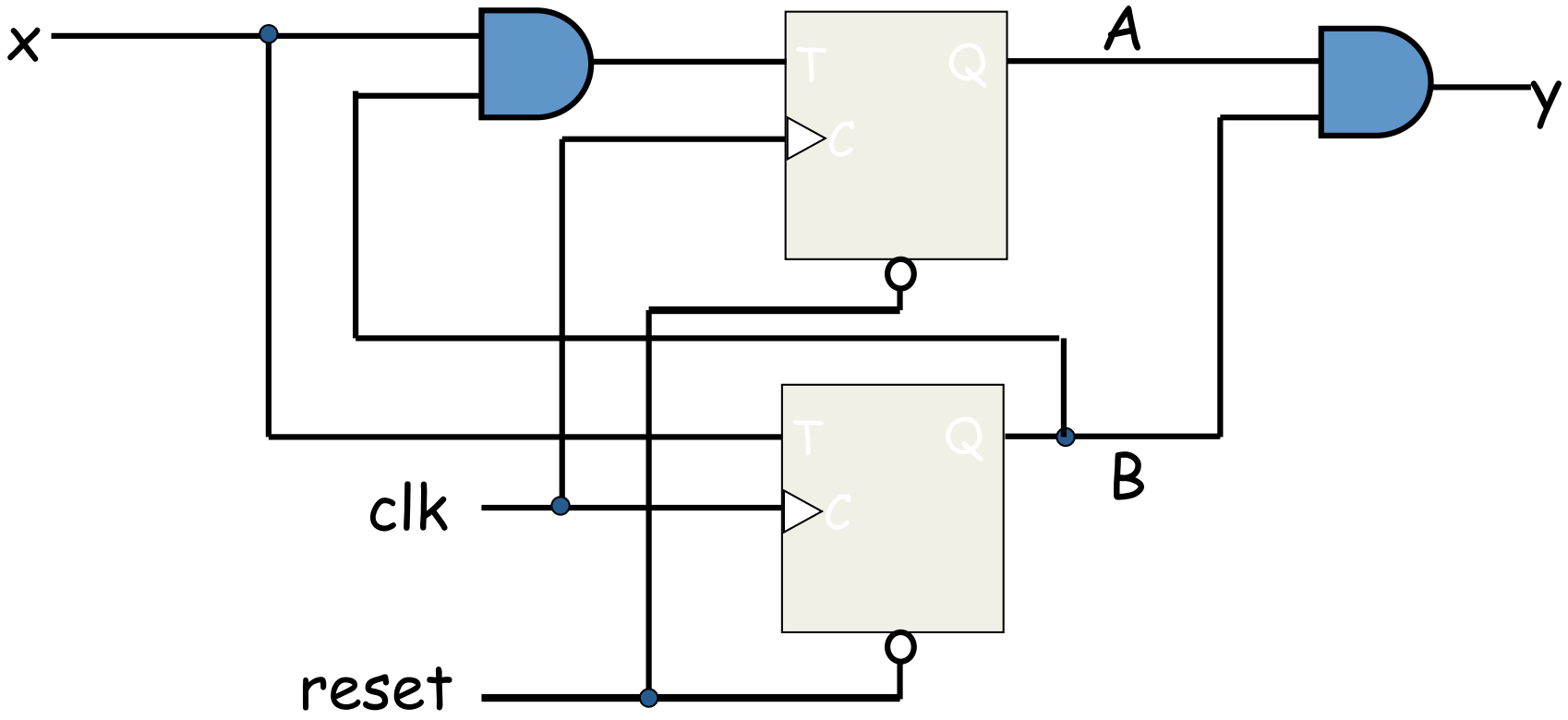
Örnek: Mealy ve Moore Makinaları



Mealy makinası

- x ve y girişleri senkron değiller.
- Bu sebeple, çıkışlar kısa süreli yanlış değerler alabilirler.
- Girişler saat işareti ile senkron hale getirilmelidir veya
- Çıkışlar sadece saatin yükselen kenarında örneklenmelidir.

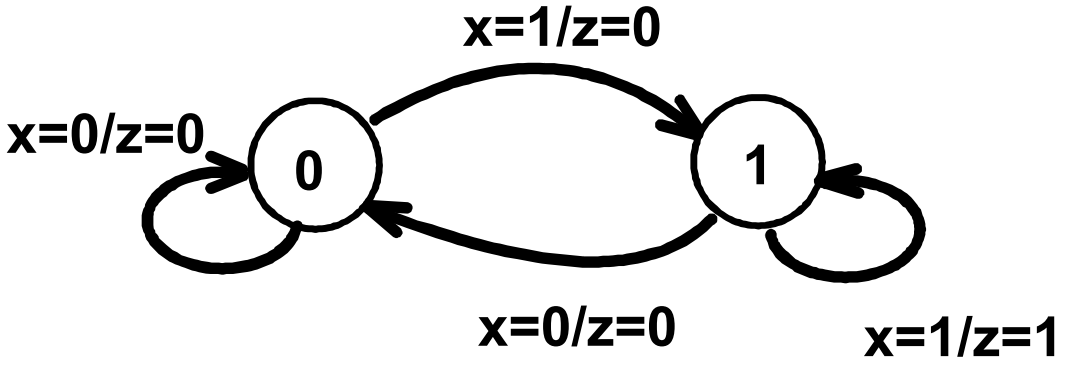
Örnek: Moore Makinası



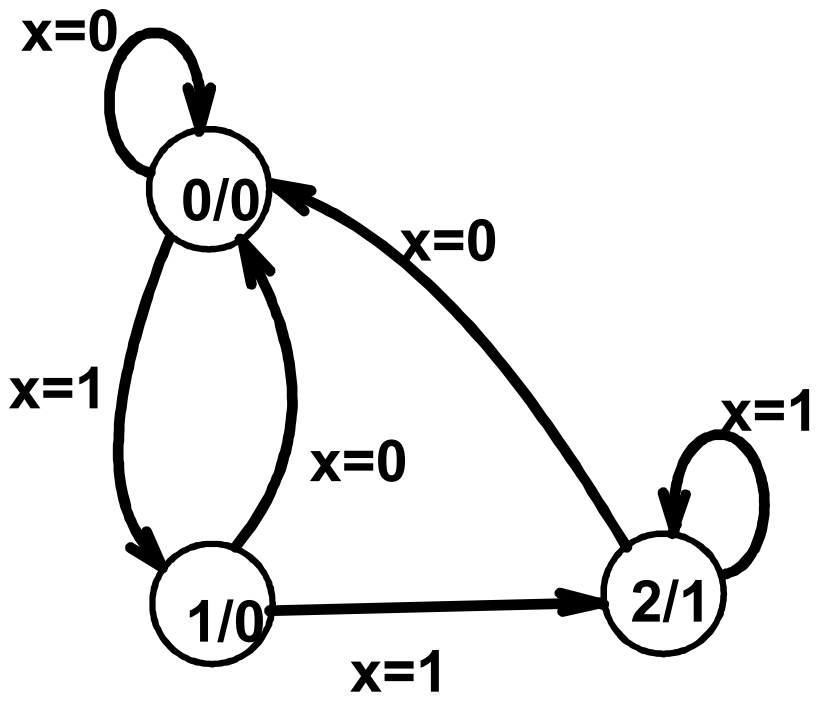
- Çıkışlar saat işareti ile senkron olarak çalışır.

Moore ve Mealy Örnek Diyagramları

- Mealy Modeli durum diyagramı



- Moore Modeli durum diyagramı



Moore ve Mealy Örnek Durum Tabloları

- Moore Model durum tablosu

Şimdiki Durum	Sonraki Durum		Çıkış
	x=0	x=1	
0	0	1	0
1	0	2	0
2	0	2	1

- Mealy Model durum tablosu

Şimdiki Durum	Sonraki Durum		Çıkış	
	x=0	x=1	x=0	x=1
0	0	1	0	0
1	0	1	0	1

Senkron Ardışıl Devre Tasarımı

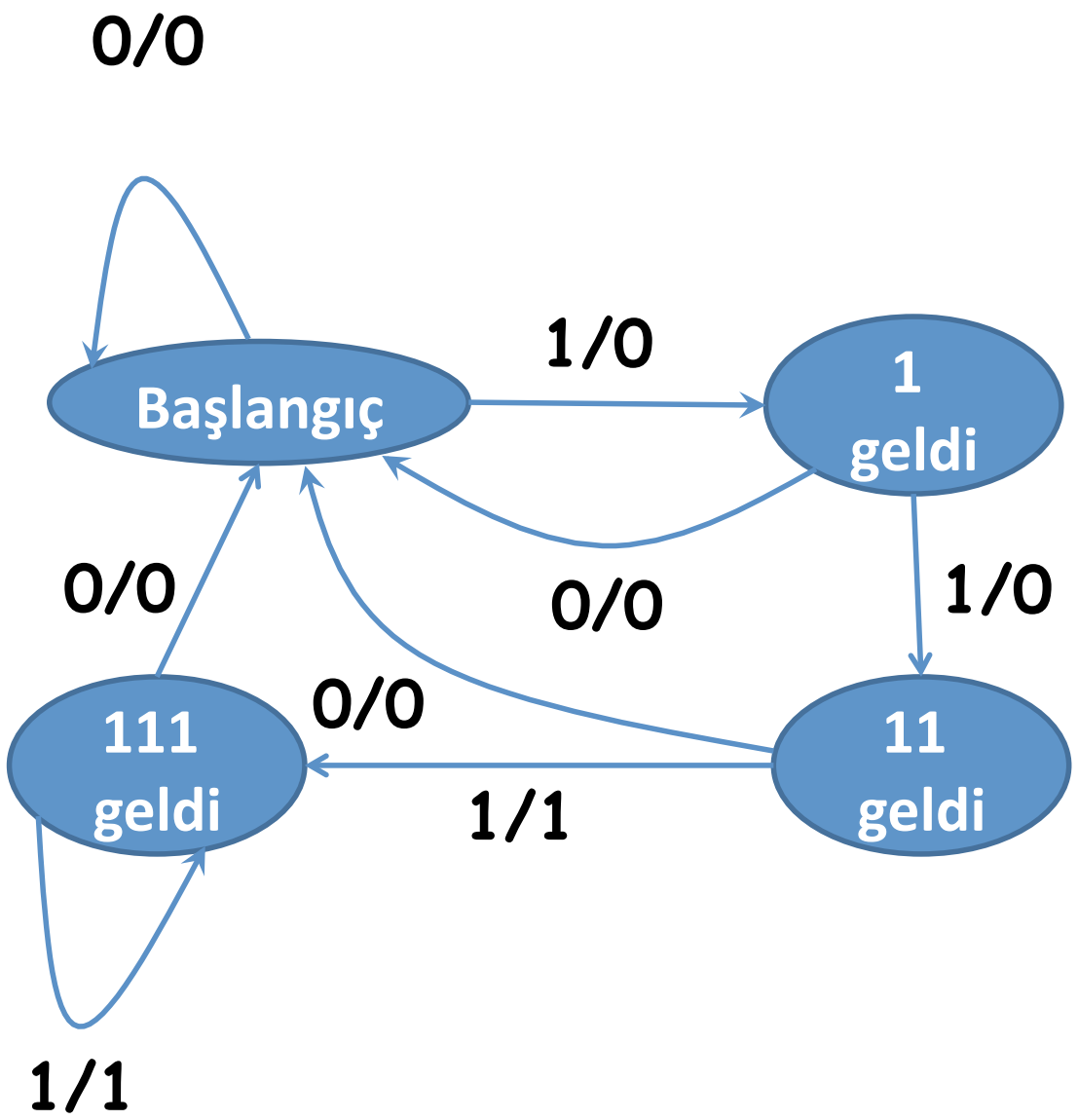
1. Problemin sözle tanımı
2. Durum diyagramının çizilmesi
3. Durumların indirgenmesi: s = durum sayısı
4. Flip-flop sayısının belirlenmesi: $\mathrm{n} = \lceil \log_2 s \rceil$
5. Durumların kodlanması: $\underbrace{00\ldots0}_{n-bit}, \underbrace{00\ldots1}_{n-bit}, \underbrace{00\ldots10}_{n-bit}, \ldots$
6. Durum tablosunun çıkarılması
7. Flip-flopların tipinin belirlenmesi
8. Boole Fonksiyonlarının elde edilmesi
    1. Flip-flopların giriş fonksiyonları
    2. Çıkış fonksiyonları
9. Boole fonksiyonlarının gerçeklenmesi

Örnek: Senkron Ardışıl Devre Tasarımı

- Sözle tanım
  - 1. Adım: 1-bitlik girişinden ard arda 3 tane veya daha fazla 1 geldiğini sezen devreyi tasarlayınız.
  - Giriş: herhangibir uzunluktaki bit dizisi
  - Çıkış:
    - “1” : eğer devre istenen diziyi yakalamışsa
    - “0” : diğer hallerde

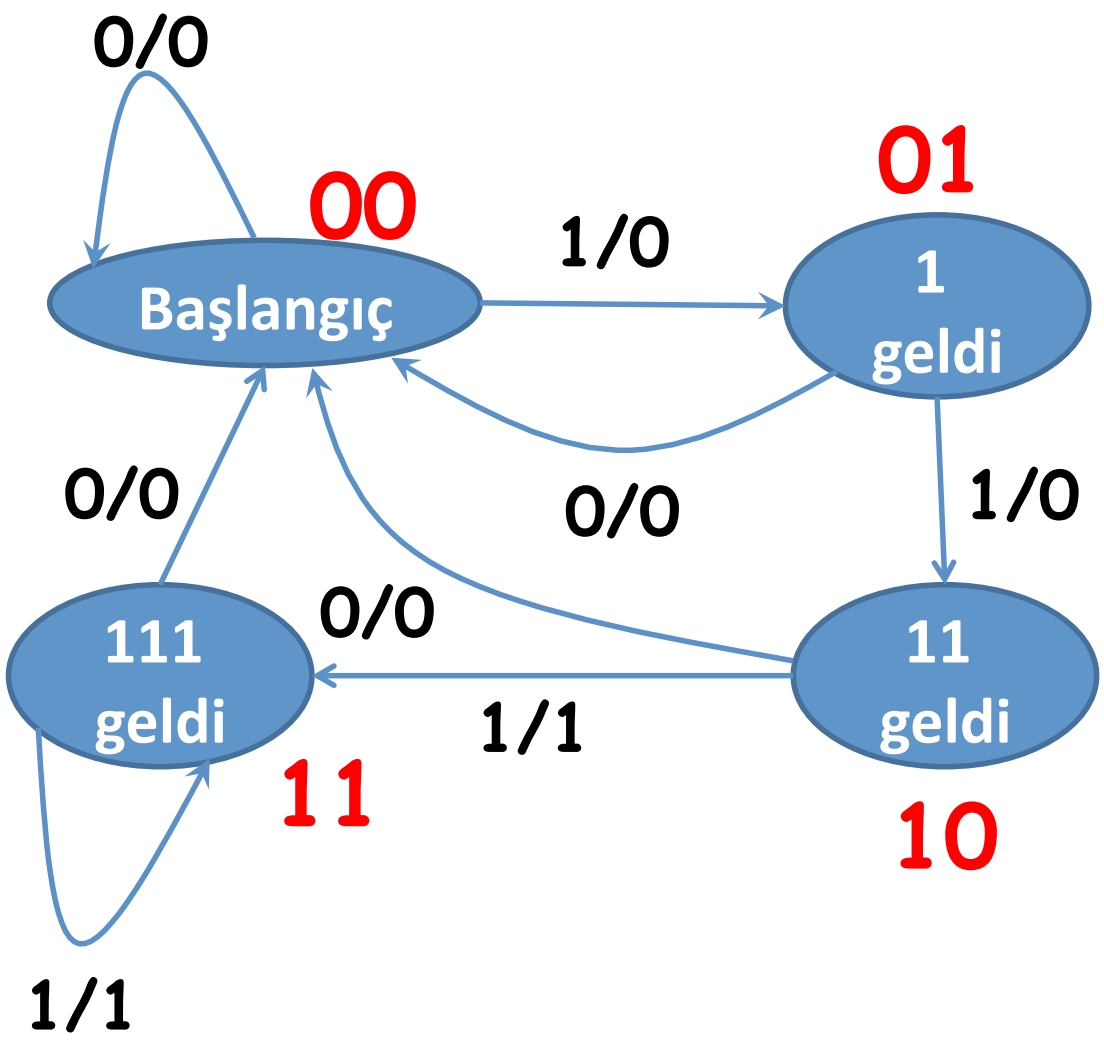
Örnek: Durum Diyagramı

2. Adım: Durum diyagramının çizilmesi



D Tipi Flip-Floplar ile tasarım

- 3. Adım: Durum indirgeme
  - Mümkün değil
- 4. Adım: Flip-flop sayısı
  - 4 durum
  - ? flip-flop
- 5. Adım: Durum kodlama



D Tipi Flip-Floplar ile tasarım

- 6. Adım: Durum tablosunun çıkarılması

Şimdiki Durum		Giriş	Sonraki Durum		Çıkış
y_1	y_2	x	Y_1	Y_2	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

D Tipi Flip-Floplar ile tasarım

- 7. Adım: Flip-flopların tipinin belirlenmesi
  - D tipi flip-floplar
- 8. Adım: Boole Fonksiyonlarının elde edilmesi
  - D_1 ve D_2 için Boole fonksiyonları

y_1 \ y_2x	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$D_1 = y_1x + y_2x$

y_1 \ y_2x	00	01	11	10
0	0	1	0	0
1	0	1	1	0

$D_2 = y_1x + y_2'x$

D Tipi Flip-Floplar ile tasarım

– Z için Boole fonksiyonu

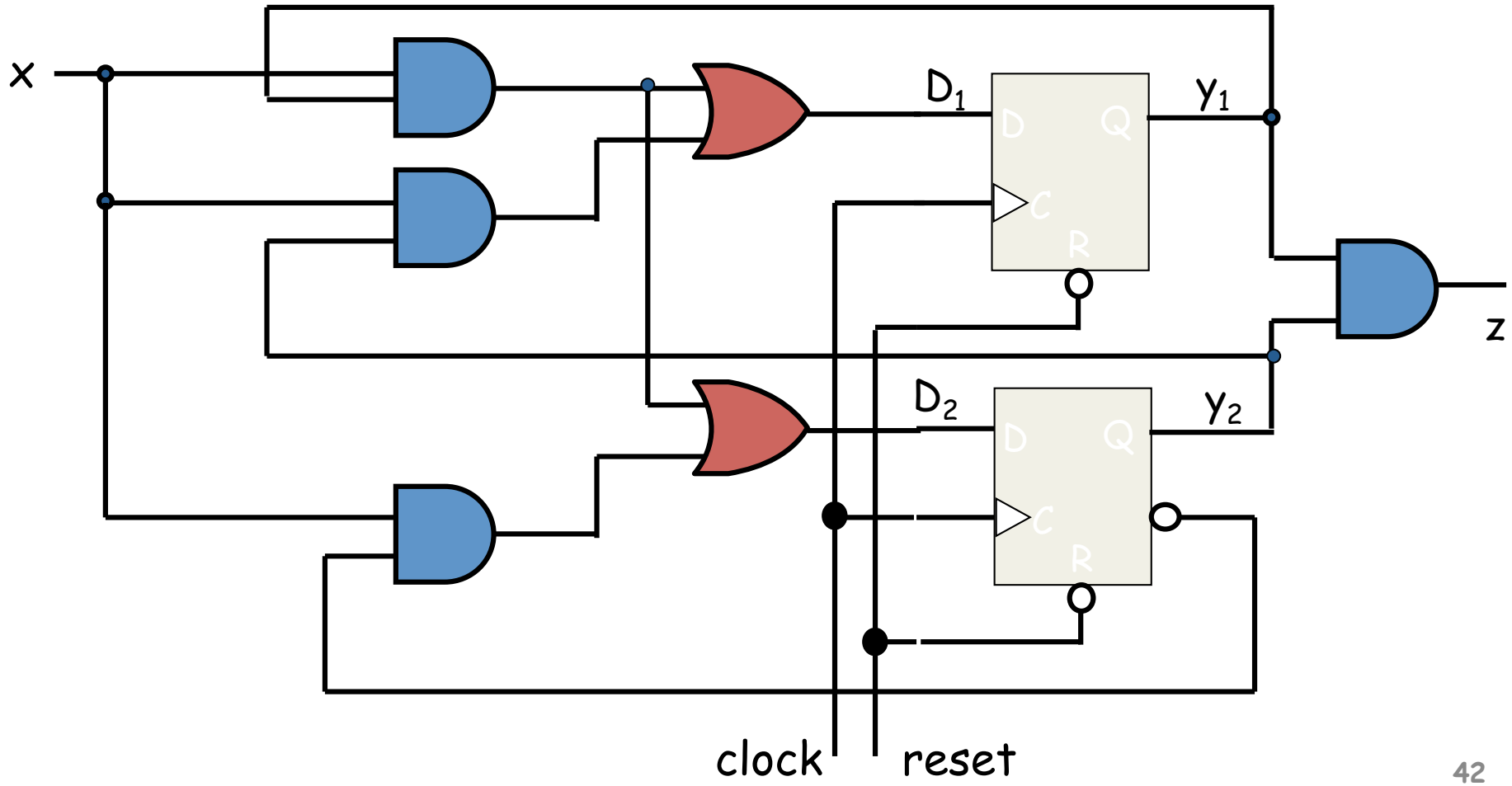
y_1 \ y_2x	00	01	11	10
0	0	0	0	0
1	0	0	1	1

$z = y_1 y_2$

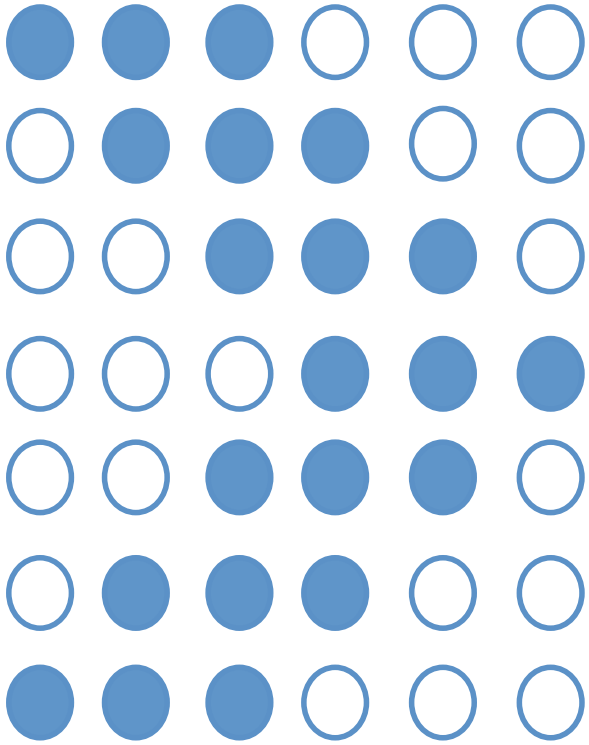
D Tipi Flip-Floplar ile tasarım

- 9. Adım: Boole fonksiyonlarının gerçeklenmesi

$D_1 = y_1x + y_2x$ $D_2 = y_1x + y_2'x$ $z = y_1y_2$



JK Tipi Flip-Floplar ve MUX ile tasarım



Durum sayısı= 6

Durum değişkeni sayısı= 3

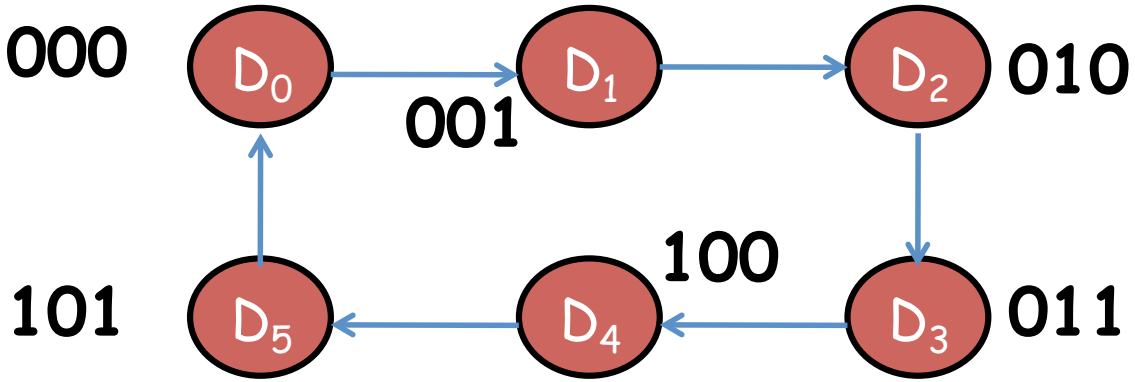
Flip-flop sayısı= 3

Giriş sayısı= 0

Çıkış sayısı= 6

- 6 tane kayan ışık
- ●= lojik-1
- O= lojik-0

Durum Diyagramı ve Tablosu



$Y = Jy' + K'y$

J	K	Y
0	0	y
0	1	0
1	0	1
1	1	Q'

Şimdiki Durum			Sonraki Durum			Flip-flop girişleri						Çıkışlar					
Y_2	Y_1	Y_0	Y_2	Y_1	Y_0	J_2	K_2	J_1	K_1	J_0	K_0	z_5	z_4	z_3	z_2	z_1	z_0
0	0	0	0	0	1	0	k	0	k	1	k	1	1	1	0	0	0
0	0	1	0	1	0	0	k	1	k	k	1	0	1	1	1	0	0
0	1	0	0	1	1	0	k	k	0	1	k	0	0	1	1	1	0
0	1	1	1	0	0	1	k	k	1	k	1	0	0	0	1	1	1
1	0	0	1	0	1	k	0	0	k	1	k	0	0	1	1	1	0
1	0	1	0	0	0	k	1	0	k	k	1	0	1	1	1	0	0

Flip-flop giriş Denklemlerinin Gerçeklenmesi

y_2 \ y_1y_0	00	01	11	10
0	0	0	0	1
1	k	k	k	k

$J_2 = y_1y_0'$

y_2 \ y_1y_0	00	01	11	10
0	k	k	k	k
1	0	1	k	k

$K_2 = y_0$

y_2 \ y_1y_0	00	01	11	10
0	0	1	k	k
1	0	0	k	k

$J_1 = y_2'y_0$

y_2 \ y_1y_0	00	01	11	10
0	k	k	1	0
1	k	k	k	k

$K_1 = y_0$

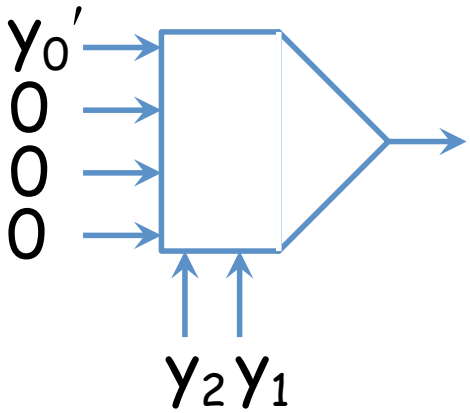
y_2 \ y_1y_0	00	01	11	10
0	1	k	k	1
1	1	k	k	k

$J_0 = 1$

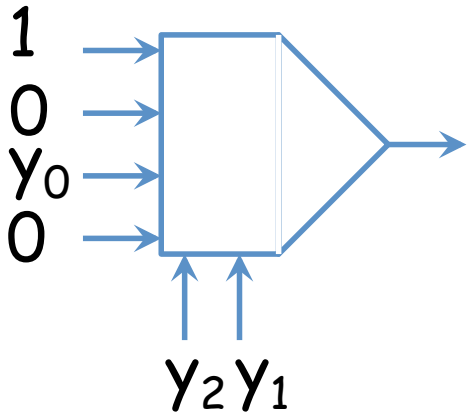
y_2 \ y_1y_0	00	01	11	10
0	k	1	1	k
1	k	1	k	k

$K_1 = 1$

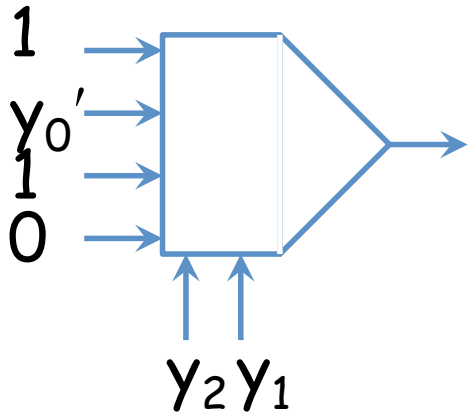
Flip-flop Çıkış Denklemlerinin Gerçeklenmesi



$z_5 = y_2'y_1'y_0' + k(y_2y_1y_0' + y_2y_1y_0)$

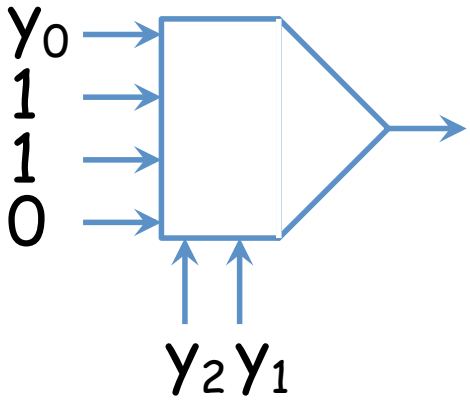


$z_4 = y_2'y_1'y_0' + y_2'y_1'y_0 + y_2y_1'y_0 + k(y_2y_1y_0' + y_2y_1y_0)$

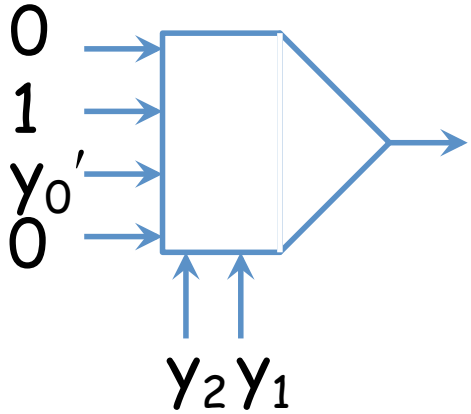


$z_3 = y_2'y_1'y_0' + y_2'y_1'y_0 + y_2'y_1y_0' + y_2y_1'y_0' + y_2y_1'y_0 + k(y_2y_1y_0' + y_2y_1y_0)$

Flip-flop Çıkış Denklemlerinin Gerçeklenmesi

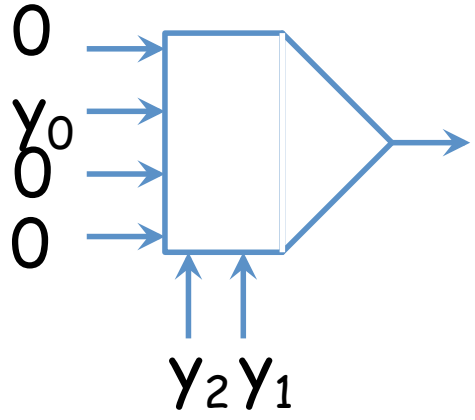


$$z_2=y_2'y_1'y_0+y_2'y_1y_0'+y_2'y_1y_0+y_2y_1'y_0'+y_2y_1'y_0+k(y_2y_1y_0'+y_2y_1y_0)$$



$$z_1=y_2'y_1y_0'+y_2'y_1y_0+y_2y_1'y_0'+k(y_2y_1y_0'+y_2y_1y_0)$$

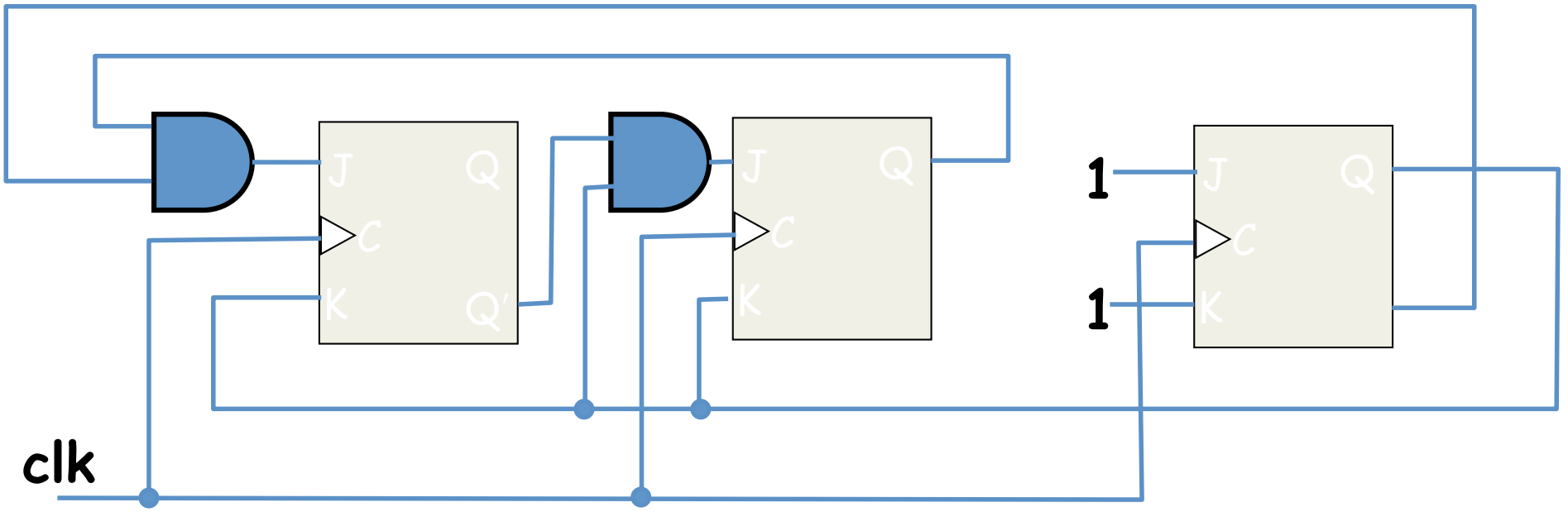
Flip-flop Çıkış Denklemlerinin Gerçeklenmesi



$$z_0 = y_2' y_1 y_0 + k(y_2 y_1 y_0' + y_2 y_1 y_0)$$

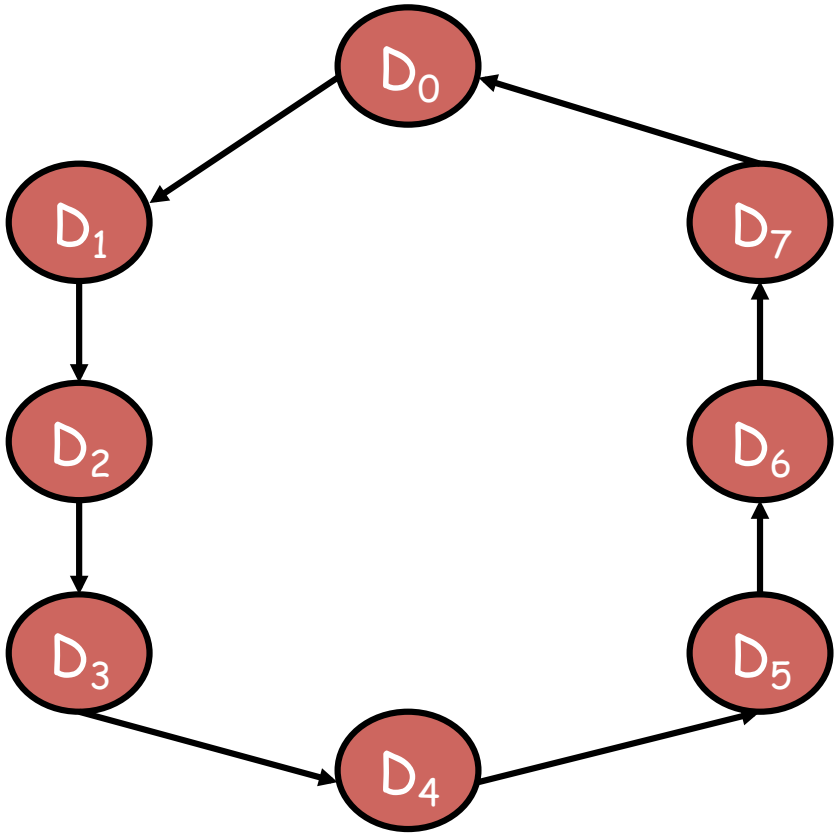
Lojik diyagram

$J_2 = y_1y_0'$ $K_2 = y_0$ $J_1 = y_2'y_0$ $K_1 = y_0$ $J_0 = 1$ $K_1 = 1$



T Tipi Flip-Floplar ile tasarım

- Örnek: 3-bit ikili sayıcı
  - 0→1→2→ … → 7 → 0 → 1 → 2



Durum Diyagramı

Kaç flip-flop?

Durum kodlama:

- D_0 → 000
- D_1 → 001
- D_2 → 010
- . . .
- D_7 → 111

T Tipi Flip-Floplar ile tasarım

- Durum tablosu

Şimdiki Durum			Sonraki Durum			FF girişleri		
y_2	y_1	y_0	Y_2	Y_1	Y_0	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

T Tipi Flip-Floplar ile tasarım

- Flip-Flop giriş denklemleri

y_2 \ y_1y_0	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$T_2 = y_1y_0$

y_2 \ y_1y_0	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$T_1 = y_0$

$T_0 = 1$

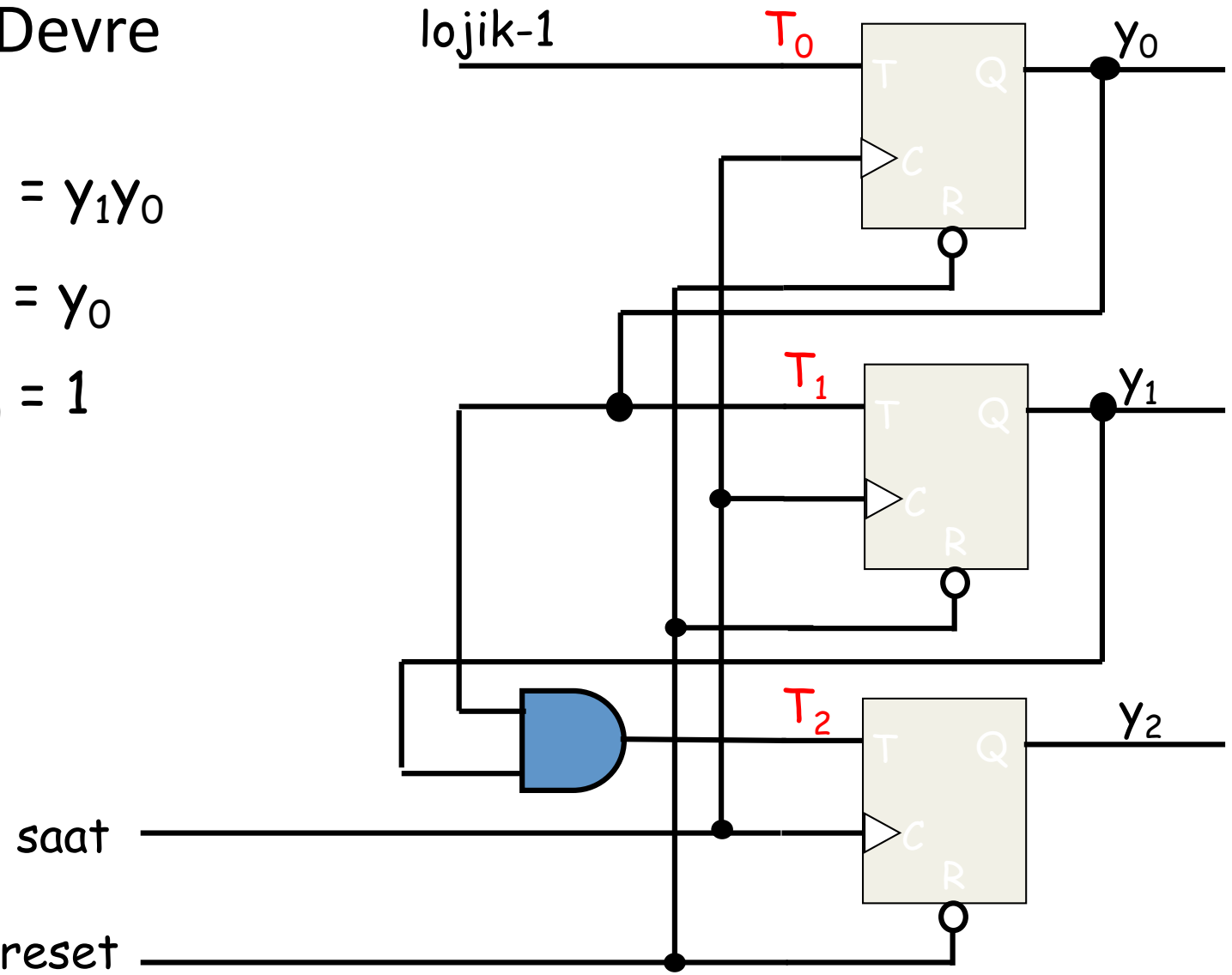
T Tipi Flip-Floplar ile tasarım

- Devre

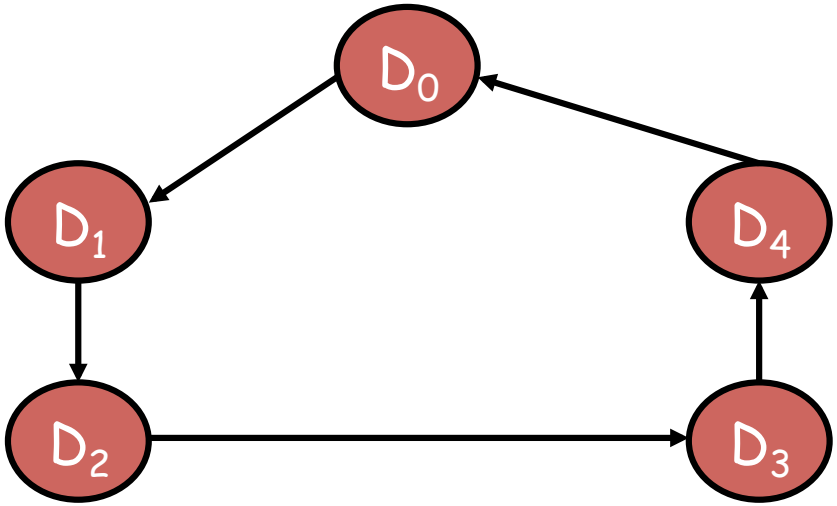
$T_2 = y_1 y_0$

$T_1 = y_0$

$T_0 = 1$



Kullanılmayan Durumlar



Modulo-5 sayıcı

Şimdiki Durum			Sonraki Durum		
y_2	y_1	y_0	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

Kullanılmayan Durumlar

Şimdiki Durum			Sonraki Durum		
y_2	y_1	y_0	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

y_2 \ y_1y_0	00	01	11	10
0	0	0	1	0
1	0	X	X	X

$$Y_2 = y_1y_0$$

y_2 \ y_1y_0	00	01	11	10
0	0	1	0	1
1	0	X	X	X

$$Y_1 = y_1' y_0 + y_1y_0'$$
$$= y_1 \oplus y_0$$

y_2 \ y_1y_0	00	01	11	10
0	1	0	0	1
1	0	X	X	X

$$Y_0 = y_2' y_0'$$

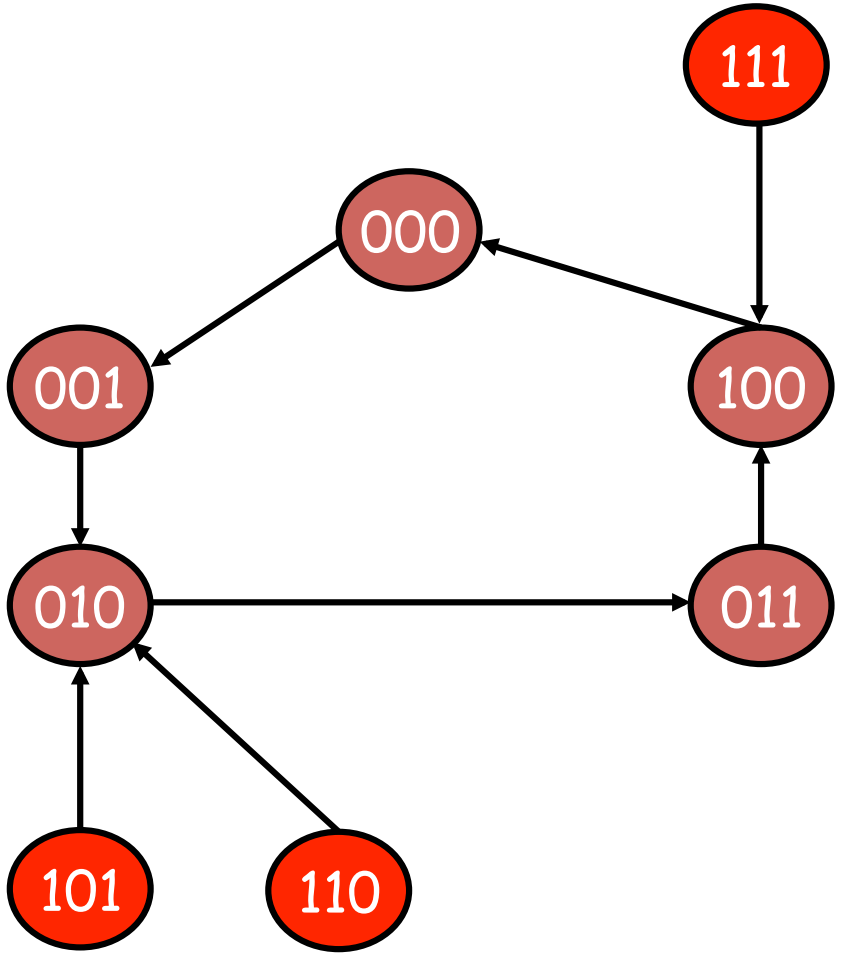
Kullanılmayan Durumlar

Şimdiki Durum			Sonraki Durum		
y_2	y_1	y_0	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	0	0

$Y_2 = y_1 y_0$

$Y_1 = y_1 \oplus y_0$

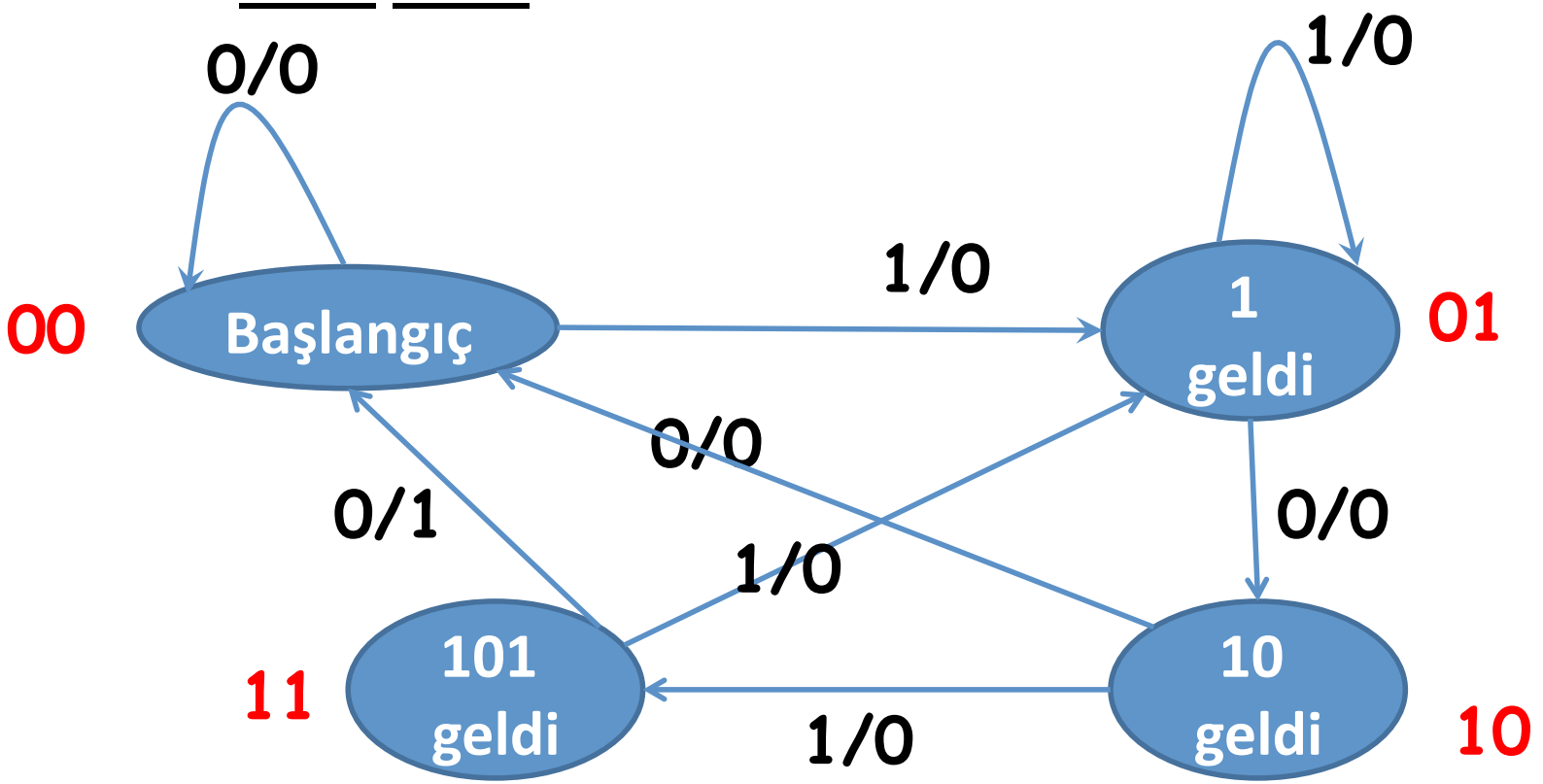
$Y_0 = y_2' y_0'$



Devre kilitlenen türden değil.

Tasarım Örneği

- 1 bitlik girişinden son 1010 geldiğinde çıkışı 1 olan devreyi tasarlayınız.
- Örnek: x= 1010 1011 ise z= 0001 0000



Meally makinası

Durum Tablosu

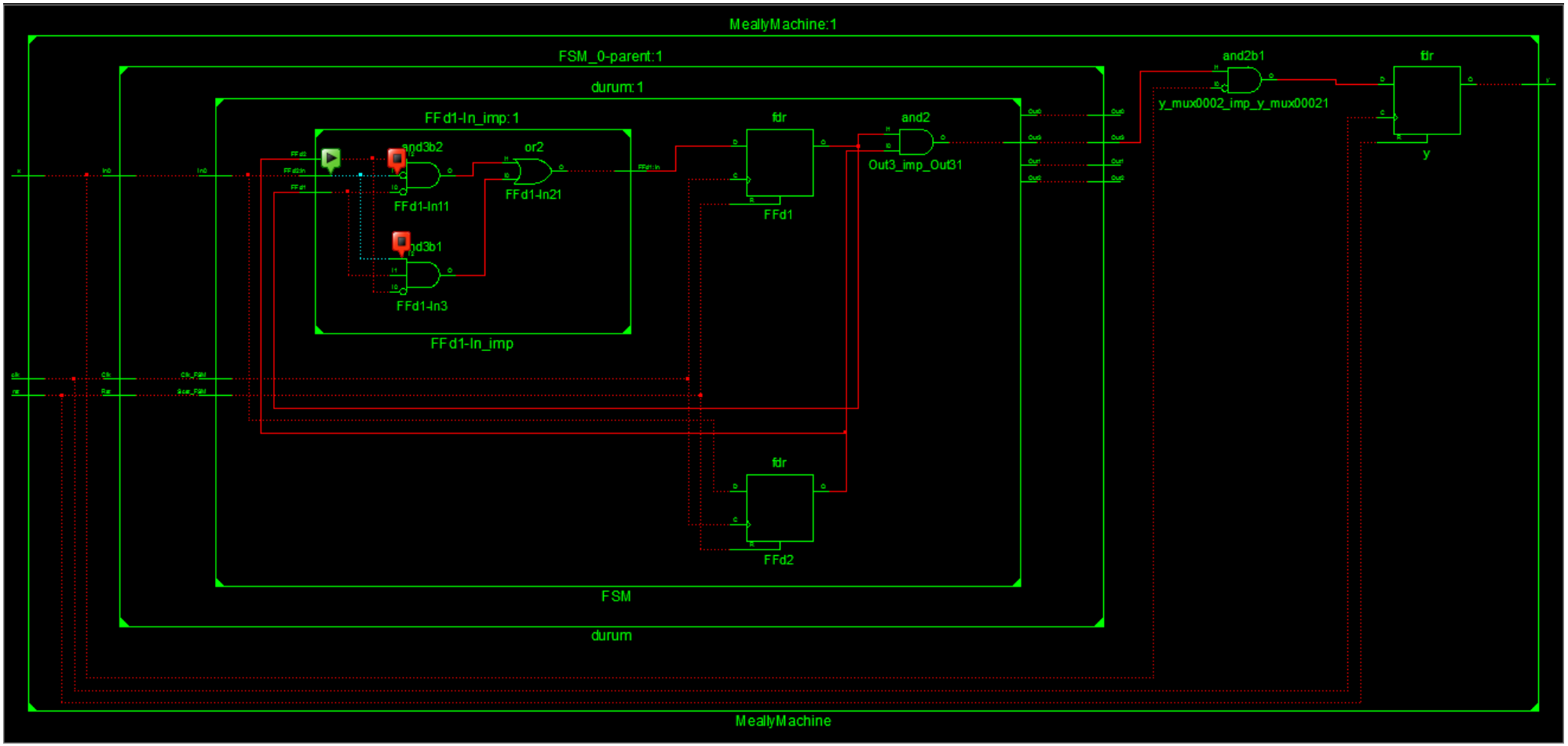
Şimdiki Durum		Giriş	Sonraki Durum		Çıkış
y_1	y_2	x	Y_1	Y_2	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	0	1	0

VHDL Kodu

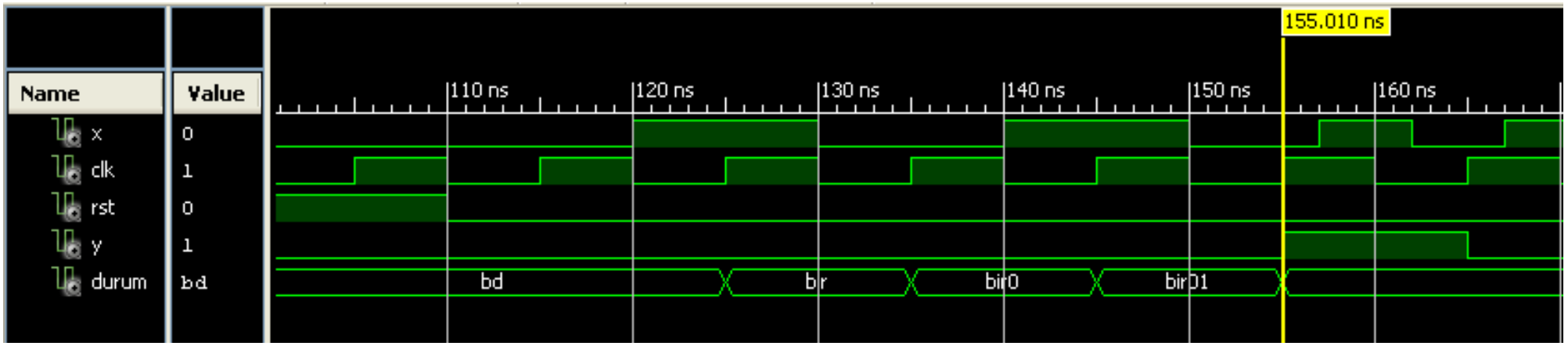
```
library IEEE;
use
      IEEE.STD_LOGIC_1164
      .ALL;

entity MeallyMachine is
   Port ( x : in  STD_LOGIC;
          y : out  STD_LOGIC;
          clk : in  STD_LOGIC;
          rst : in  STD_LOGIC);
end MeallyMachine;
```

```
architecture Behavioral of MeallyMachine is
   type state_type is (BD,Bir,Bir0,Bir01); signal durum : state_type;
Begin
process(clk) begin
   if clk'event and clk='1' then
      if rst='1' then  durum <= BD; y <= '0';
      else
         case durum is
            when BD =>
               y <= '0';
               if x='1' then durum <= Bir;  else durum <= BD;  end if;
            when Bir=>
               y <= '0';
               if x='1' then durum <= Bir;  else  durum <= Bir0;  end if;
            when Bir0 =>
               y <= '0';
               if x='1' then durum <= Bir01;  else  durum <= BD;  end if;
            when Bir01=>
               if x='1' then durum <= Bir; y <= '1‘; else durum <= Bir0; y <= '0‘; end if;
         end case;
      end if;
      end if;
end process;
end Behavioral;
```

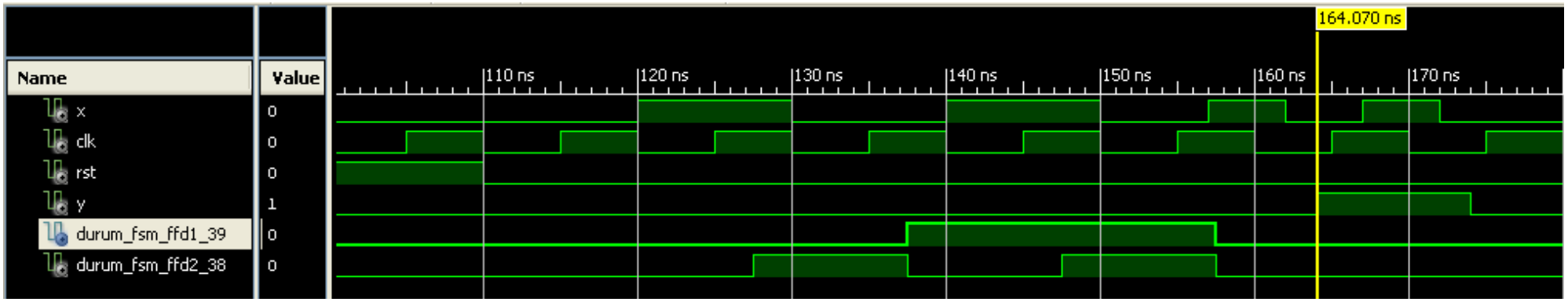
Devre Şeması



Zamanlama Diyagramı

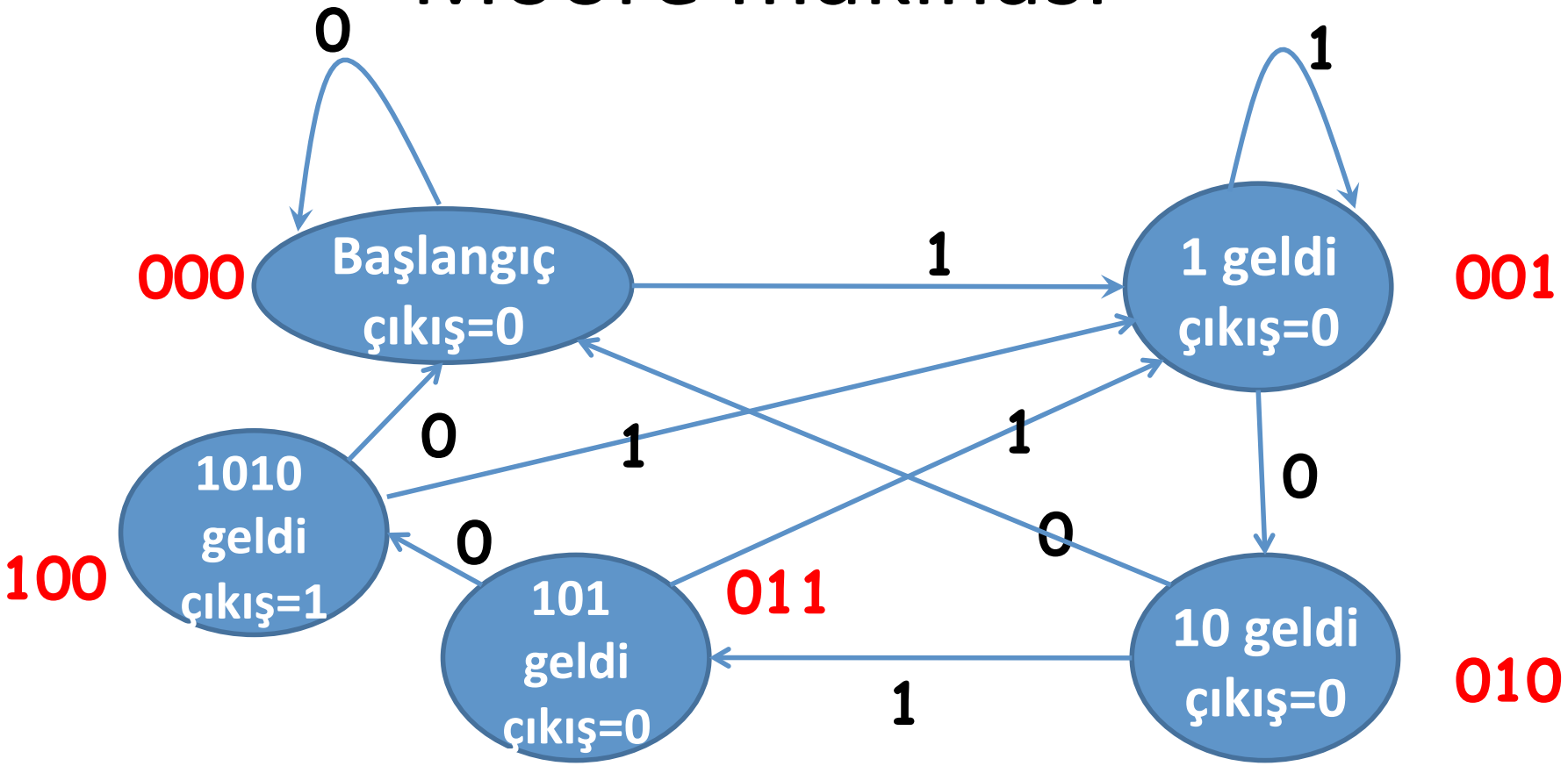


İdeal Hal: Gecikmeler = 0



İdeal Olmayan Hal: Gecikmeler ≠ 0

Moore makinası

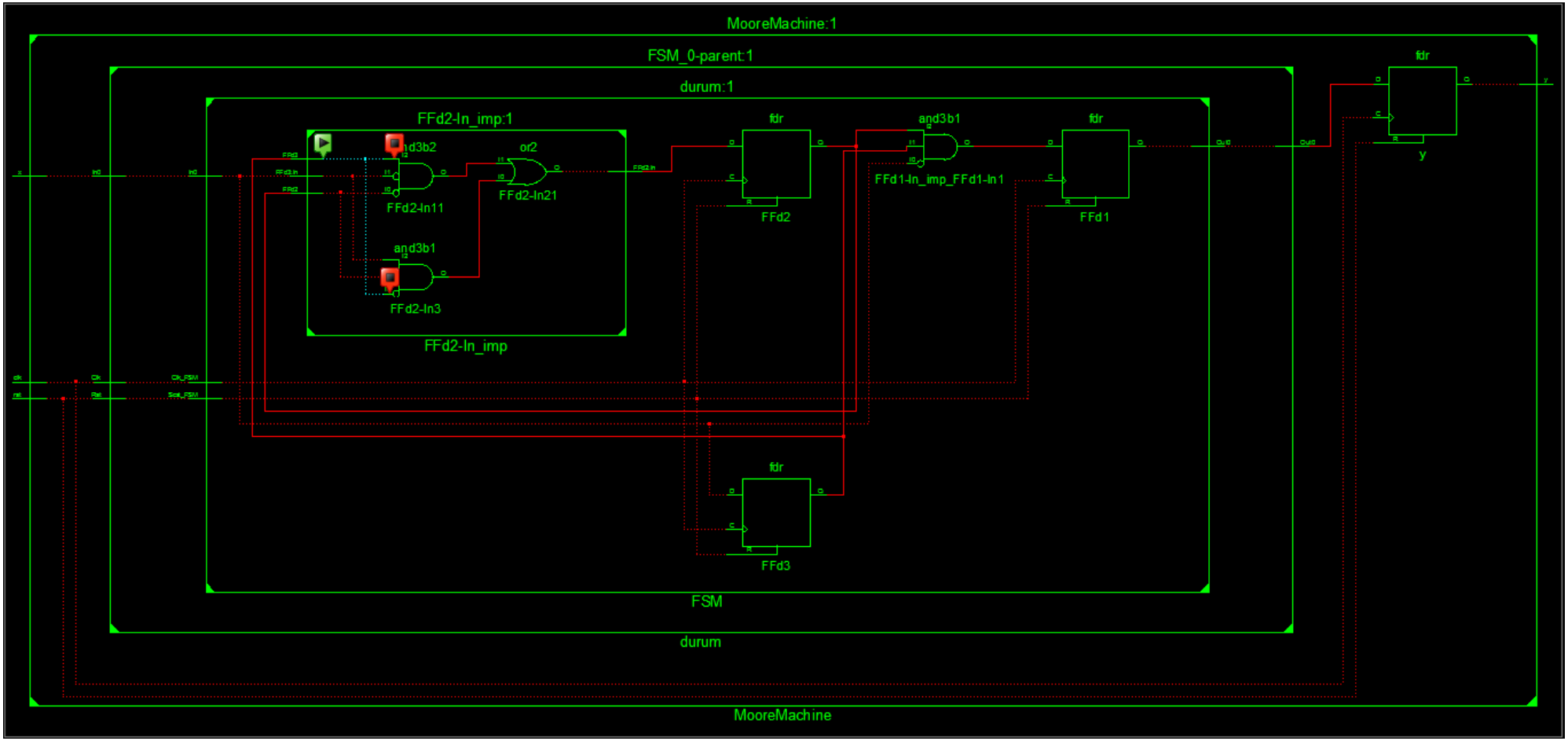


VHDL Kodu

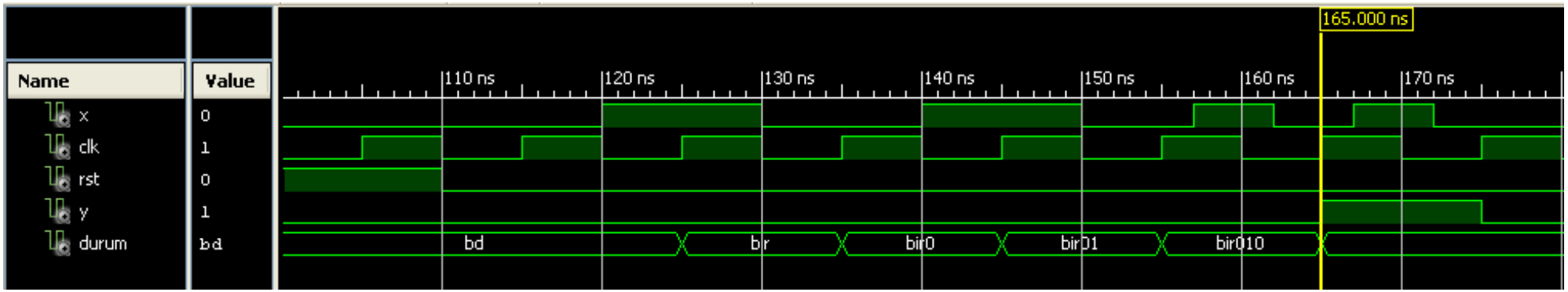
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MooreMachine is
    Port ( x : in  STD_LOGIC;
           y : out  STD_LOGIC;
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC);
end MooreMachine;
```

```
architecture Behavioral of MooreMachine is
type state_type is (BD,Bir,Bir0,Bir01,Bir010); signal durum : state_type;
Begin
process(clk) begin
    if clk'event and clk='1' then
        if rst='1' then durum <= BD; y <= '0';
        else
            case durum is
                when BD =>
                    y <= '0';
                    if x='1' then durum <= Bir; else durum <= BD; end if;
                when Bir =>
                    y <= '0';
                    if x='1' then durum <= Bir; else durum <= Bir0; end if;
                when Bir0 =>
                    y <= '0';
                    if x='1' then durum <= Bir01; else durum <= BD; end if;
                when Bir01 =>
                    y <= '0';
                    if x='1' then durum <= Bir; else durum <= Bir010; end if;
                when Bir010 =>
                    y <= '1';
                    if x='1' then durum <= Bir; else durum <= BD; end if;
        end case;
     end if;
  end if;
end process;
end Behavioral;
```

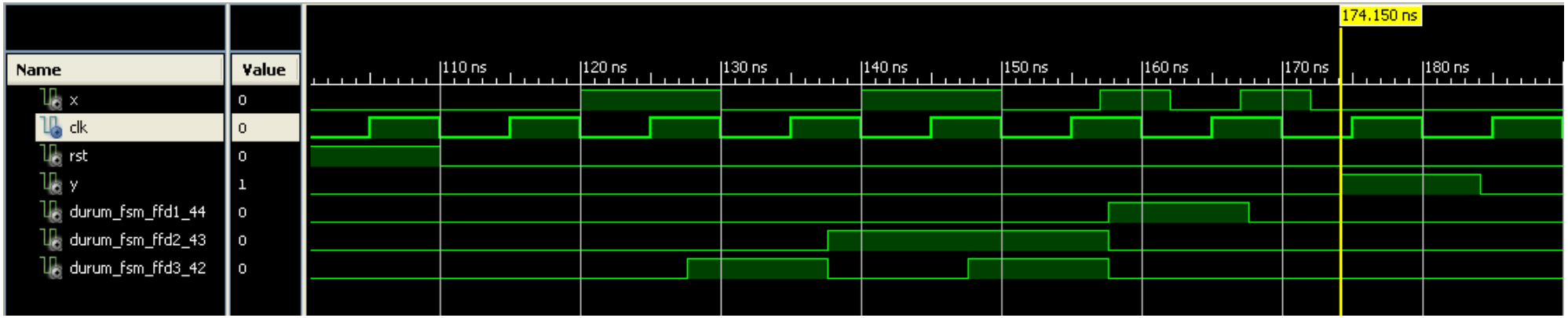
Devre Şeması



Zamanlama Diyagramı



İdeal Hal: Gecikmeler = 0



İdeal Olmayan Hal: Gecikmeler ≠ 0