

# BİLGİSAYAR ORGANİZASYONU ve TASARIMI

YRD. DOÇ. DR. FATİH KELEŞ

- ▶ Aritmetik Mikroişlemleri
- ▶ Lojik Mikroişlemleri
- ▶ Öteleme Mikroişlemleri
- ▶ Aritmetik Lojik Kaydırma Birimi

# Mikroişlemeler

- ▶ Saklayıcılar içinde depolanan bilgi ile gerçekleştirilen en basit ve temel işlemlere **Mikroişlem** denir.

Bilgisayar Sistem Mikroişlemleri 4 farklı sınıfa ayrılabilir:

- Saklayıcı Transfer Mikroişlemleri
- Aritmetik Mikroişlemleri
- Lojik Mikroişlemleri
- Kaydırma (Öteleme) Mikroişlemleri

# Mikroişlemler

- ▶ Sayısal bilgisayarlarda sıkça görülen mikro işlemler 4 kategoride sınıflandırılır.
  - Saklayıcı aktarım mikroişlemleri, ikili sisteme göre hazırlanmış bilgiyi bir saklayıcıdan diğerine aktarır.
  - Aritmetik mikro işlem, saklayıcılarda saklanan sayısal bilgilerin aritmetik işlemlerini gerçekleştirir.
  - Lojik (mantıksal) mikro işlemler, saklayıcılarda saklanan sayısal olmayan bilgi ile ilgili bit kullanım işlemlerini gerçekleştirir.
  - Kaydırma (öteleme) mikro işlemleri, saklayıcılarda saklanan veriler üzerinde kaydırma işlemlerini gerçekleştirir.

# Aritmetik Mikroişlemleri

## Temel Aritmetik Mikroişlemler

- Toplama (Addition)
  - Çıkarma (Subtraction)
  - Artırma (Increment)
  - Azaltma (Decrement)
- 
- **İlave Aritmetik Mikroişlemler**
  - Elde ile toplama (Add with carry)
  - Borç ile çıkarma (Subtract with borrow)
  - Aktarma/Yükleme (Transfer/Load)
  - v.b. gibi...

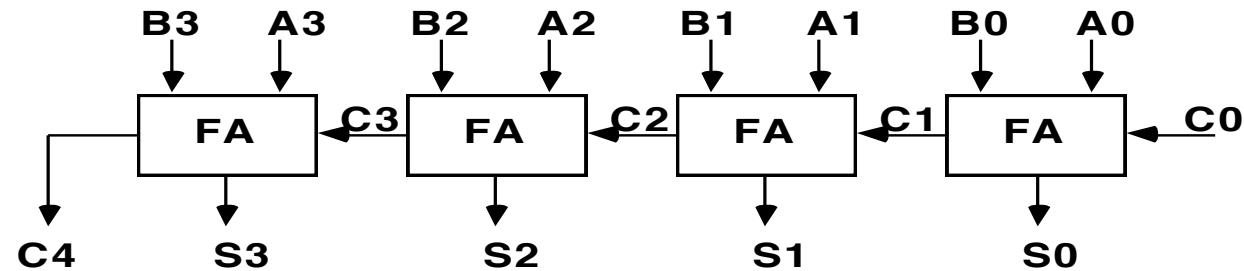
# Aritmetik Mikroişlemleri

## Tipik Aritmetik Mikroişlemler Özeti :

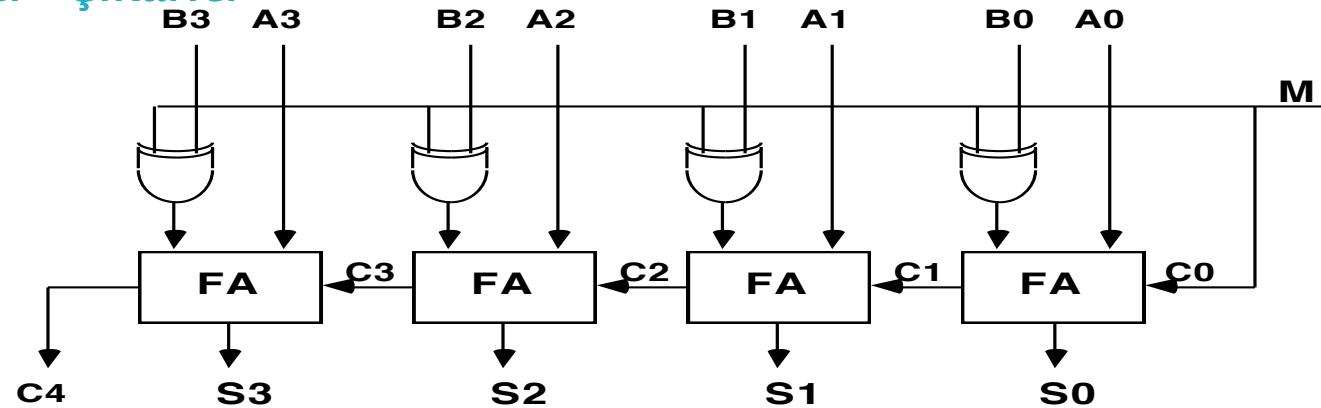
$R3 \leftarrow R1 + R2$	Contents of R1 plus R2 transferred to R3
$R3 \leftarrow R1 - R2$	Contents of R1 minus R2 transferred to R3
$R2 \leftarrow R2'$	Complement the contents of R2
$R2 \leftarrow R2' + 1$	2's complement the contents of R2 (negate)
$R3 \leftarrow R1 + R2' + 1$	Subtraction
$R1 \leftarrow R1 + 1$	Increment
$R1 \leftarrow R1 - 1$	Decrement

# İkili Toplayıcı / Çıkarıcı / Artırıcı

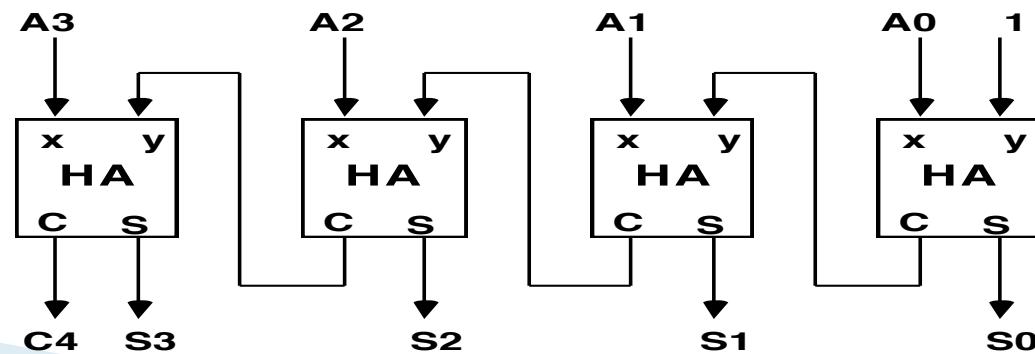
İkili Toplayıcı



İkili Toplayıcı- Çıkarıcı

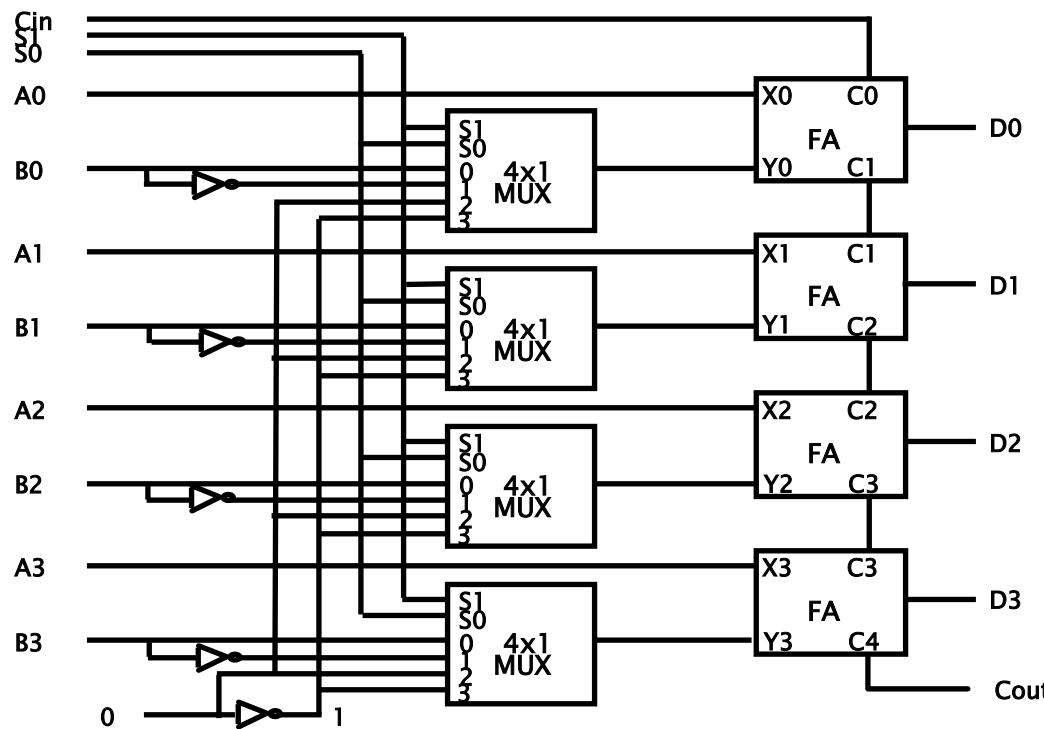


İkili Artırıcı  
(İkili Sayıcı ile  
de yapılabılır)



# Aritmetik Birim Tasarımı

4-bit aritmetik devre



$$D = A + Y + C_{in}$$

$$D_0 = A_0(X_0) + B_0(Y_0) + C_{in}$$

- Y >> B:  $S_0, S_1$  **B, B', 0, 1**

S1	S0	Cin	Y	Output	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	B'	$D = A + B'$	Subtract with borrow
0	1	1	B'	$D = A + B' + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

- $D = A + B' + 1 - 1 = A - B - 1$
- $D = A + B' + 1 = A - B$
- $D = A + 1111 = A - 1 \quad \{(0001)' + 1 = -1\}$
- $D = A - 1 + 1 = A$

# Lojik Mikroişlemeleri

- ▶ Saklayıcılardaki bit dizinleri üzerinde ikili işlemleri belirler.
  - Lojik mikroişlemler bit düzeyinde gerçekleşen işlemlerdir.
  - İkili data üzerinde bit manipasyonu için yararlıdır.
  - Bit değerine bağlı lojik karar verme durumu için kullanılır.
- ▶ İki değişkenle oluşturulabilecek 16 lojik fonksiyon tanımlanabilir.

Bununla birlikte, çoğu sistemler sadece bunlardan 4 tanesini gerçekler. Bunlar:

- AND ( $\wedge$ ), OR ( $\vee$ ), XOR ( $\oplus$ ), Complement/NOT
- ▶ Diğer fonksiyonlar bunlar cinsinden tanımlanabilirler.

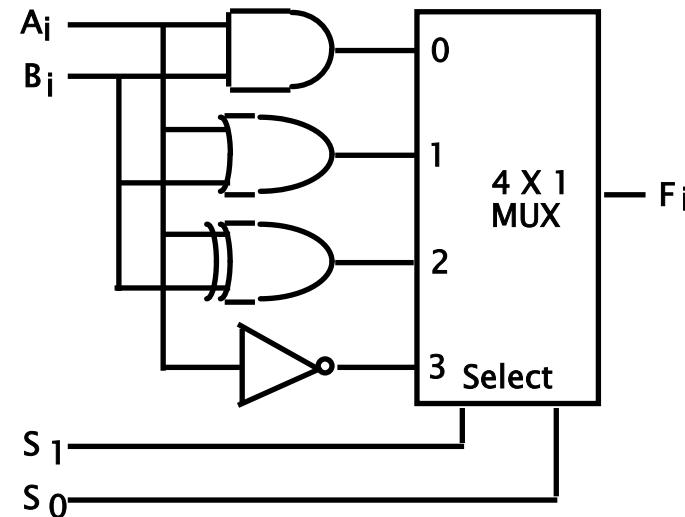
A	B	$F_0$	$F_1$	$F_2$	...	$F_{13}$	$F_{14}$	$F_{15}$
0	0	0	0	0	...	1	1	1
0	1	0	0	0	...	1	1	1
1	0	0	0	1	...	0	1	1
1	1	0	1	0	...	1	0	1

# Lojik Mikroişlemlerin Listesi

2 değişkenli 16 farklı lojik fonksiyon ve onlara karşı gelen 16 lojik mikroişlem listesi

<i>x</i>	0 0 1 1	<i>Boolean Function</i>	<i>Micro-Operations</i>	<i>Name</i>
<i>y</i>	0 1 0 1			
	0 0 0 0	$F_0 = 0$	$F \leftarrow 0$	Clear
	0 0 0 1	$F_1 = xy$	$F \leftarrow A \wedge B$	AND
	0 0 1 0	$F_2 = x'y'$	$F \leftarrow A \wedge B'$	
	0 0 1 1	$F_3 = x$	$F \leftarrow A$	Transfer A
	0 1 0 0	$F_4 = x'y$	$F \leftarrow A' \wedge B$	
	0 1 0 1	$F_5 = y$	$F \leftarrow B$	Transfer B
	0 1 1 0	$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
	0 1 1 1	$F_7 = x + y$	$F \leftarrow A \vee B$	OR
	1 0 0 0	$F_8 = (x + y)'$	$F \leftarrow (A \vee B)'$	NOR
	1 0 0 1	$F_9 = (x \oplus y)'$	$F \leftarrow (A \oplus B)'$	Exclusive-NOR
	1 0 1 0	$F_{10} = y'$	$F \leftarrow B'$	Complement B
	1 0 1 1	$F_{11} = x + y'$	$F \leftarrow A \vee B'$	
	1 1 0 0	$F_{12} = x'$	$F \leftarrow A'$	Complement A
	1 1 0 1	$F_{13} = x' + y$	$F \leftarrow A' \vee B$	
	1 1 1 0	$F_{14} = (xy)'$	$F \leftarrow (A \wedge B)'$	NAND
	1 1 1 1	$F_{15} = 1$	$F \leftarrow \text{all } 1's$	Set to all 1's

# Lojik Mikroişlemelerin Donanımsal Gerçeklemesi



Fonksiyon Tablosu

$S_1$	$S_0$	Output	$\mu$ -operation
0	0	$F = A \wedge B$	AND
0	1	$F = A \vee B$	OR
1	0	$F = A \oplus B$	XOR
1	1	$F = A'$	Complement

# Lojik Mikroişlem Uygulamaları

- ▶ Lojik mikroişlemler bir saklayıcının **mevcut bitlerini** veya **bir kısmını değiştirmek** (manüple etmek) amacıyla kullanılabilir. Aynı zamanda **bir grup biti silmeyi** ya da bir saklayıcının içine **yeni bir bit eklemeyi** sağlayabilir.
- ▶ Kabul: Register A bitleri üzerinde işlem yapılıyor.  
Register B ise register A'nın içeriğini değiştirmek üzere kullanılabilecek datayı tutsun.

◦ Seçici Birleme	Selective-set	$A \leftarrow A + B$
◦ Seçici Tümleme	Selective-complement	$A \leftarrow A \oplus B$
◦ Seçici Silme	Selective-clear	$A \leftarrow A \bullet B'$
◦ Maskeleme	Mask (Delete)	$A \leftarrow A \bullet B$
◦ Silme	Clear	$A \leftarrow A \oplus B$
◦ Yerleştirme	Insert	$A \leftarrow (A \bullet B) + C$
◦ ...		

# Seçici Birleme İşlemi

## OR

- Register A daki belirli bitleri birlemek için, register B bit paterni kullanılmaktadır.

$$\begin{array}{r} \textcolor{red}{1} \textcolor{magenta}{0} \textcolor{black}{1} \textcolor{black}{0} \\ \textcolor{red}{1} \textcolor{red}{1} \textcolor{black}{0} \textcolor{black}{0} \\ \hline \textcolor{red}{1} \textcolor{magenta}{1} \textcolor{black}{1} \textcolor{black}{0} \end{array} \quad \begin{array}{l} A_t \\ B \\ A_{t+1} \end{array} \quad (A \leftarrow A + B)$$

- B deki 1 olan bitler A da birlenecek (set) olan bit pozisyonlarını göstermektedir. B deki 0 olan bitler A da hiçbir etki yapmazlar.

# Seçici Tümleme İşlemi

## EX-OR

- ▶ Register A daki belirli bitleri tümlemek için, register B bit paterni kullanılmaktadır.

$$\begin{array}{r} \textcolor{red}{1} \textcolor{magenta}{0} \textcolor{red}{1} \textcolor{magenta}{0} \\ \underline{\textcolor{red}{1} \textcolor{magenta}{1} \textcolor{red}{0} \textcolor{magenta}{0}} \\ \textcolor{magenta}{0} \textcolor{red}{1} \textcolor{magenta}{1} \textcolor{red}{0} \end{array} \quad \begin{array}{l} A_t \\ B \\ A_{t+1} \quad (A \leftarrow A \oplus B) \end{array}$$

- ▶ B deki 1 olan bitler A da tümlenecek olan bit pozisyonlarını göstermektedir. B deki 0 olan bitler A da hiçbir etki yapmazlar.

# Seçici Silme İşlemi

## AND ve NOT

Register A daki belirli bitleri silmek (clear, reset) için, register B bit paterni kullanılmaktadır.

$$\begin{array}{r} 1 \textcolor{red}{0} \ 1 \ 0 \\ 1 \textcolor{red}{1} \ 0 \ 0 \\ \hline 0 \textcolor{red}{0} \ 1 \ 0 \end{array} \quad \begin{array}{l} A_t \\ B \\ A_{t+1} \quad (A \leftarrow A \cdot B') \end{array}$$

- ▶ B deki 1 olan bitler A da silinecek (clear) olan bit pozisyonlarını göstermektedir. B deki 0 olan bitler A da hiçbir etki yapmazlar.

# Maskleme İşlemi

## AND

Register A daki belirli bitleri masklemek (mask) için, register B bit paterni kullanılmaktadır.

$$\begin{array}{r} \textcolor{red}{1} \textcolor{blue}{0} \textcolor{red}{1} \textcolor{blue}{0} \\ \textcolor{red}{1} \textcolor{blue}{1} \textcolor{red}{0} \textcolor{blue}{0} \\ \hline \textcolor{red}{1} \textcolor{blue}{0} \textcolor{red}{0} \textcolor{blue}{0} \end{array} \quad \begin{array}{l} A_t \\ B \\ A_{t+1} \quad (A \leftarrow A \cdot B) \end{array}$$

- ▶ B deki 0 olan bitler A da maskelenecek olan bit pozisyonlarını göstermektedir. B deki 1 olan bitler A da hiçbir etki yapmazlar.

# İçerik Silme İşlemi

## EX-OR

Register A daki tüm bitleri silmek (clear, reset) için, register B bit paterni kullanılmaktadır.

Register B, register A içeriğine sahip olmalıdır.

$$\begin{array}{r} 1010 \\ \underline{1010} \\ 0000 \end{array} \quad \begin{array}{l} A_t \\ B \\ A_{t+1} \end{array} \quad (A \leftarrow A \oplus B)$$

# Yerleştirme (Insert) İşlemi

## AND - OR

- ▶ Register A içeriğinin bir kısmı yerine yeni bir data yerleştirilmesidir.
- ▶ Bu işlem 2 aşamada gerçekleşir:
  - 1. aşama **maskleme** ile A da yeni data yerleştirilmek istenen bit pozisyonları silinir. (AND)
  - 2. aşama **yerleştirme** işlemi bu silinen pozisyonlara istenilen yeni data konulur. (OR)

$$\begin{array}{r} \textcolor{red}{0110} \quad 1010 \quad A_t \\ \underline{\textcolor{red}{0000} \quad 1111} \quad B \\ 0000 \quad 1010 \quad A_{t+1} \end{array}$$

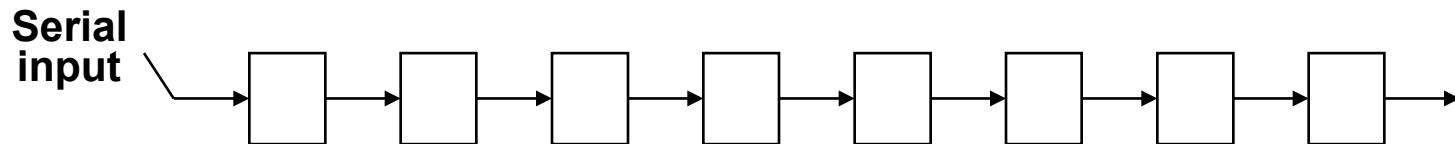
$(A \leftarrow A \cdot B)$

$$\begin{array}{r} 0000 \quad 1010 \quad A_t \\ \underline{\textcolor{red}{1001} \quad 0000} \quad B \\ \textcolor{red}{1001} \quad 1010 \quad A_{t+1} \end{array}$$

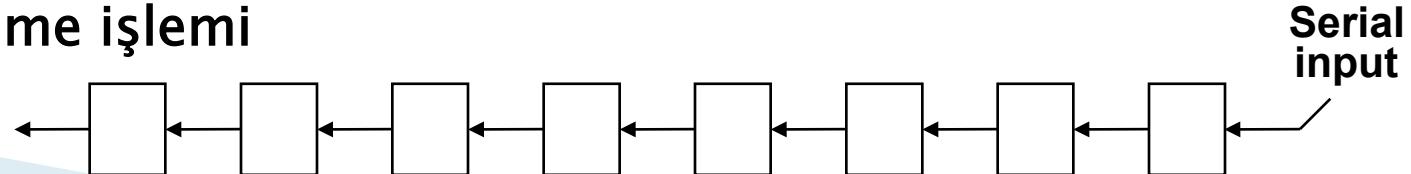
$(A \leftarrow A + B)$

# Öteleme Mikroişlemeleri

- ▶ 3 farklı öteleme tipi bulunmaktadır:
  - *Lojik öteleme*
  - *Dairesel öteleme*
  - *Aritmetik öteleme*
- ▶ Öteleme türlerinin aralarındaki temel fark seri giriş bilgisinin farklılığından kaynaklanır.
- Sağa Öteleme işlemi

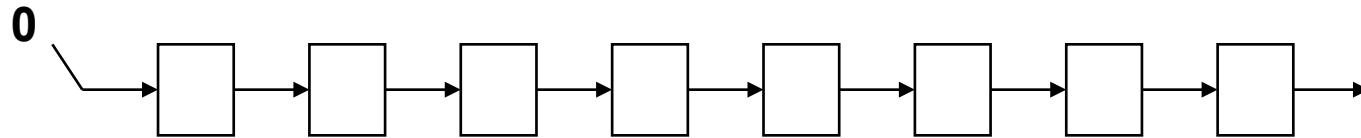


- Sola öteleme işlemi

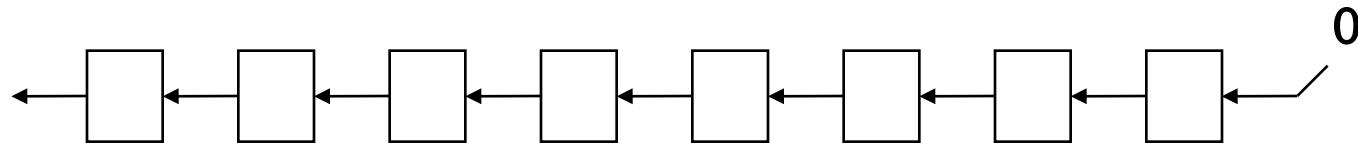


# Lojik Öteleme

- ▶ Lojik ötelemede seri giriş 0 olur.
- ▶ Sağa lojik öteleme işlemi,



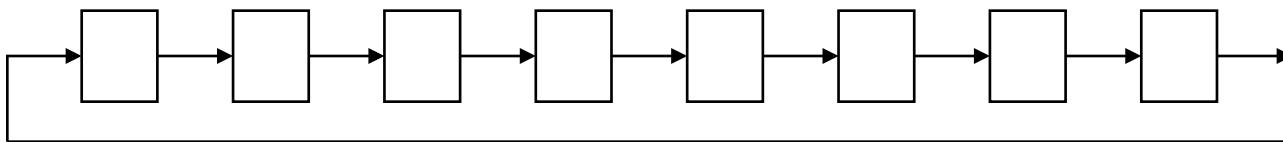
- ▶ Sola lojik öteleme işlemi,:



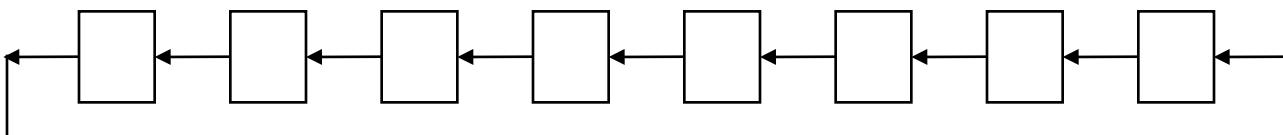
- ▶ *RTL de aşağıdaki notasyon kullanılır:*
- ▶ **shl** Sola lojik öteleme
- ▶ **shr** Sağa lojik öteleme
  - Örnekler:
    - **R2  $\leftarrow shr R2$**
    - **R3  $\leftarrow shl R3$**

# Dairesel Öteleme

- ▶ Seri giriş, saklayıcının diğer ucundan dışarı ötelenen bitdir.
- ▶ Dairesel sağa öteleme işlemi:



- ▶ Dairesel sola öteleme işlemi:



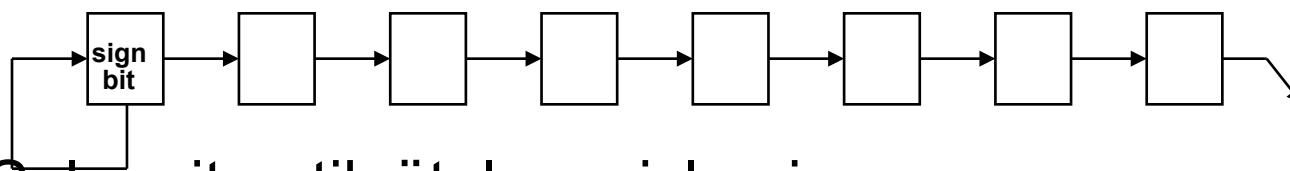
- ▶ RTL de, aşağıdaki notasyon kullanılır:

- *cil* Dairesel sola öteleme
- *cir* Dairesel sağa öteleme
- Örnekler:

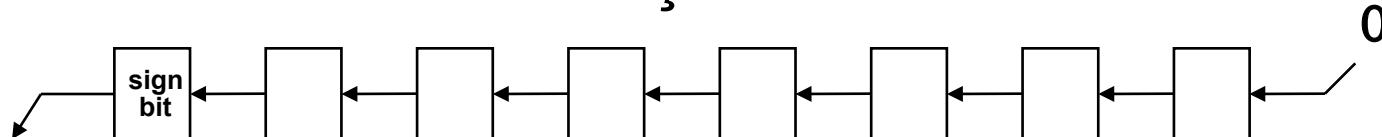
- **R2  $\leftarrow \text{cir} R2$**
- **R3  $\leftarrow \text{cil} R3$**

# Aritmetik Öteleme

- ▶ İşaretli sayıların ötelenmesi için kullanılır (işaretli tamsayı)
- ▶ Aritmetik sola öteleme işaretli sayıyı 2 ile çarpmak demektir.
- ▶ Aritmetik sağa öteleme işaretli sayıyı 2 ye bölmek demektir.  
İşaretli sayılar 2 ile çarpıldığında veya bölündüğünde işaretleri değişimeyeceğinden işaret bitleri korunmalıdır.
- ▶ Sağa aritmetik öteleme işlemi:

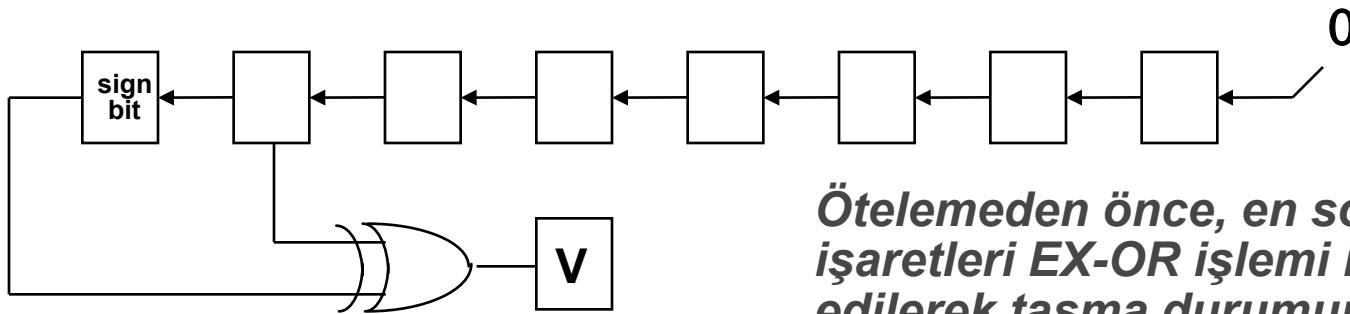


- ▶ Sola aritmetik öteleme işlemi:



# Aritmetik Öteleme

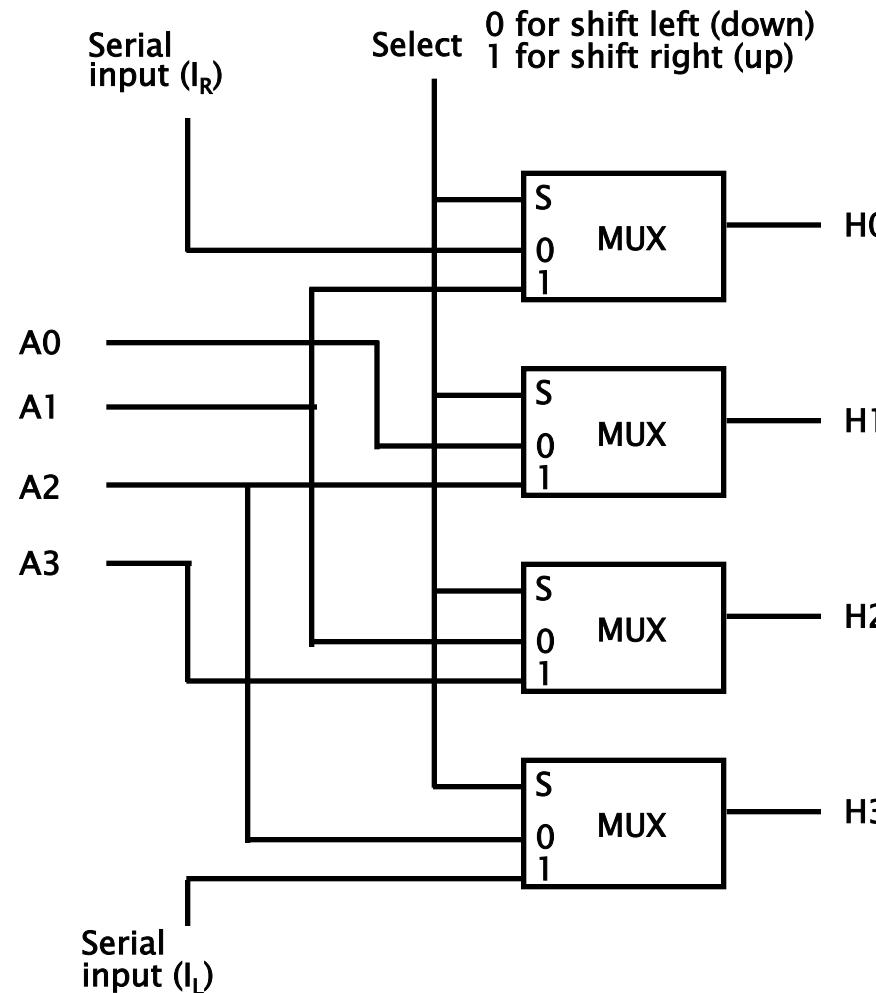
- Sola aritmetik öteleme işleminde sayı aralığının dışına taşıma olabileceğinden ( işaret değişimi) **taşma kontrolu** yapılmalıdır:



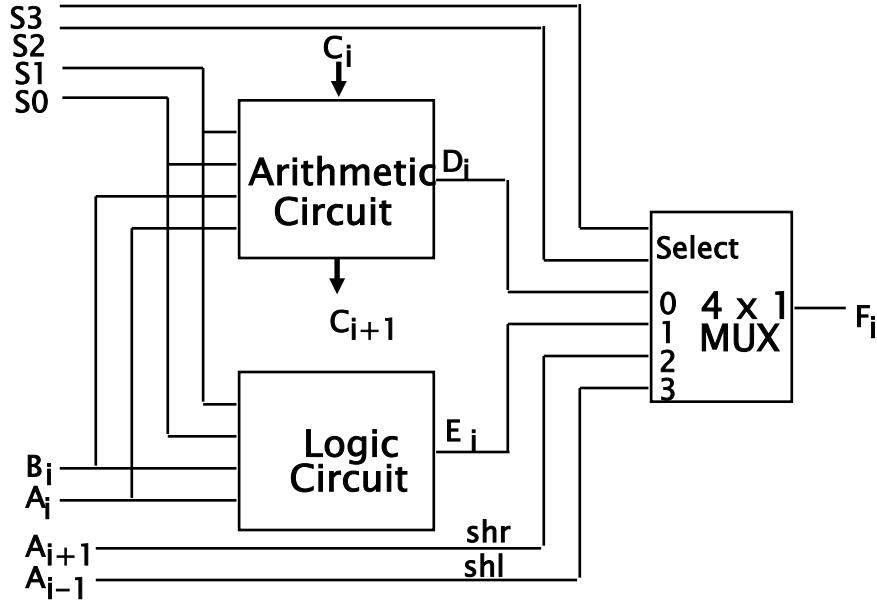
*Ötelemeden önce, en soldaki 2 bitin işaretleri EX-OR işlemi ile kontrol edilerek taşıma durumuna karar verilir.*

- RTL de, aşağıdaki gösterim kullanılır:
  - *ashl* Sola aritmetik öteleme
  - *ashr* Sağa aritmetik öteleme
  - Örnekler:
    - $R2 \leftarrow ashr R2$
    - $R3 \leftarrow ashl R3$

# Öteleme Mikroişlemelerinin Donanımsal Gerçeklemesi



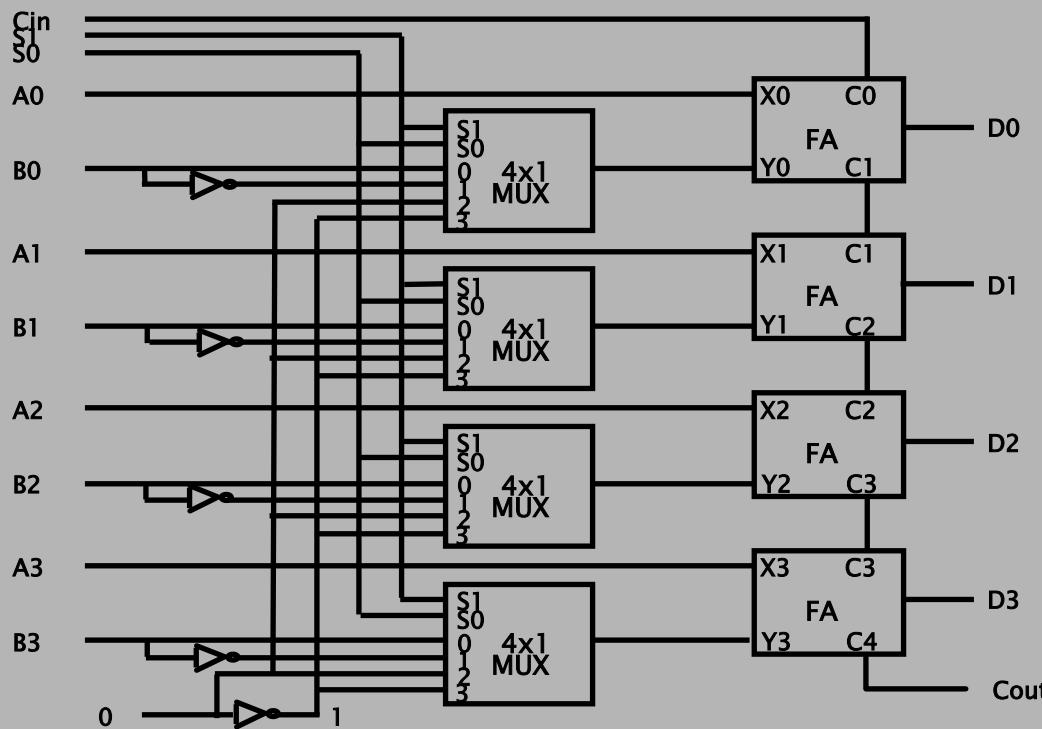
# Aritmetik Lojik Kaydırma Birimi



$S_3$	$S_2$	$S_1$	$S_0$	$Cin$	Operation	Function
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + B'$	Subtract with borrow
0	0	1	0	1	$F = A + B' + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	X	$F = A \wedge B$	AND
0	1	0	1	X	$F = A \vee B$	OR
0	1	1	0	X	$F = A \oplus B$	XOR
0	1	1	1	X	$F = A'$	Complement A
1	0	X	X	X	$F = shr\ A$	Shift right A into F
1	1	X	X	X	$F = shl\ A$	Shift left A into F

# Aritmetik Birim Tasarımı

4-bit aritmetik devre



$$D = A + Y + C_{in}$$

$$D_0 = A_0(X_0) + B_0(Y_0) + C_{in}$$

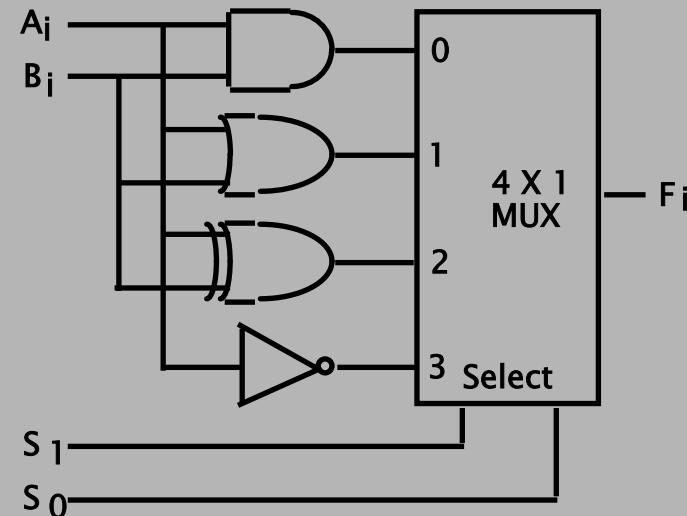
- Y >> B:  $S_0, S_1$  **B, B', 0, 1**

S1	S0	Cin	Y	Output	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	B'	$D = A + B'$	Subtract with borrow
0	1	1	B'	$D = A + B' + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

- $D = A + B' + 1 - 1 = A - B - 1$
- $D = A + B' + 1 = A - B$

- $D = A + 1111 = A - 1 \quad \{(0001)' + 1 = -1\}$
- $D = A - 1 + 1 = A$

# Lojik Mikroişlemelerin Donanımsal Gerçeklemesi



Fonksiyon Tablosu

$S_1$	$S_0$	Output	$\mu$ -operation
0	0	$F = A \wedge B$	AND
0	1	$F = A \vee B$	OR
1	0	$F = A \oplus B$	XOR
1	1	$F = A'$	Complement