# BİLGİSAYAR MİMARİSİ ve ORGANİZASYONU

## 9. BÖLÜM

# MIPS islemcilerde 32-bit isaretli sayılar






$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{(2)} = 0_{(10)}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{(2)} = +1_{(10)}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{(2)} = +2_{(10)}$

.....

.....

$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{(2)} = +2{,}147{,}483{,}646_{(10)}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{(2)} = +2{,}147{,}483{,}647_{(10)}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{(2)} = -2{,}147{,}483{,}648_{(10)}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{(2)} = -2{,}147{,}483{,}647_{(10)}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{(2)} = -2{,}147{,}483{,}646_{(10)}$

.....

.....

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{(2)} = -3_{(10)}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{(2)} = -2_{(10)}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{(2)} = -1_{(10)}$

# Sayının Negatifini Almak



MK



- Bir sayının negatifi, 2'ye tümleyenine esittir.
  - 2'ye tümleme = 0'lar 1'e, 1'ler 0'a çevrilir ve sayıya 1 eklenir
  - 0000 0010 -> 1111 1010

# Toplama-Çıkarma

- Toplama sağdan sola doğru bit-bit yapılır
- Eldeler de sağdan sola iletilir
- Çıkarma da toplamayı kullanır
- Çıkanın negatif değer karşılığı alınarak toplama, çıkarmaya karşılık gelir

```
  0000.....00 0110 =6
+ 0000.....00 0101 =5
-------------------
  0000.....00 1011 =11
```



# Toplama-Çıkarma






```
   0000 ..... 00 1111 =15
-  0000 .....00 0111 = 7
-------------------------
   0000 ..... 00 1000 = 8
```

```
   0000 ..... 00 1111 = 15
+ 1111 ..... 11 1001 =-7
-------------------------
   0000 .....00 1000 = 8
```

# Taşma (Overflow)

- Aritmetik işlemlerin sonucu maksimum bit uzunluğunu asarsa taşma olur.
- 32bitlik MIPS işlemcilerde işaretli sayılar için taşma işlemi **+2,147,483,647$_{(10)}$** veya **–2,147,483,648$_{(10)}$** değerlerinin asılması ile gerçekleşir.

```
  0111 ..... 11 1111  =  2 147 483 647
+ 0000..... 00 0010  =              2
-------------------------------------
  1000 ..... 00 0001  =  -2147 483 647
```

- Gerçek sonuç 2 147 483 649 olmalı idi
- Fakat bu sayı ancak 33-bit işaretli sayı olarak gösterilebilir.



# Taşma ne zaman meydana gelir

- Bir negatif sayı ile pozitif sayının toplanmasında taşma olmaz.
- İşaretleri aynı olan iki sayının çıkartılması ile taşma olmaz.
- Taşma, işlem sonucunun işareti değiştirmesi ile meydana gelir.
  - İki pozitif sayının toplanması ile sonucun negatif çıkması durumu
  - İki negatif sayının toplanması ile sonucun pozitif çıkması durumu
  - Pozitif sayıdan negatif sayının çıkarılması ve sonucun negatif olması durumu
  - Negatif sayıdan pozitif sayının çıkarılması ve sonucun pozitif olması durumu
- MIPS'te taşma olayı istisnai durum oluşturur.

MK



# Çarpma İşlemi

- Çarpma zor aritmetik işlemlerden birisidir.
- Donanımsal olarak çarpma işlemini gerçekleştirmenin 2 temel yolu bulunmaktadır.
  - **Daha çok devre kullanarak hızlı çarpma işlemi gerçekleştirmek.**
  - **Daha az devre ile daha yavaş çarpma işlemi gerçekleştirmek**
- Günümüzde işaretli ve işaretsiz sayıların çarpılması için çeşitli yöntemler bulunmaktadır.
- Bu yöntemler hem donanımsal hem de yazılımsal olarak kullanılabilmektedir.



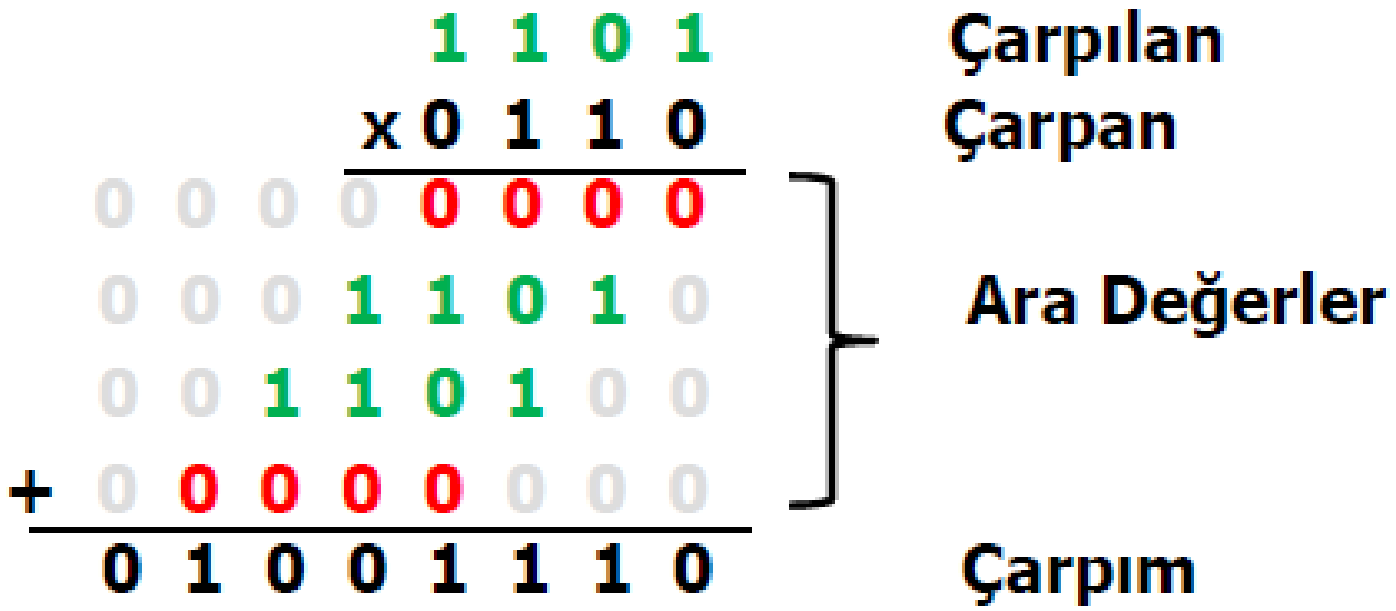
# Çarpma İşlemi



- İşaretsiz iki sayının çarpımı : 13 ×6 = 78

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1 | 1 | 0 | 1 | Çarpılan |
| | | × | 0 | 1 | 1 | 0 | Çarpan |
| | | | 0 | 0 | 0 | 0 | |
| | | 1 | 1 | 0 | 1 | | Ara değerler |
| | 1 | 1 | 0 | 1 | | | |
| +0 | 0 | 0 | 0 | | | | |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | Çarpım |

- Sadece 0 ve 1 değerleri ile çarpma yapıldığından ara değerler daima 0'a veya çarpılana (Bu örnek için 1101) eşittir.
- Ara değerlerin hepsi toplanarak çarpım elde edilir.
- İki tane n-bit sayı çarpılırsa çarpım, 2.n bit'e kadar çıkabilir.



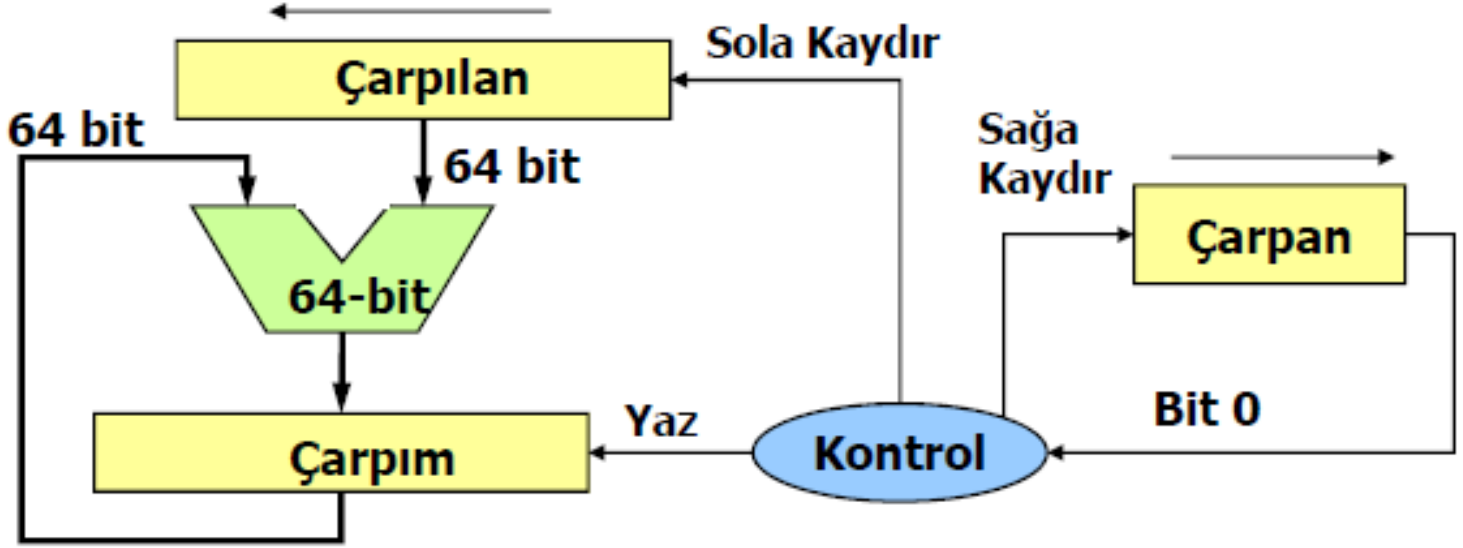



- Çarpma işlemi, ara değerlerin art arda elde edilmesi yardımıyla gerçekleştirilebilir.
- Çarpma işleminde her bir adım için **çarpılan** bir kaydırmalı kaydediciye kaydedilir ve **çarpan'ın** bitleri ile VE işlemine tabi tutulmadan bir kez sola kaydırılır.
- **Çarpan** da bir kaymalı kaydedicide tutularak her bir biti kolaylıkla elde edilebilir.

# Çarpma İşlemi (1.Yöntem)




- 32-bit'lik bir **çarpan** için çarpılan 64-bit'lik bir kaymalı kaydediciye kaydedilir.
- 32-bit'lik **çarpan** sağa kaydırılarak her bir adımda Bit 0'ın elde edilmesi sağlanır.
- 64-bit'lik toplayıcı, mevcut **çarpım** ile kaydırılmış **çarpılan** değerlerini toplar.

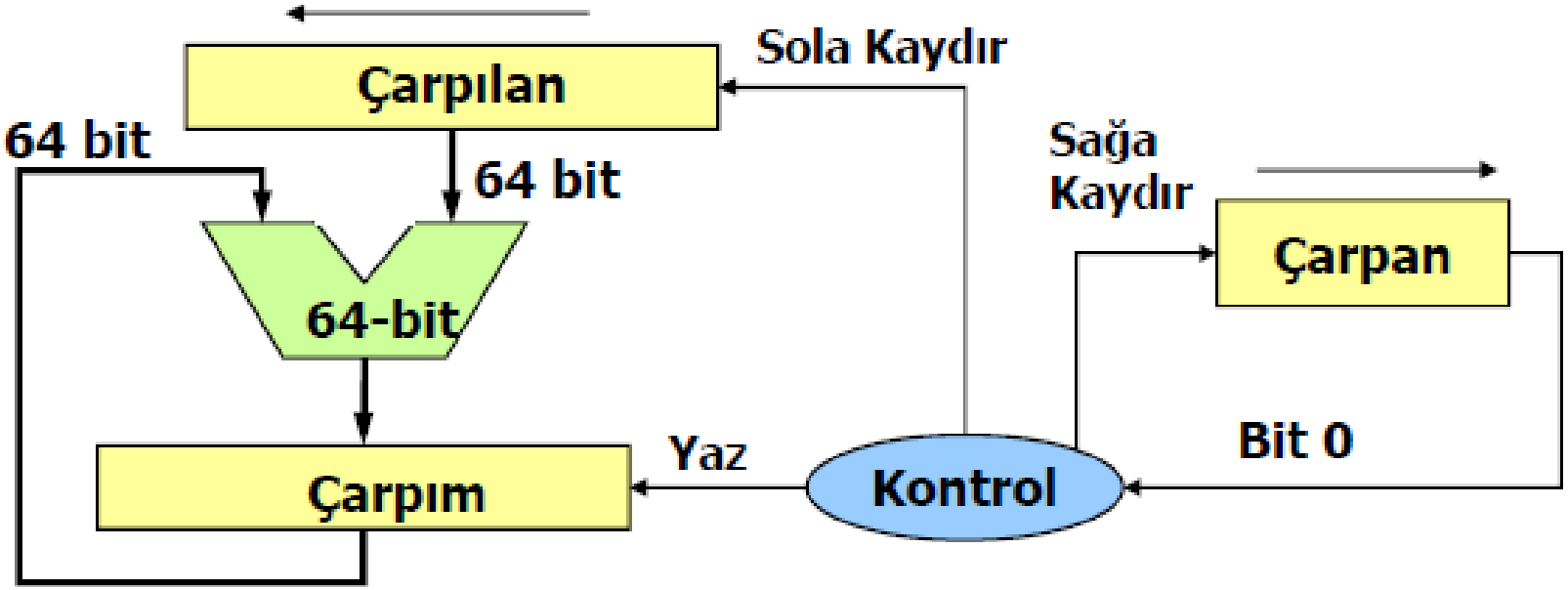



# Çarpma İşlemi (1.Yöntem)






- Çarpma başlamadan önce kaydediciler başlangıç değerlerine kurulur.
- Çarpılan kaydedicisinin düşük değerlikli 32 bit'lik kısmına **çarpılan** kaydedilir.
- Yüksek değerlikli 32 biti ise 0'a kurulur
- 32-bit'lik çarpan kaydedicisine de çarpan değeri yüklenir.
- 64-bit'lik çarpım kaydedicisi 0'a kurulur.



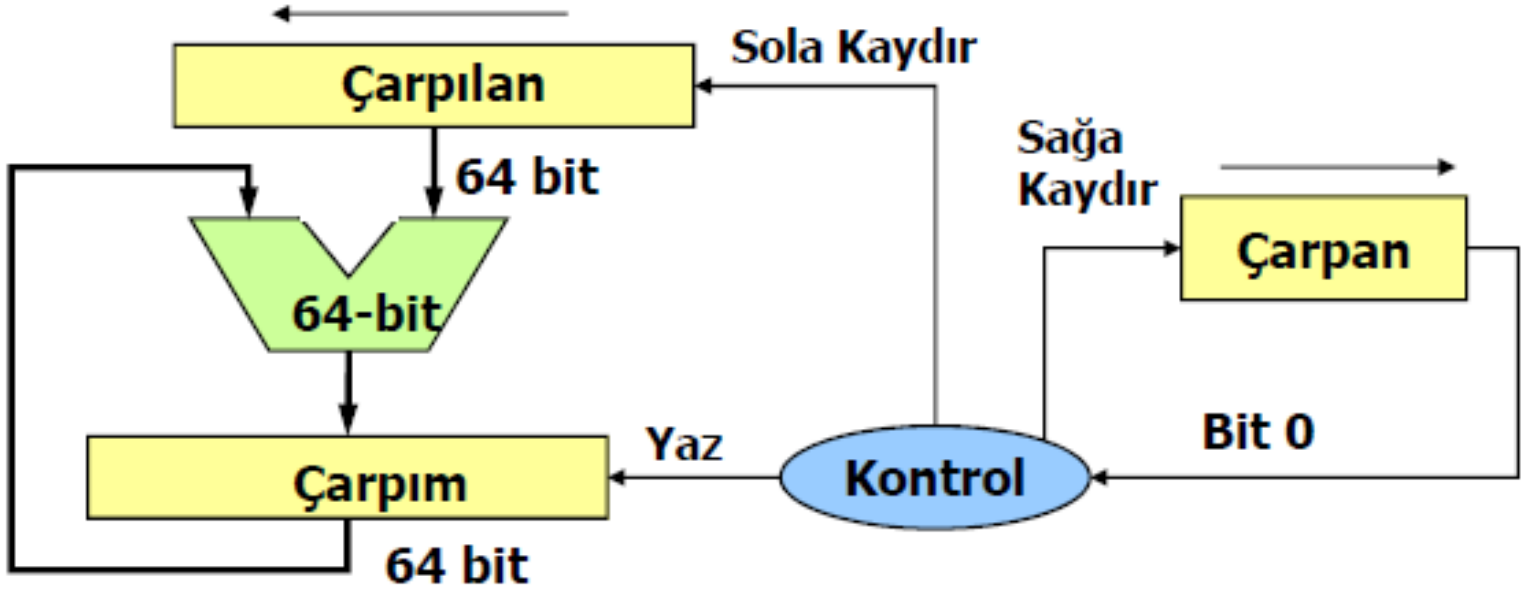

# Çarpma İşlemi (1.Yöntem)



MK



- Her bir adımda kontrol birimi çarpan kaydedicisinin Bit 0 değerini kontrol eder
  - Eğer Bit 0=0 ise ilgili ara çarpım değeri 0'a eşittir. Bu yüzden Çarpım kaydedicisine Yaz=0 sinyali gönderilerek işlem iptal edilir.
  - Eğer Bit 0=1 ise kaydırılmış çarpılan değerinin mevcut çarpıma eklenmesi gerekmektedir. Bu yüzden çarpım kaydedicisine Yaz=1 sinyali gönderilir.

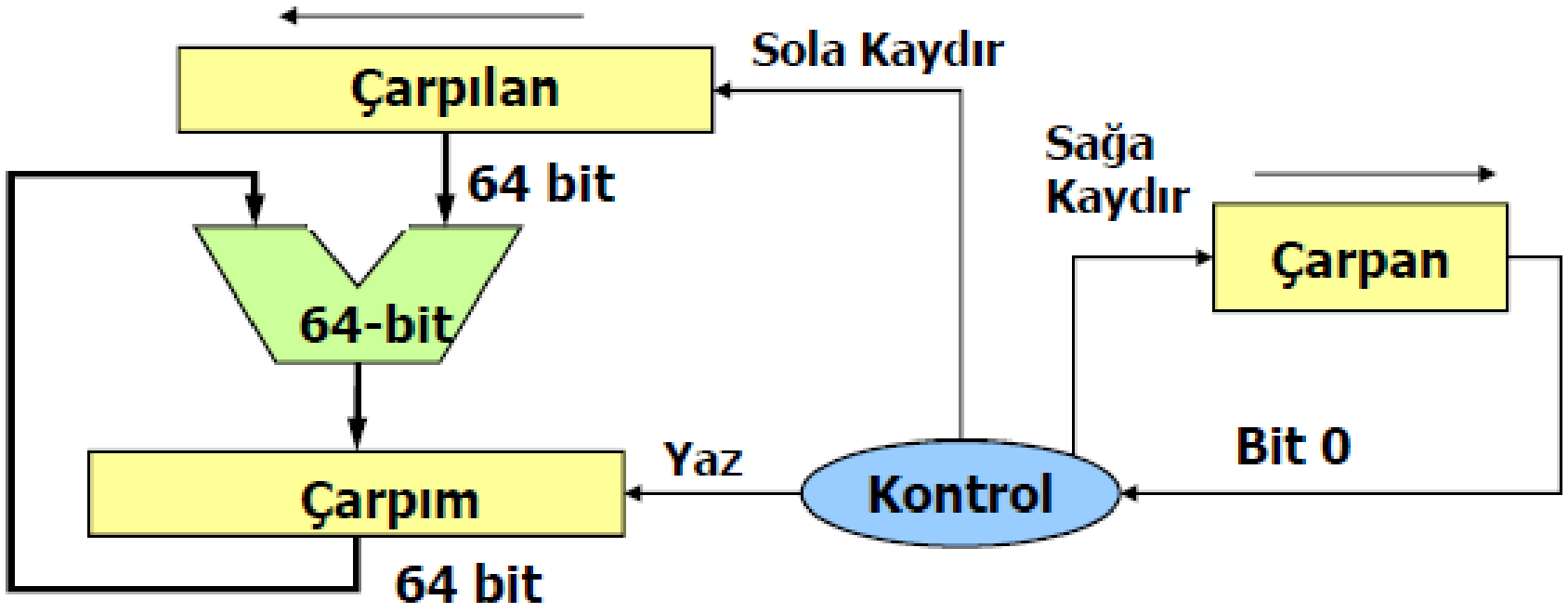



# Çarpma İşlemi (1.Yöntem)



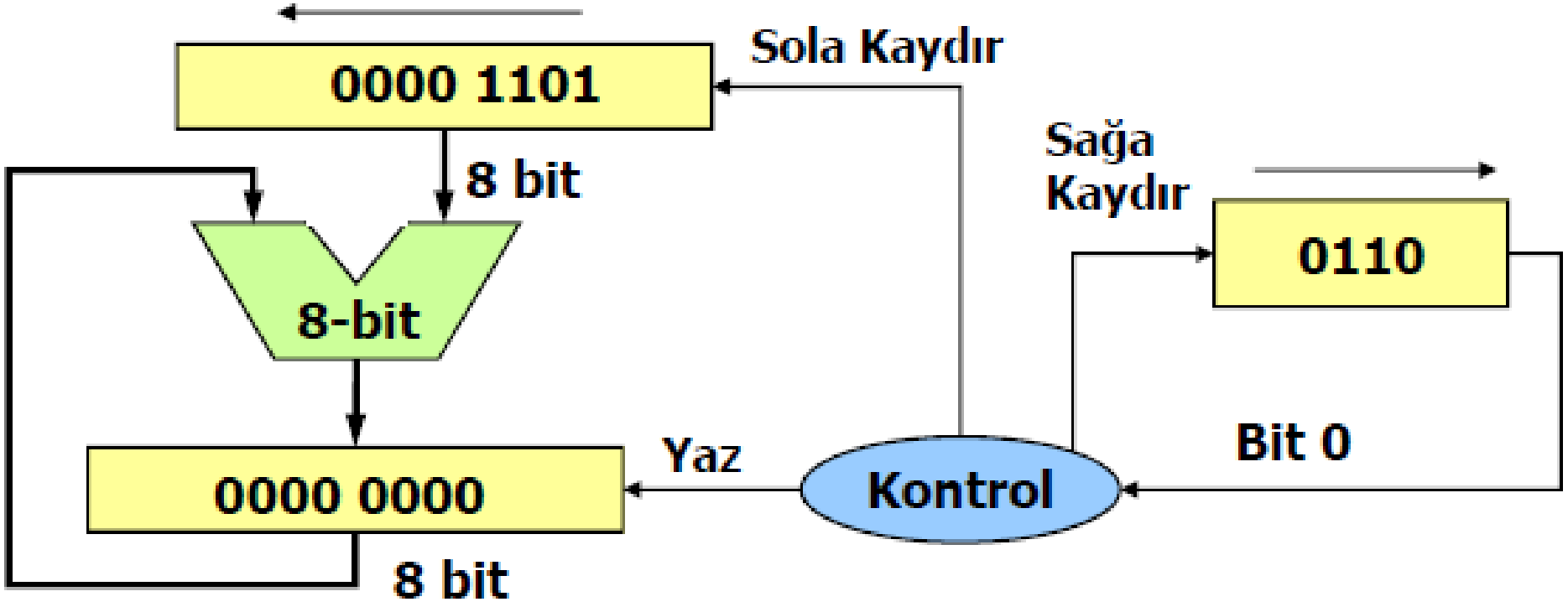



- 32 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.
- Çarpılanı bir bit sola kaydır
- Çarpanı bir bit sağa kaydır.




# 4-Bitlik Çarpma Örneği (1.Yöntem)

- 4-bitlik çarpma işlemi için bir adet 4-bitlik, 2 adet 8-bitlik kaydediciye ve 8-bitlik bir toplayıcı ihtiyaç vardır.
- 1101 değeri 0110 değeri ile çarpılmak istenmektedir.



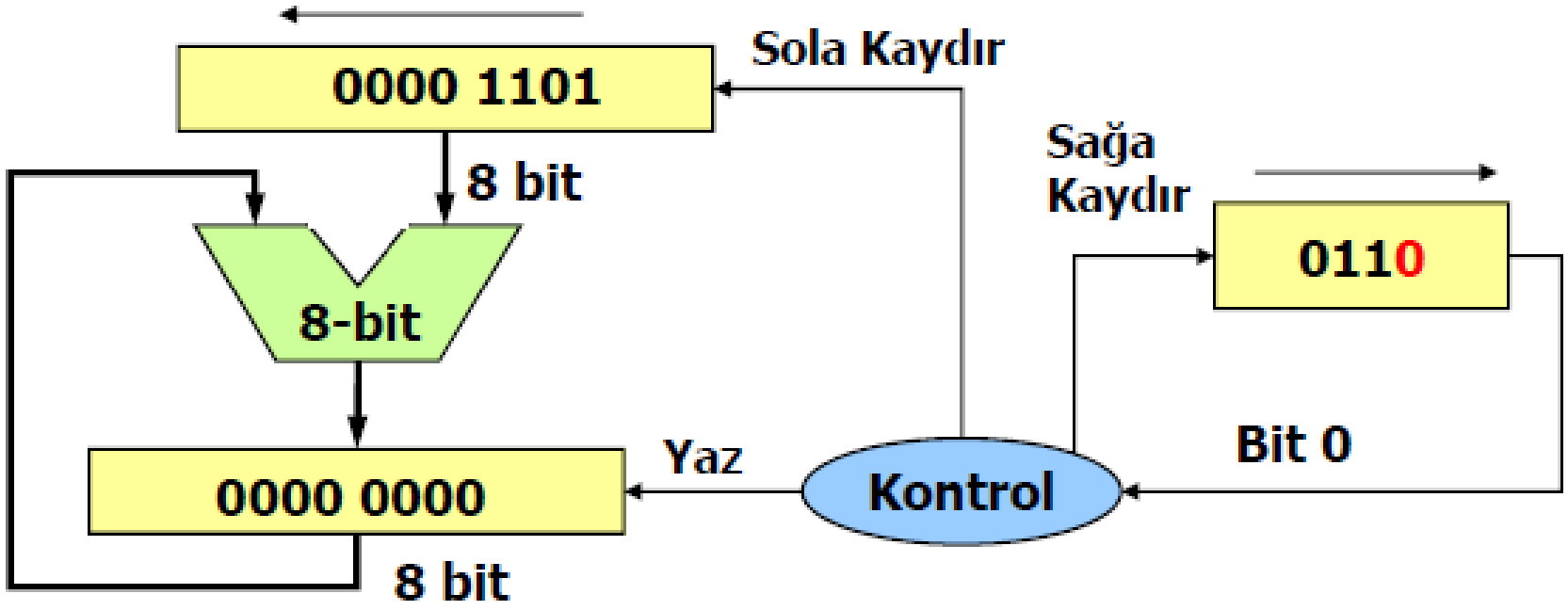



# 4-Bitlik Çarpma Örneği (1.Yöntem) 1a






- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.**
- Çarpılanı bir bit sola kaydır
- Çarpanı bir bit sağa kaydır.

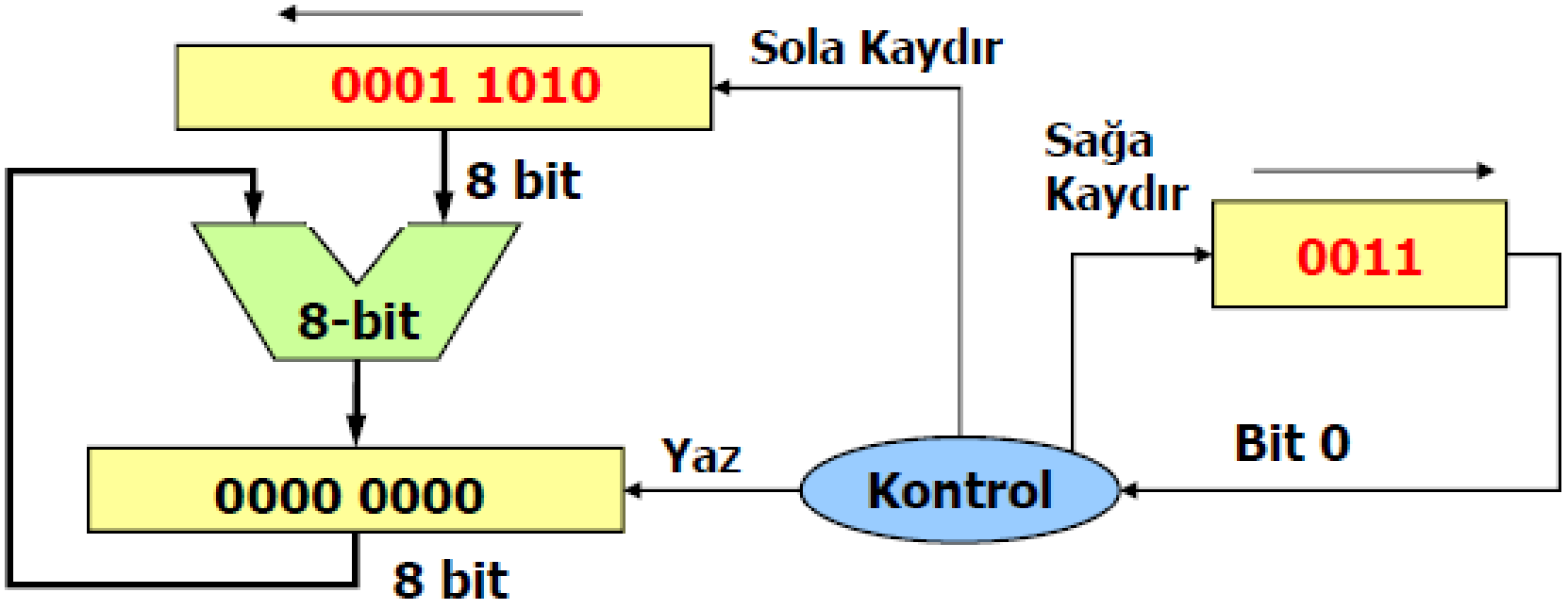



# 4-Bitlik Çarpma Örneği (1.Yöntem) 1b






- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.
- **Çarpılanı bir bit sola kaydır**
- **Çarpanı bir bit sağa kaydır.**

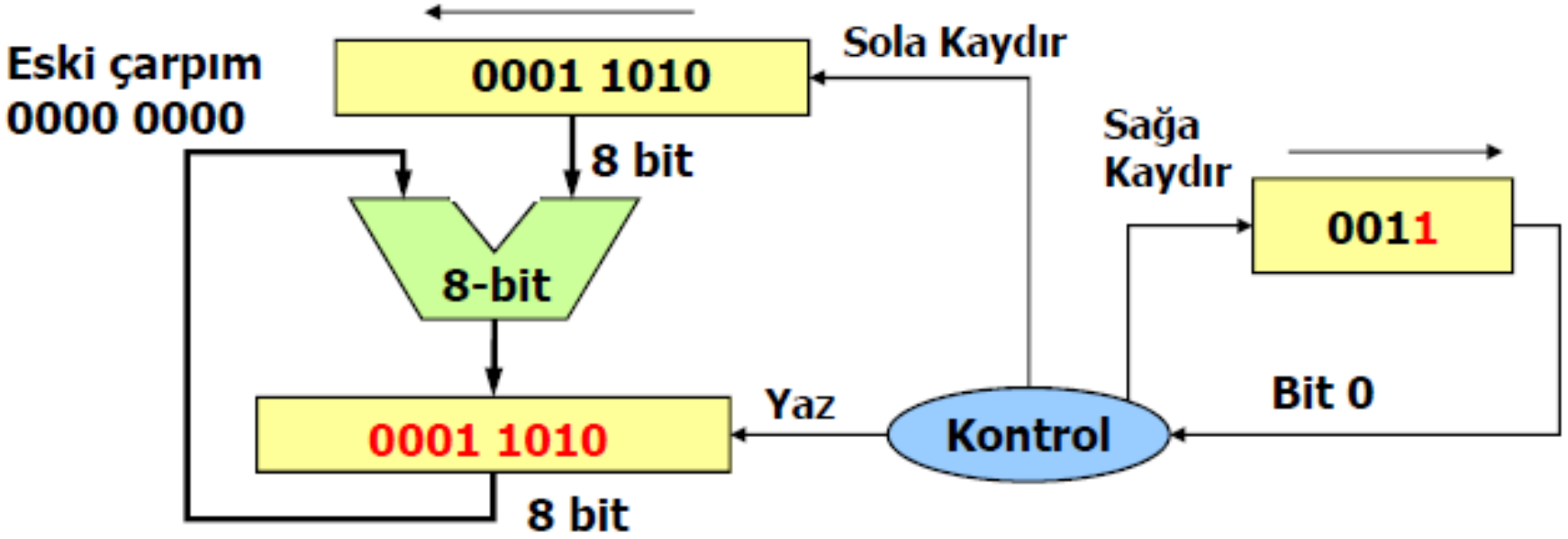



# 4-Bitlik Çarpma Örneği (1.Yöntem) 2a






- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.**
- Çarpılanı bir bit sola kaydır
- Çarpanı bir bit sağa kaydır.

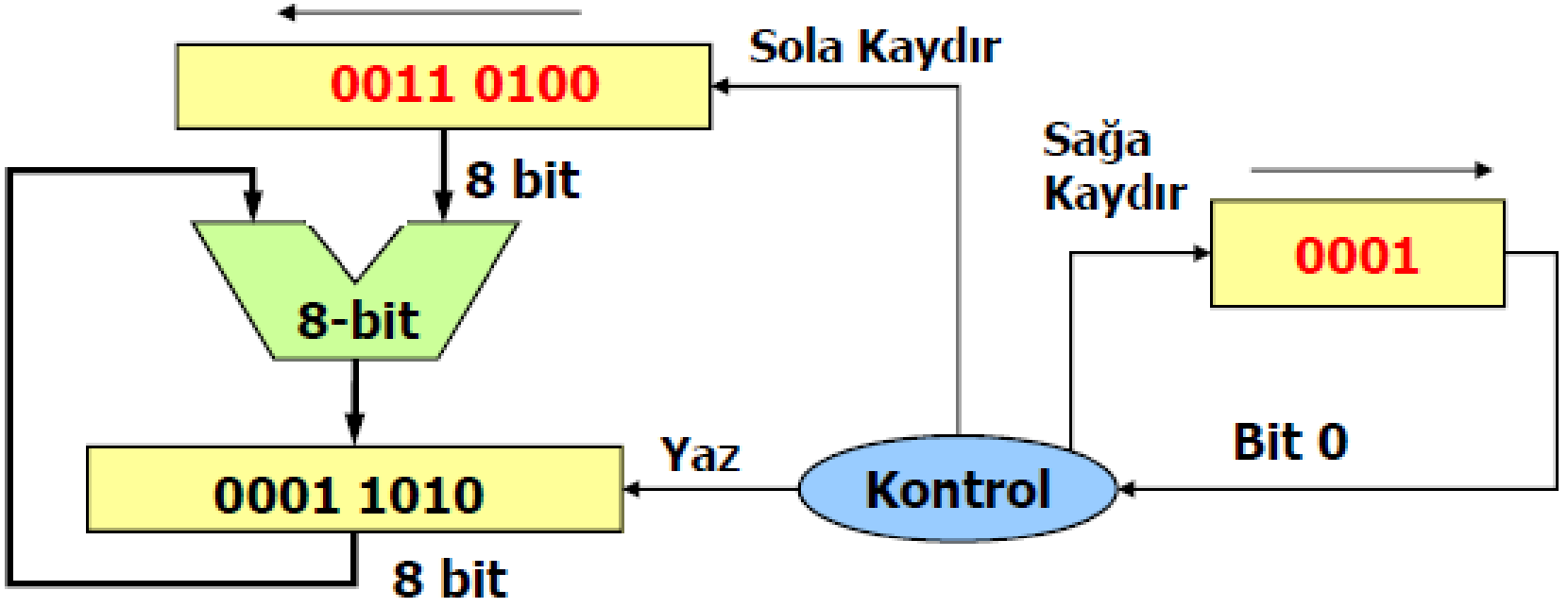



# 4-Bitlik Çarpma Örneği (1.Yöntem) 2b



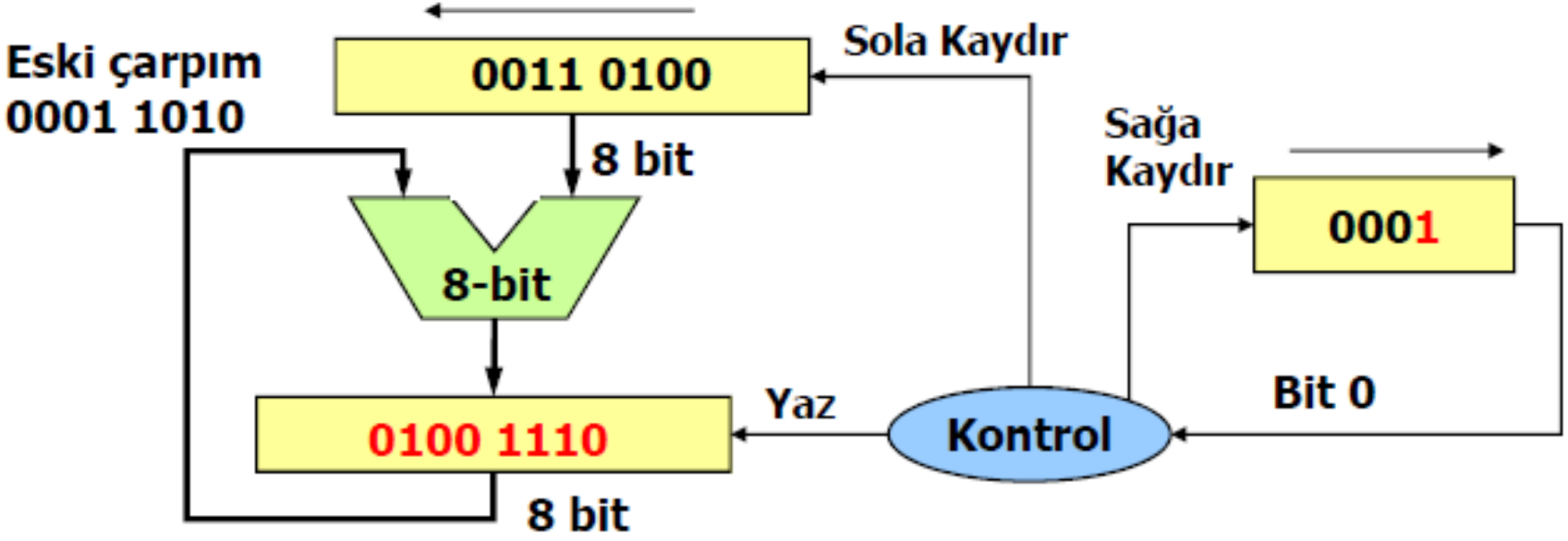



- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.
- **Çarpılanı bir bit sola kaydır**
- **Çarpanı bir bit sağa kaydır.**




# 4-Bitlik Çarpma Örneği (1.Yöntem) 3a

- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.**
- Çarpılanı bir bit sola kaydır
- Çarpanı bir bit sağa kaydır.

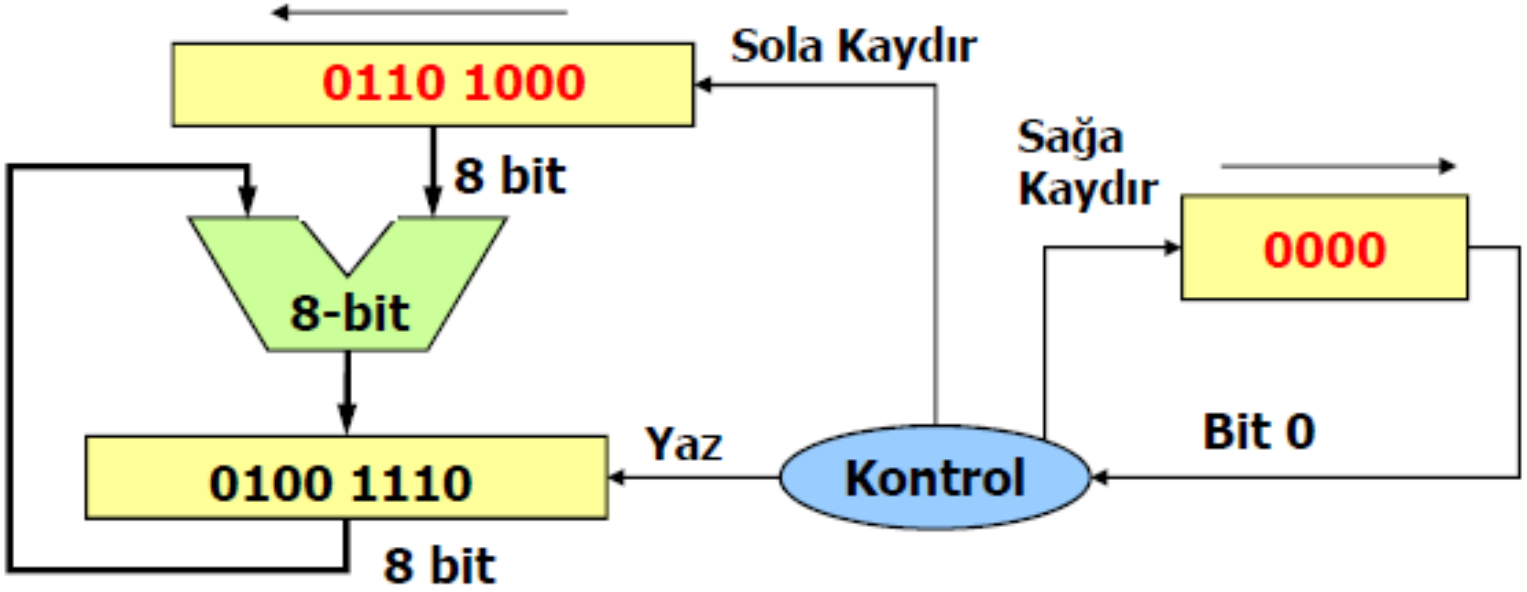



# 4-Bitlik Çarpma Örneği (1.Yöntem) 3b






- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.
- **Çarpılanı bir bit sola kaydır**
- **Çarpanı bir bit sağa kaydır.**

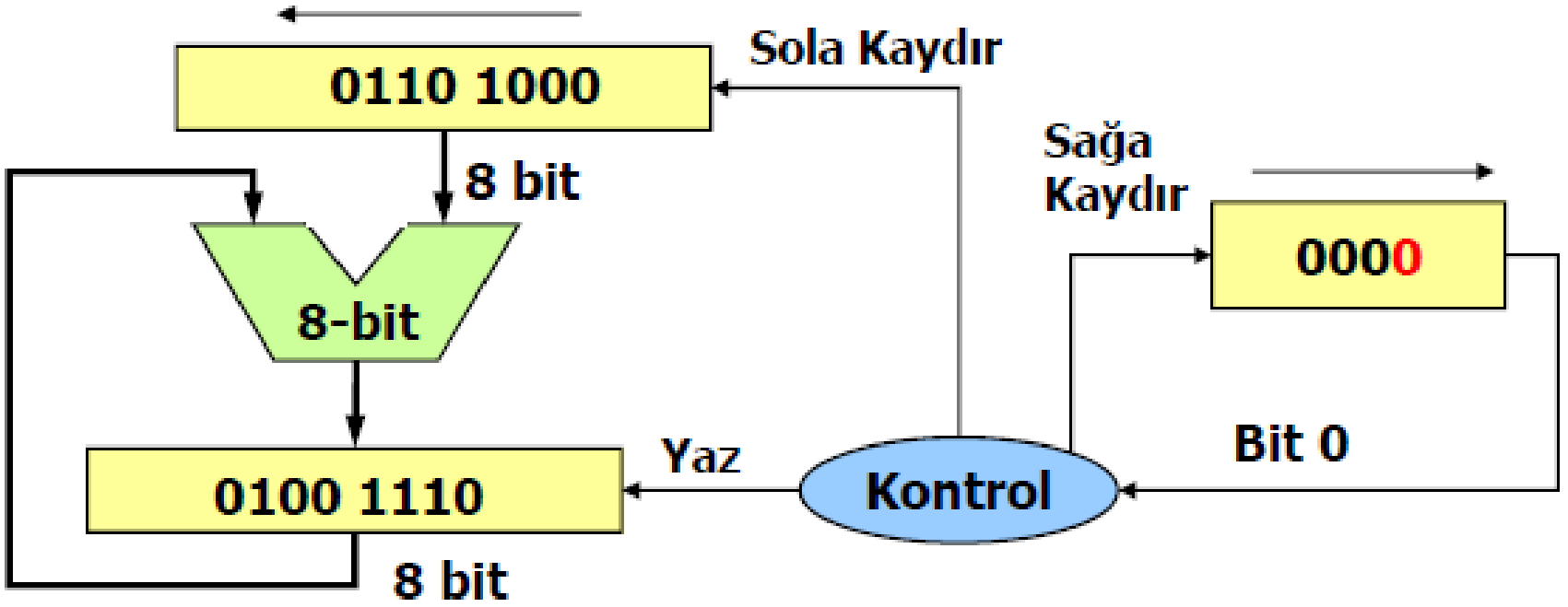



# 4-Bitlik Çarpma Örneği (1.Yöntem) 4a






- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.**
- Çarpılanı bir bit sola kaydır
- Çarpanı bir bit sağa kaydır.

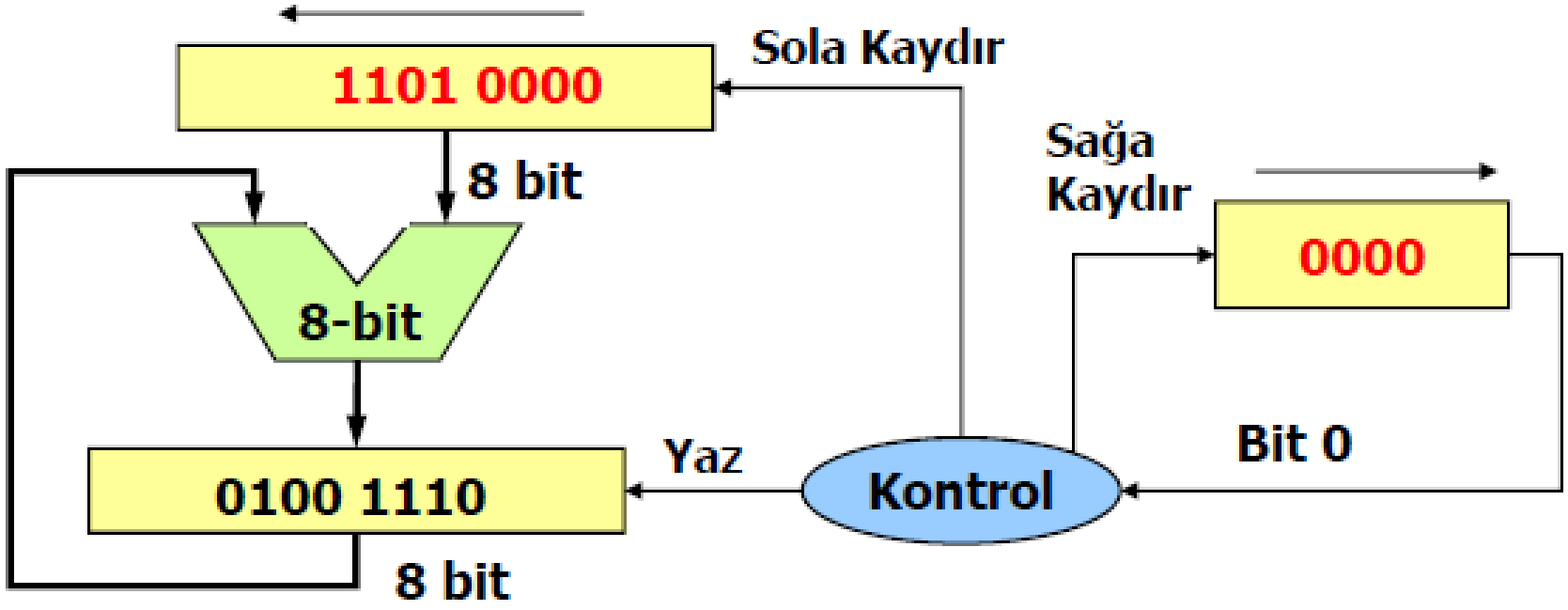



# 4-Bitlik Çarpma Örneği (1.Yöntem) 4b






- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değerin çarpıma eklenmesi için Toplayıcıyı yetkilendir.
- **Çarpılanı bir bit sola kaydır**
- **Çarpanı bir bit sağa kaydır.**

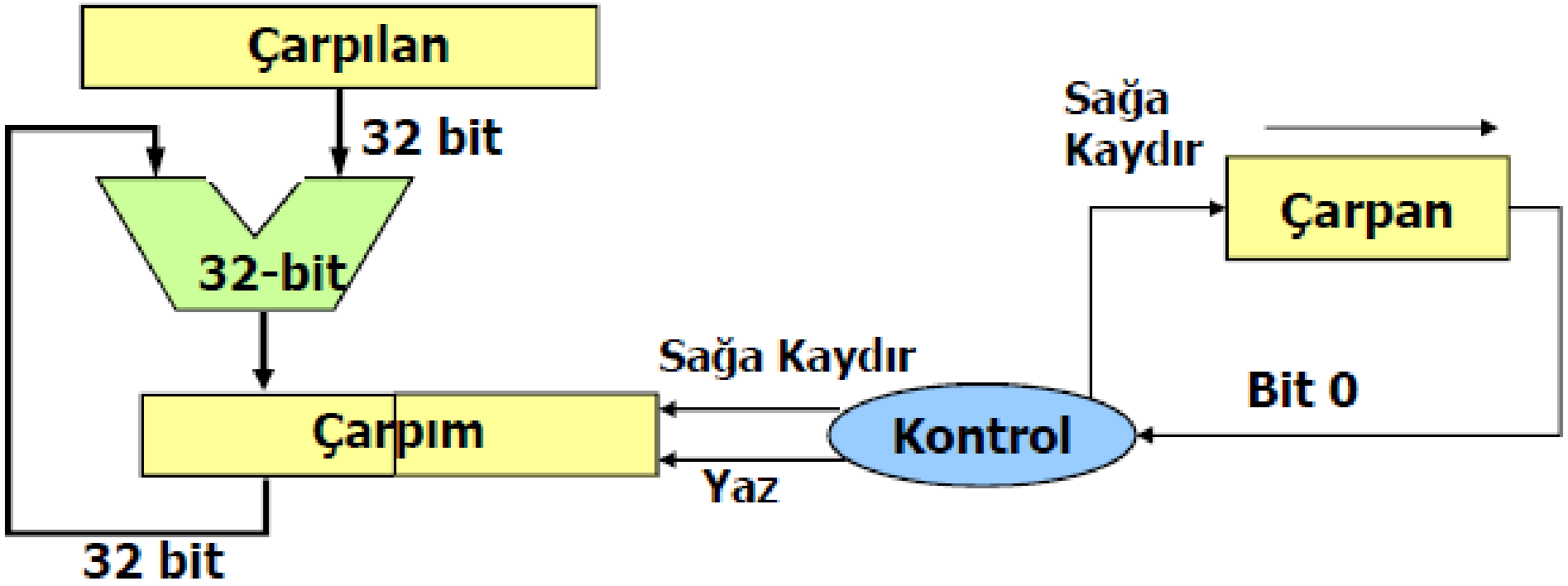



# Çarpma İşlemi (2.Yöntem)






- Çarpılanı sola kaydırmak yerine çarpım sağa kaydırılabilir.
- Bu yaklaşım ile çarpılan 64 yerine 32-bitlik kaydedicide saklanabilir.
- Ayrıca 64-bitlik toplayıcı yerine 32-bitlik toplayıcı kullanılabilir.

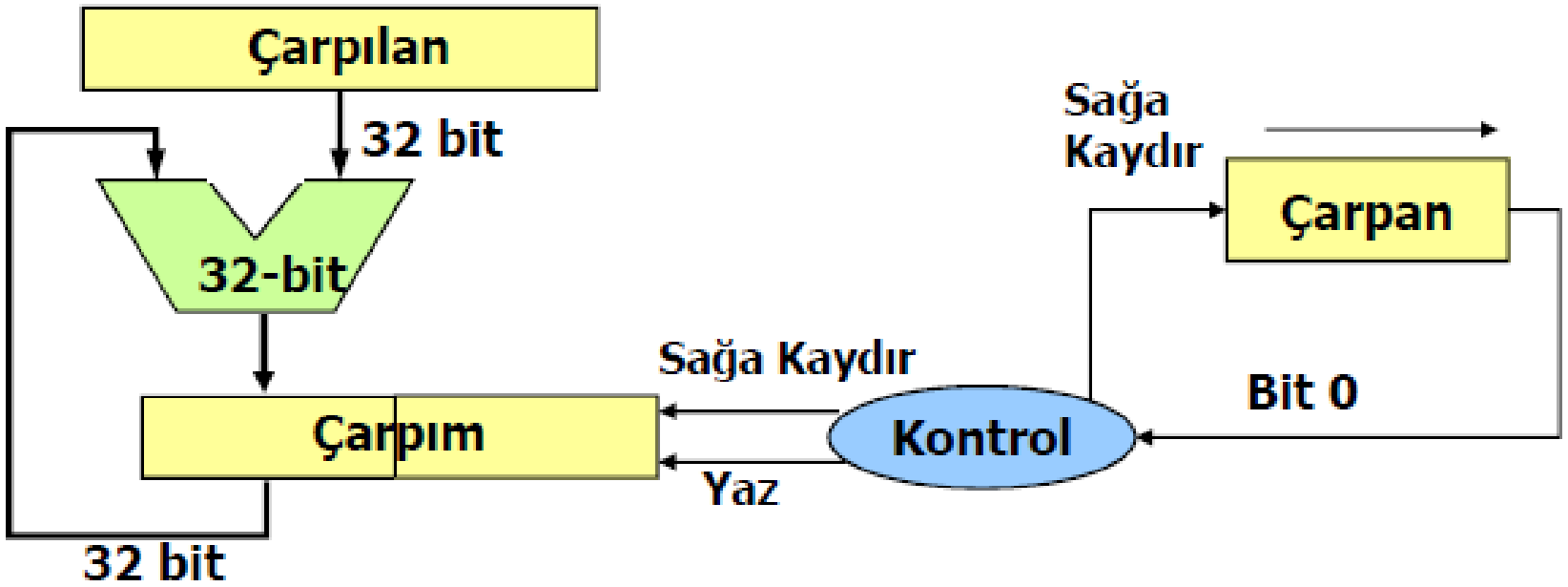



# Çarpma İşlemi (2.Yöntem)



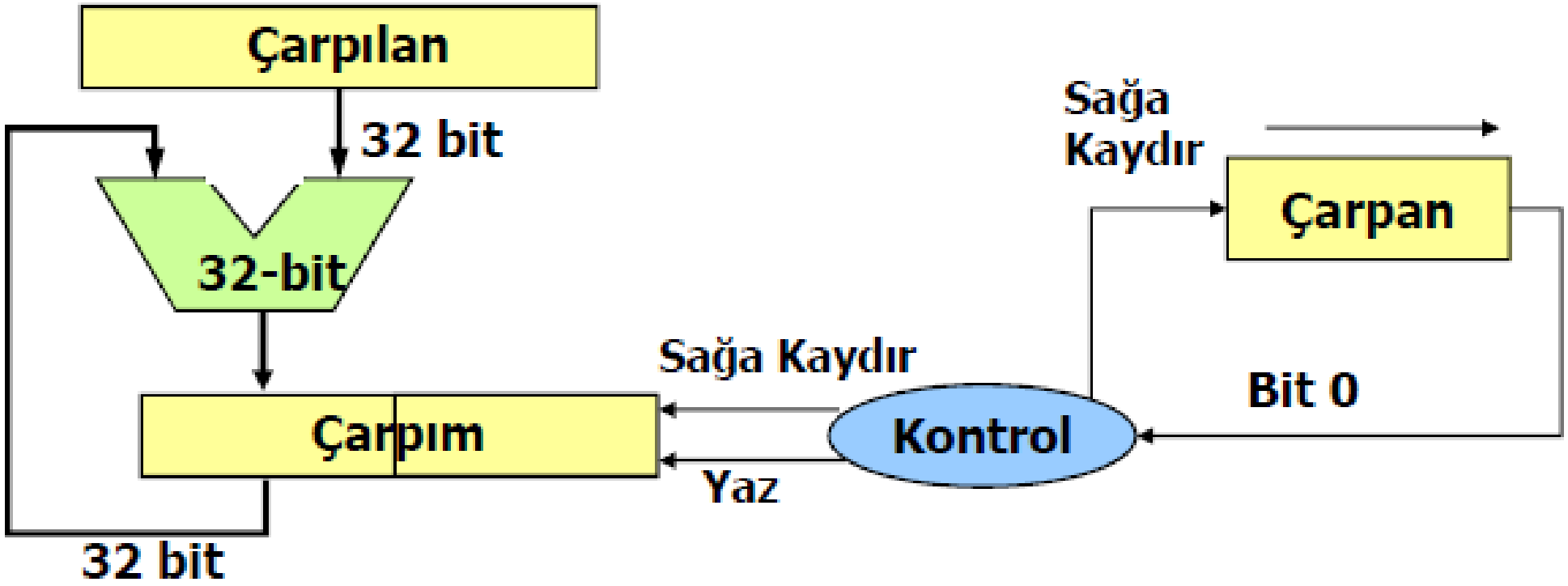



- 32 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımın sol yarısına eklemek için Toplayıcıyı yetkilendir
- Çarpımı bir bit sağa kaydır
- Çarpanı bir bit sağa kaydır.




# 4-Bitlik Çarpma Örneği (2.Yöntem)

- 1101 değeri 0110 değeri ile 2. algoritmaya göre çarpılmak istenmektedir.



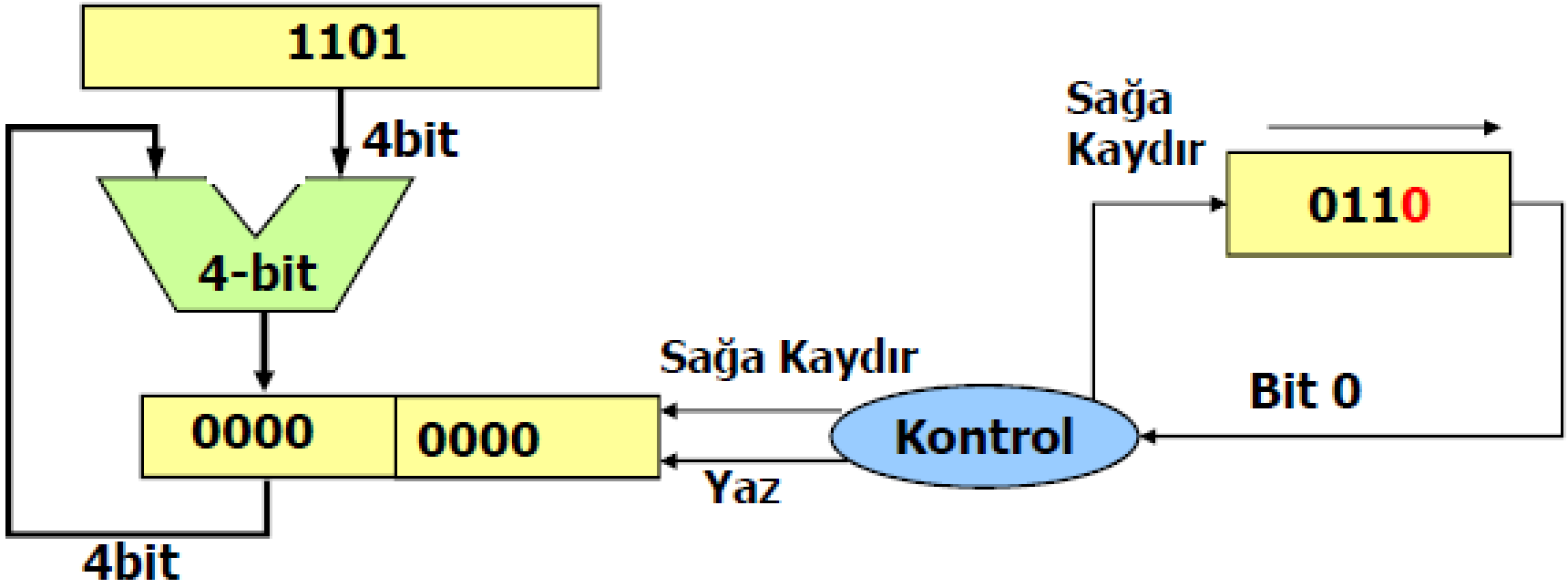



# 4-Bitlik Çarpma Örneği (2.Yöntem) 1a






- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.**
- Çarpımı bir bit sağa kaydır
- Çarpanı bir bit sağa kaydır.

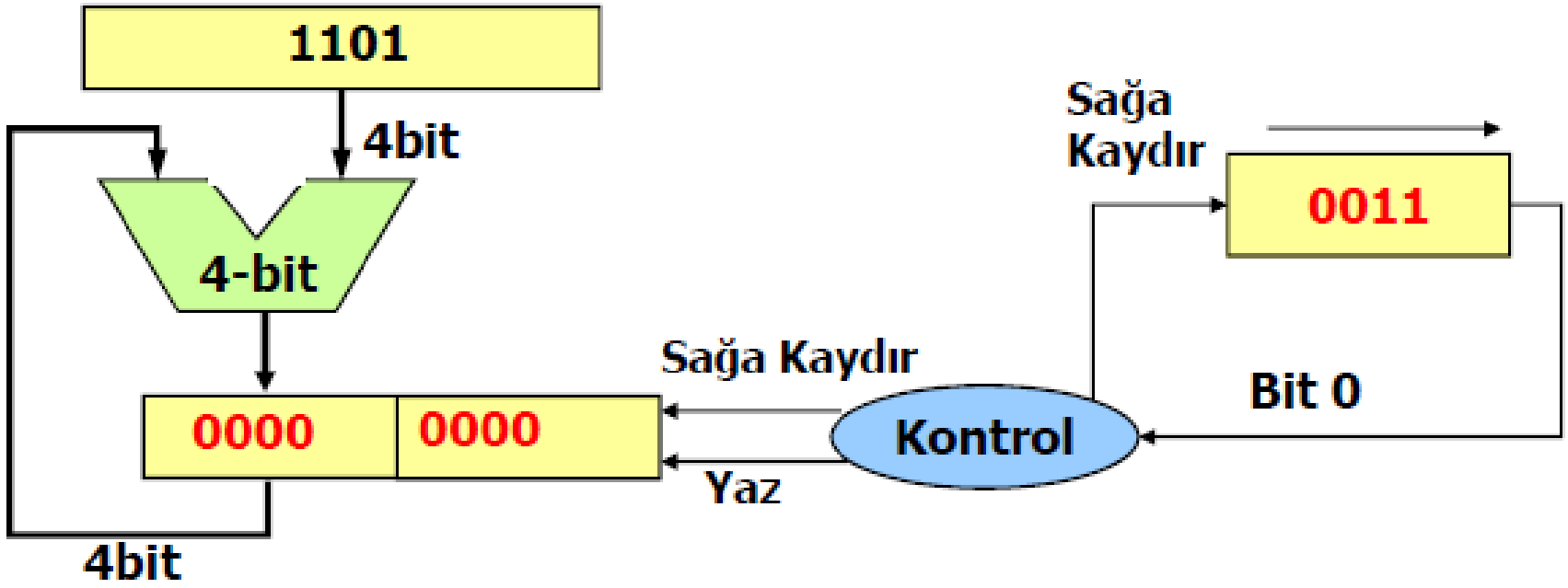



# 4-Bitlik Çarpma Örneği (2.Yöntem) 1b






- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.
- **Çarpımı bir bit sağa kaydır**
- **Çarpanı bir bit sağa kaydır.**



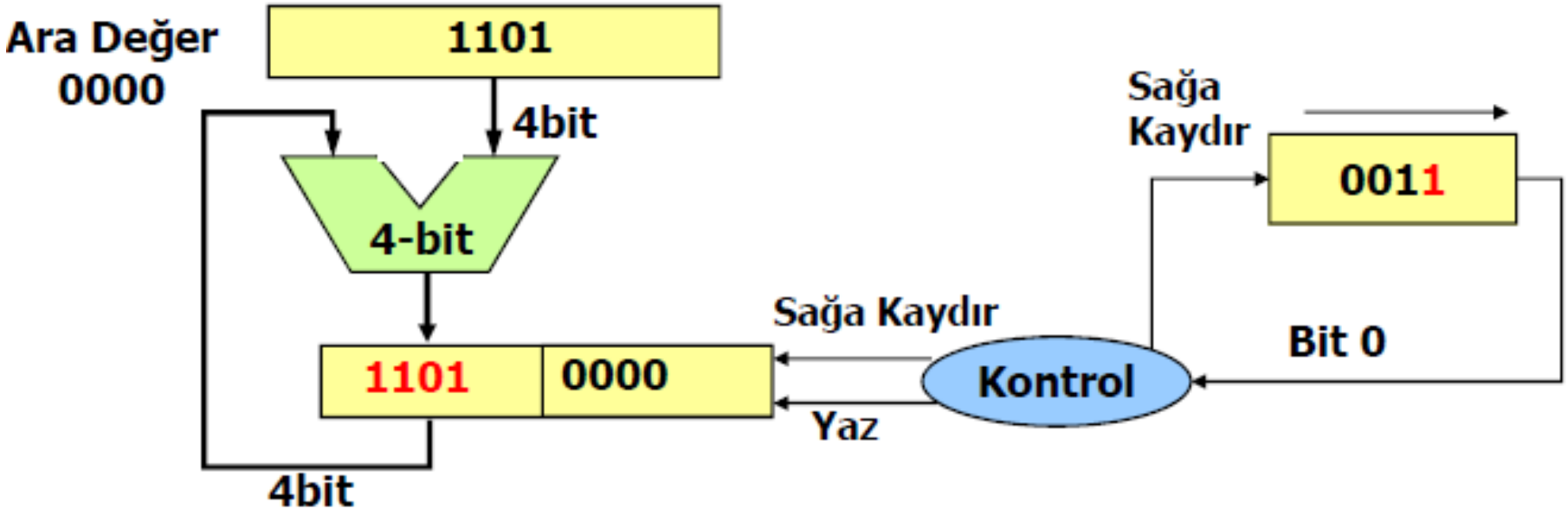

# 4-Bitlik Çarpma Örneği (2.Yöntem) 2a



MK



- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.**
- Çarpımı bir bit sağa kaydır
- Çarpanı bir bit sağa kaydır.

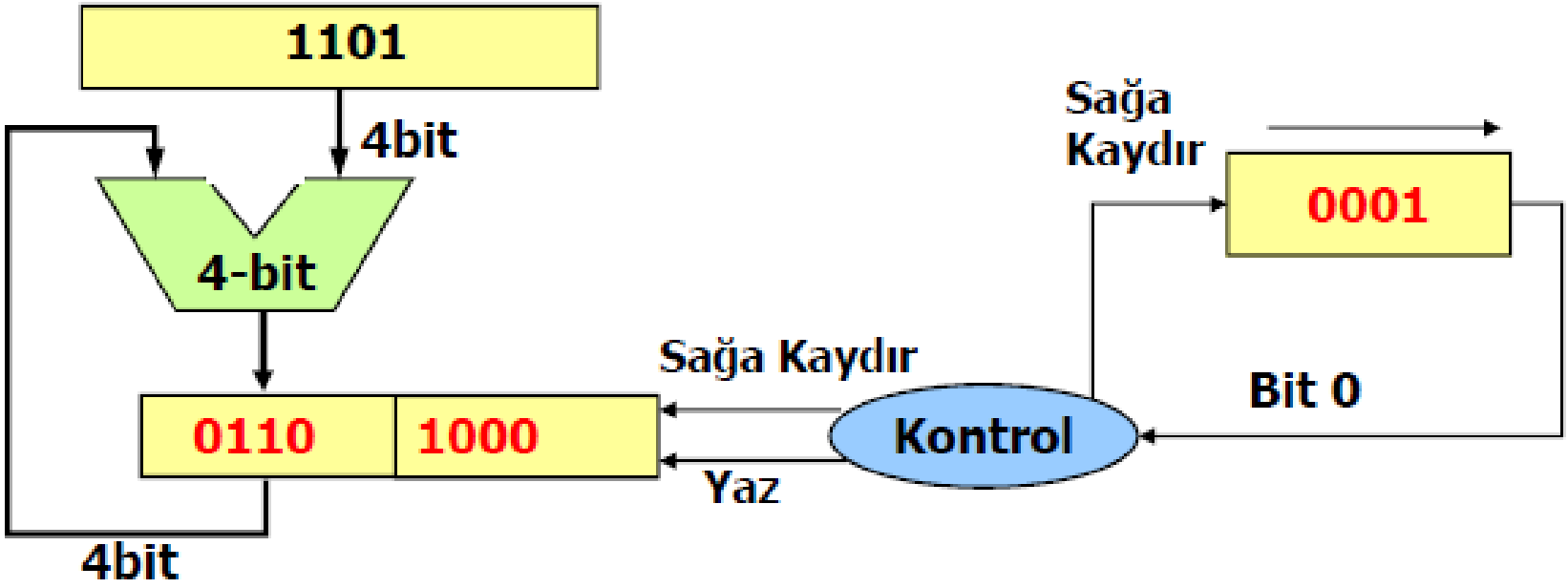



# 4-Bitlik Çarpma Örneği (2.Yöntem) 2b






- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.
- **Çarpımı bir bit sağa kaydır**
- **Çarpanı bir bit sağa kaydır.**

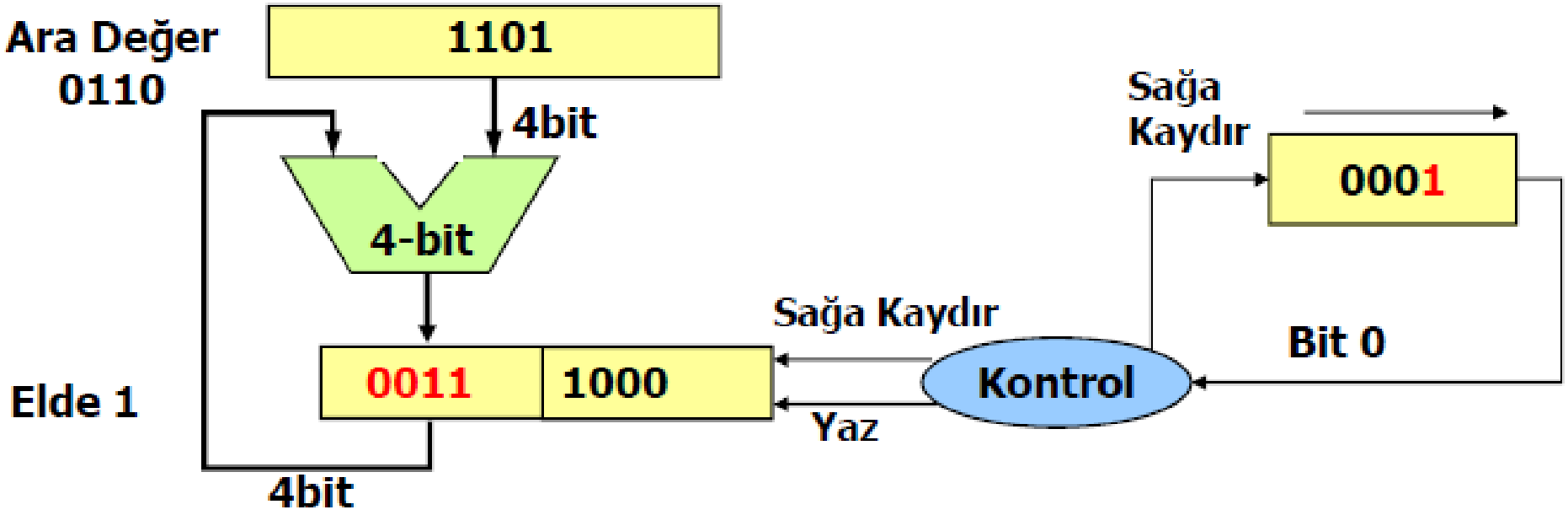



# 4-Bitlik Çarpma Örneği (2.Yöntem) 3a






- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.**
- Çarpımı bir bit sağa kaydır
- Çarpanı bir bit sağa kaydır.

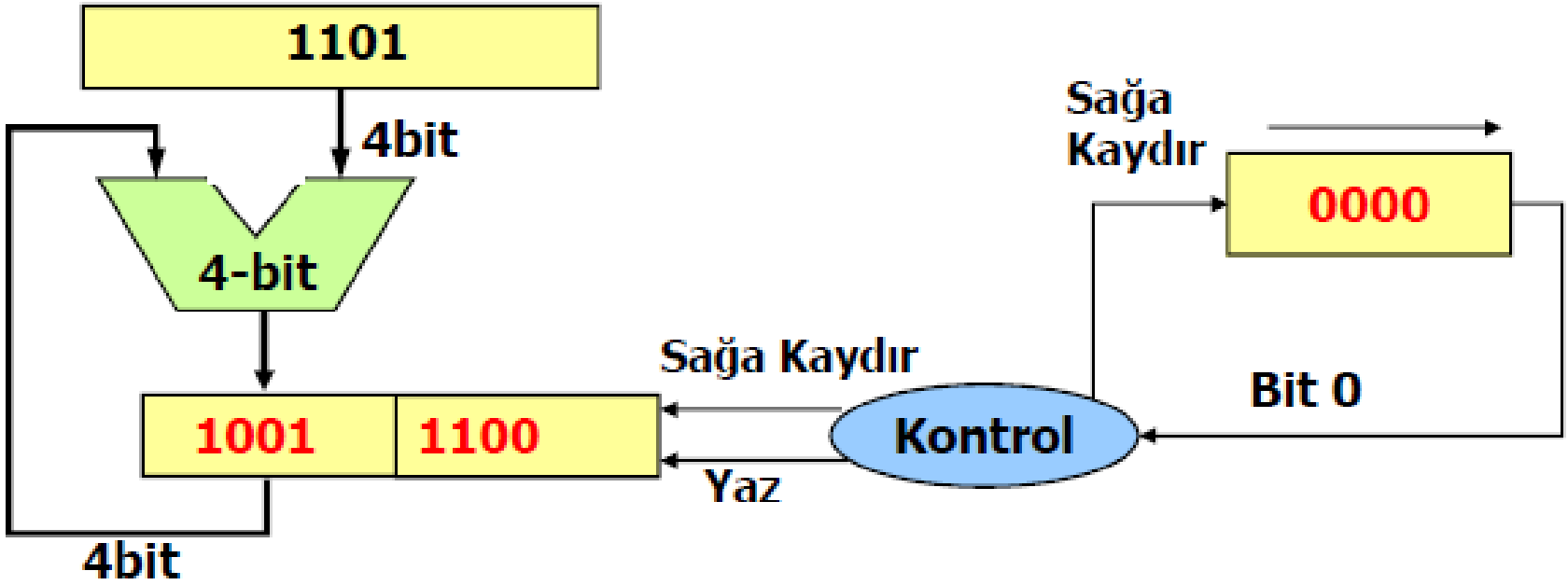



# 4-Bitlik Çarpma Örneği (2.Yöntem) 3b






- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.
- **Çarpımı bir bit sağa kaydır**
- **Çarpanı bir bit sağa kaydır.**

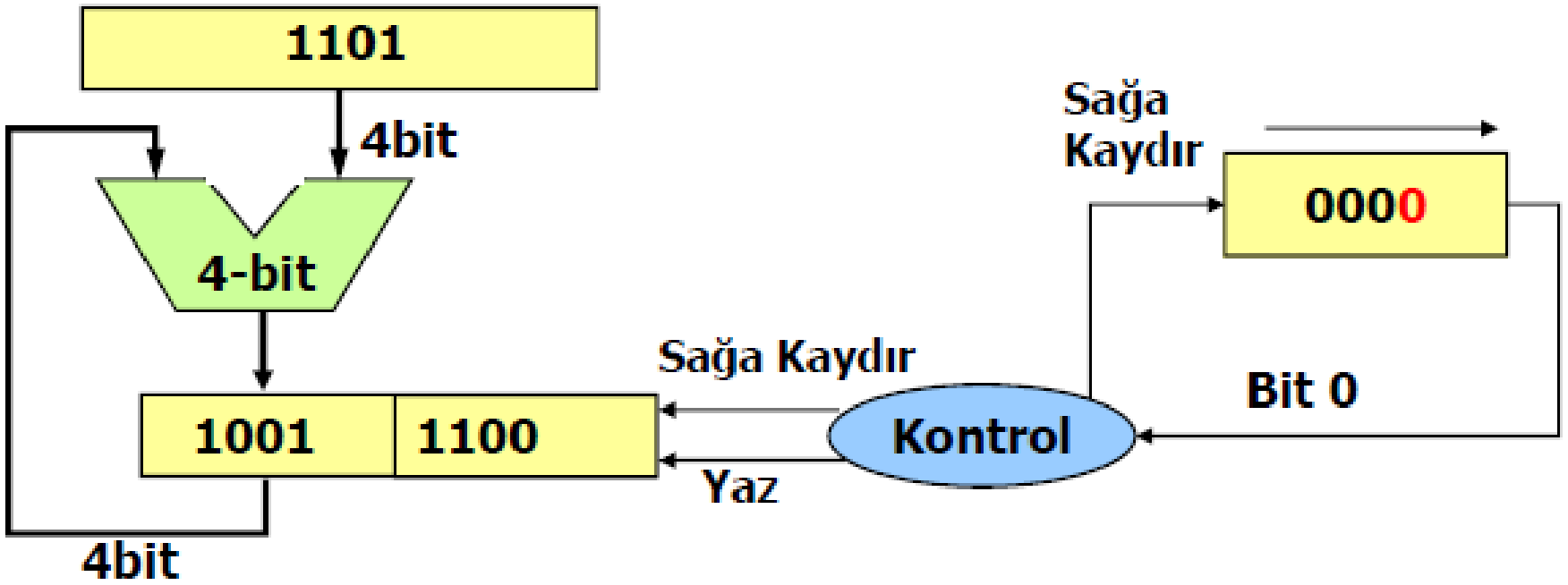



# 4-Bitlik Çarpma Örneği (2.Yöntem) 4a






- 4 kez tekrarla
- **if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.**
- Çarpımı bir bit sağa kaydır
- Çarpanı bir bit sağa kaydır.

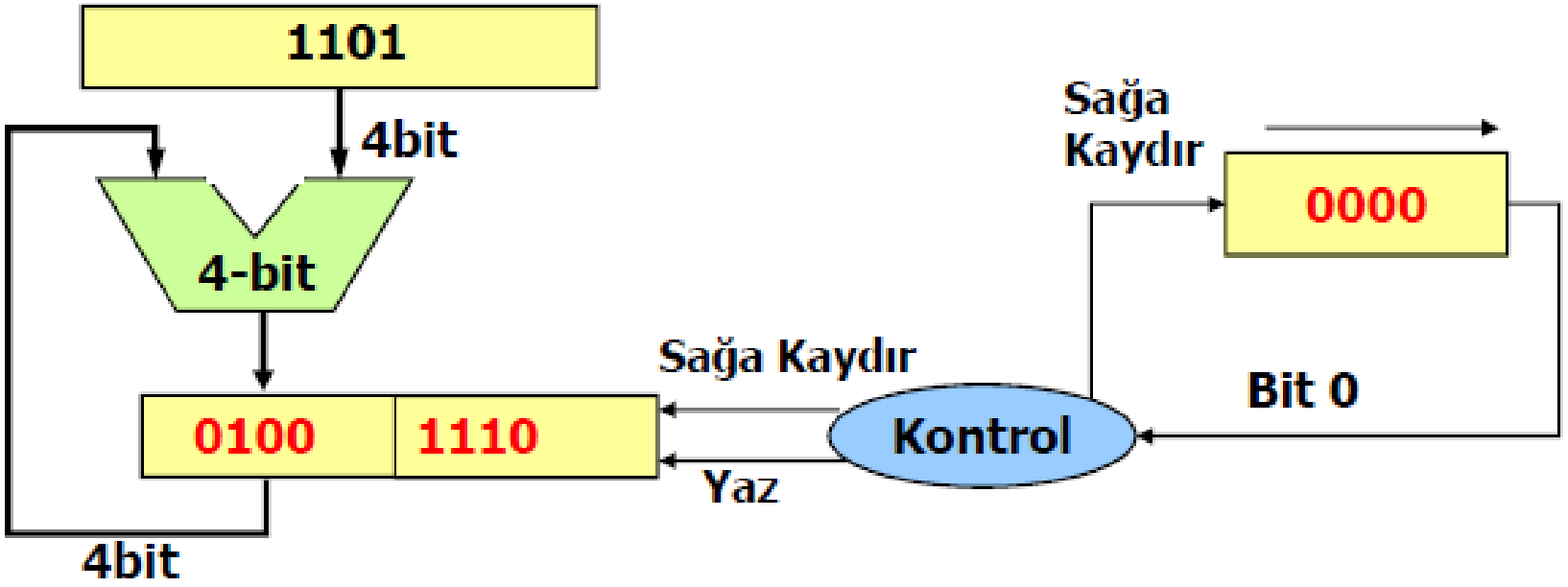



# 4-Bitlik Çarpma Örneği (2.Yöntem) 4b






- 4 kez tekrarla
- if Çarpan'nın 0. bit değeri 1'e eşitse ara değeri çarpımının sol yarısına eklemek için Toplayıcıyı yetkilendir.
- **Çarpımı bir bit sağa kaydır**
- **Çarpanı bir bit sağa kaydır.**

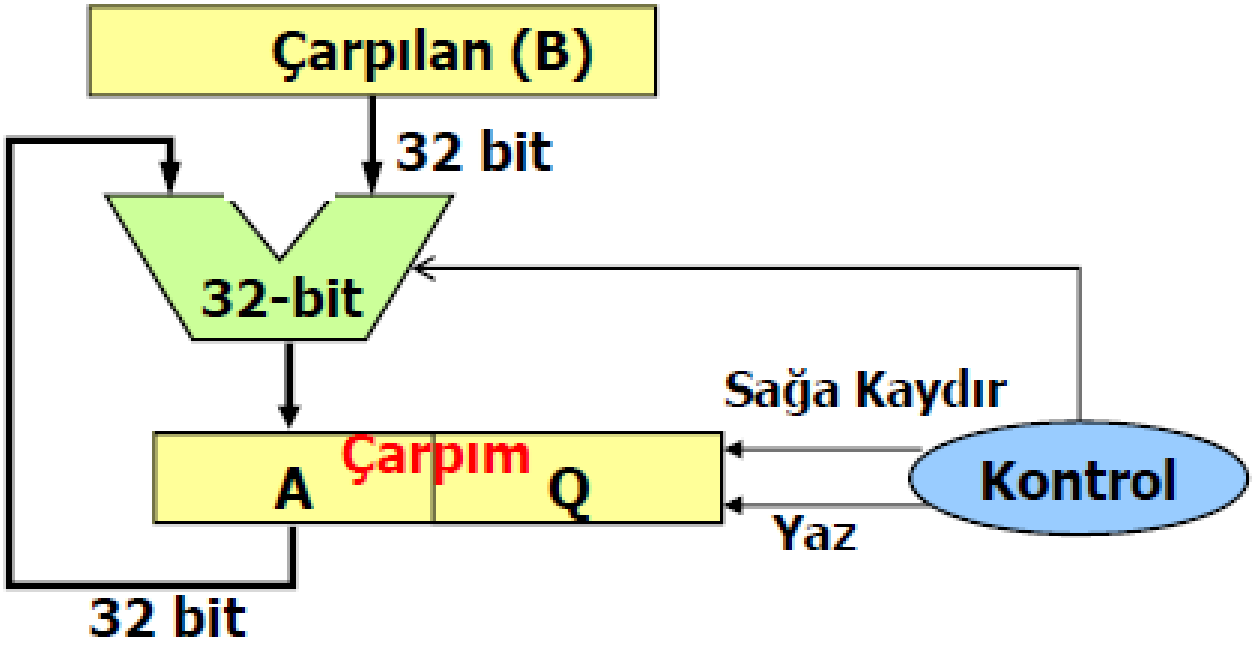



# Çarpma İşlemi (3.Yöntem)



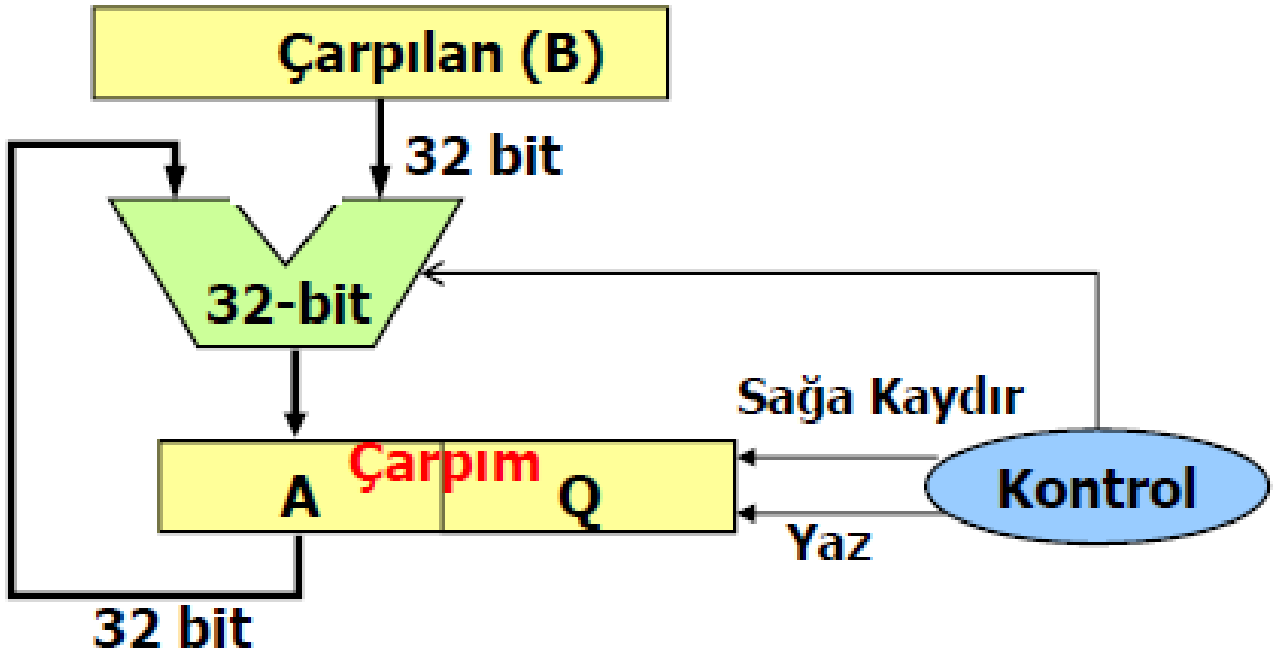
- Bu yöntemde çarpan ayrı bir kaydedicide saklanmak yerine çarpım'ın düşük değerlikli (sağ yarısı) 32 bitinde saklanmaktadır.

- B ⟶ Çarpılan
- Q ⟶ Çarpan
- Çarpım = AQ




# Çarpma İşlemi (3.Yöntem)

- Bu yöntemde çarpan ayrı bir kaydedicide saklanmak yerine çarpım'ın düşük değerlikli (sağ yarısı) 32 bitinde saklanmaktadır.
- B ⟶ Çarpılan
- Q ⟶ Çarpan
- Çarpım = AQ



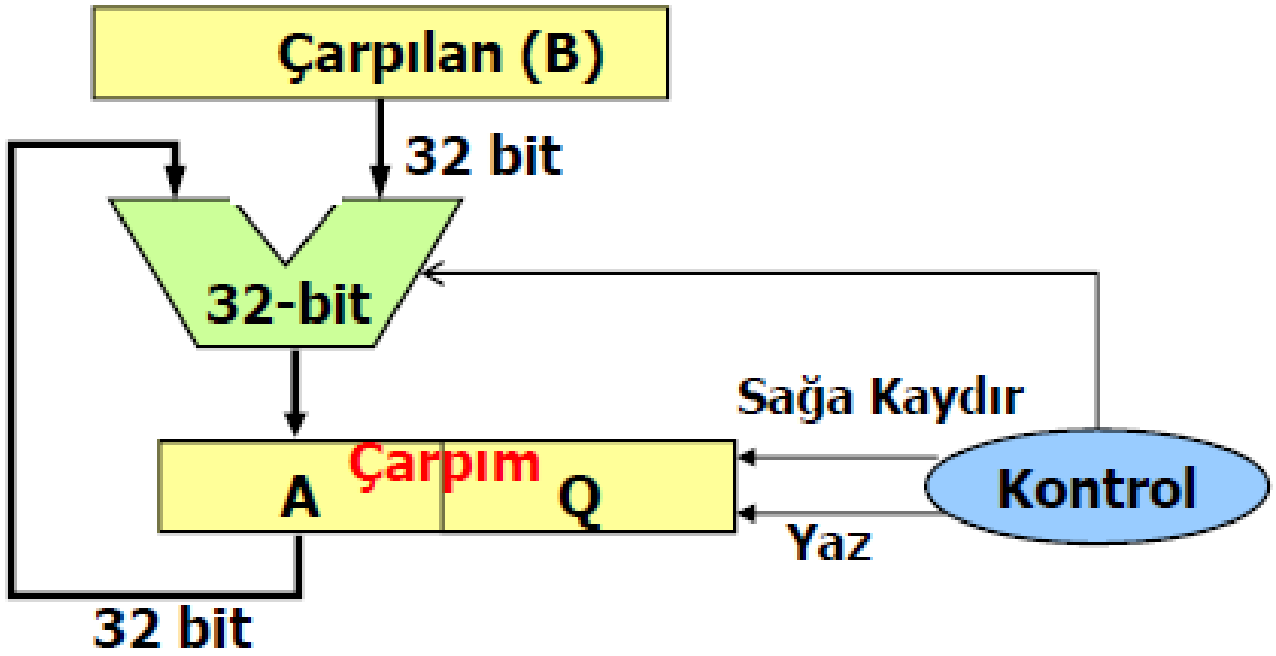



# Çarpma İşlemi (3.Yöntem)






- 32 kez tekrarla.
- if Çarpımın 0. bit değeri ($Q_0$) 1'e eşitse Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- Çarpımı bir bit sağa kaydır.



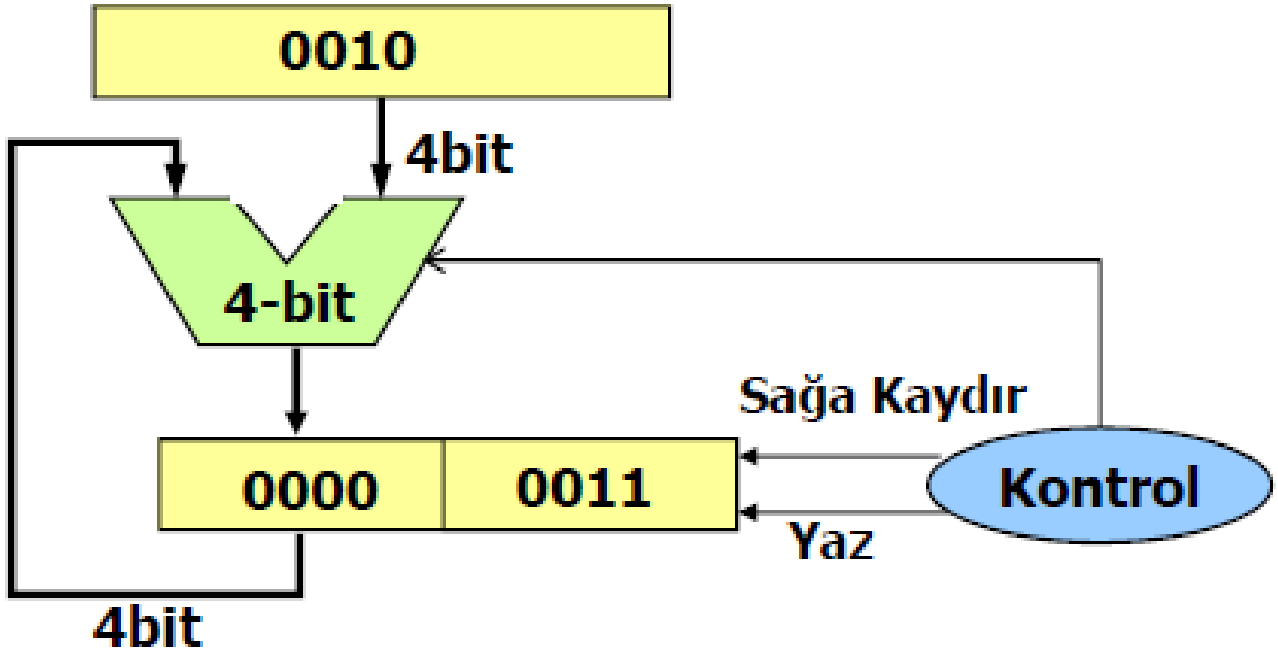

# 4-Bitlik Çarpma Örneği (3.Yöntem)








- 0010 değeri 0011 değeri ile çarpılmak istenmektedir.

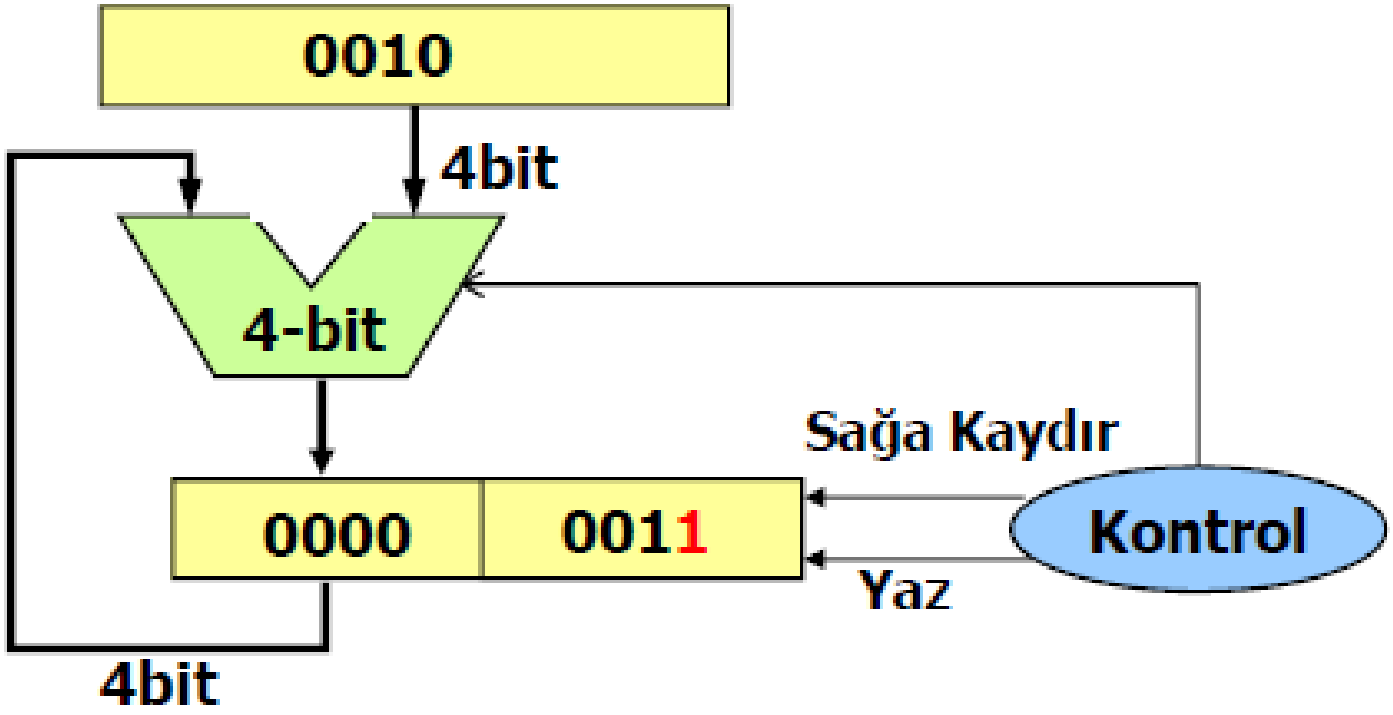



# Çarpma İşlemi (3.Yöntem) 1a






- 4 kez tekrarla.
- **if Çarpımın 0. bit değeri ($Q_0$) 1'e eşitse**
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- Çarpımı bir bit sağa kaydır.



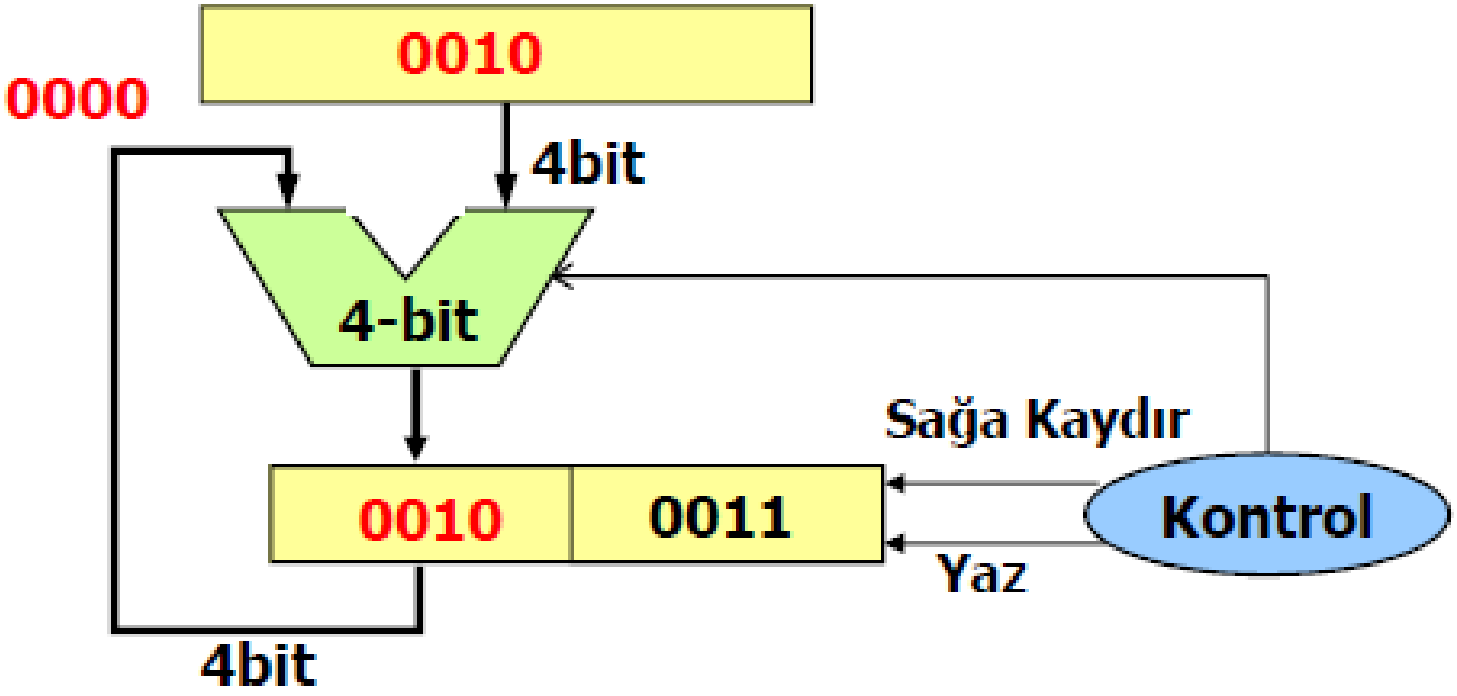

# Çarpma İşlemi (3.Yöntem) 1b







- 4 kez tekrarla.
- if Çarpımın 0. bit değeri ($Q_0$) 1'e eşitse
- **Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.**
- Çarpımı bir bit sağa kaydır.



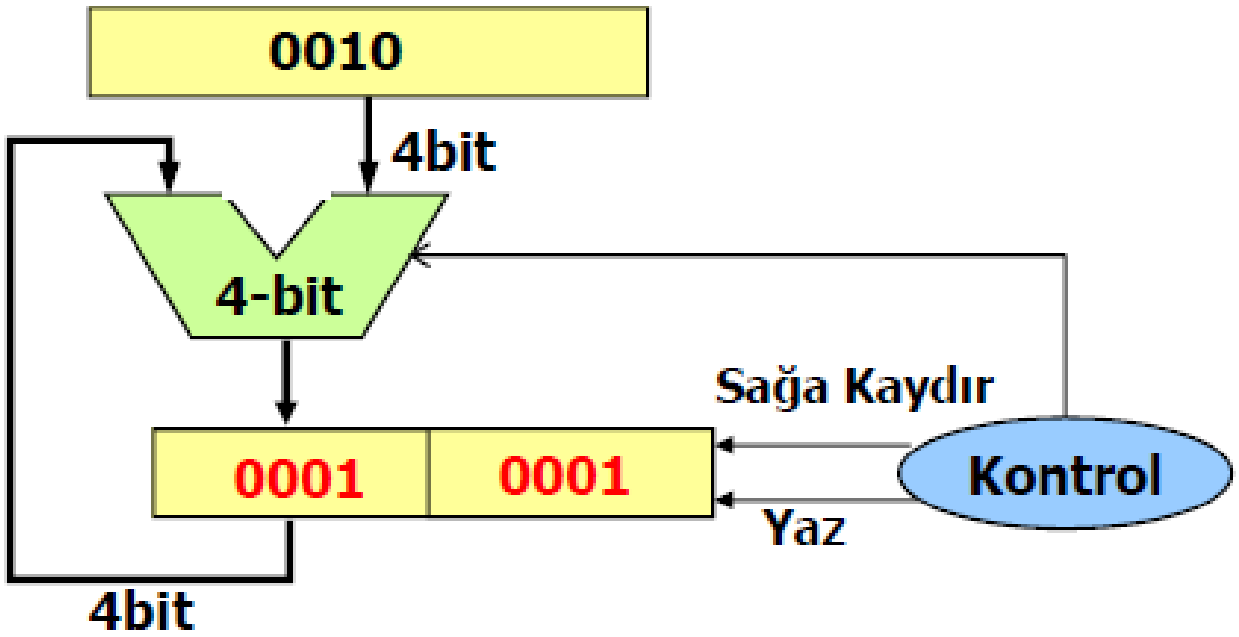

# Çarpma İşlemi (3.Yöntem) 1c







- 4 kez tekrarla.
- if Çarpımın 0. bit değeri (Q0) 1'e eşitse
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- **Çarpımı bir bit sağa kaydır.**

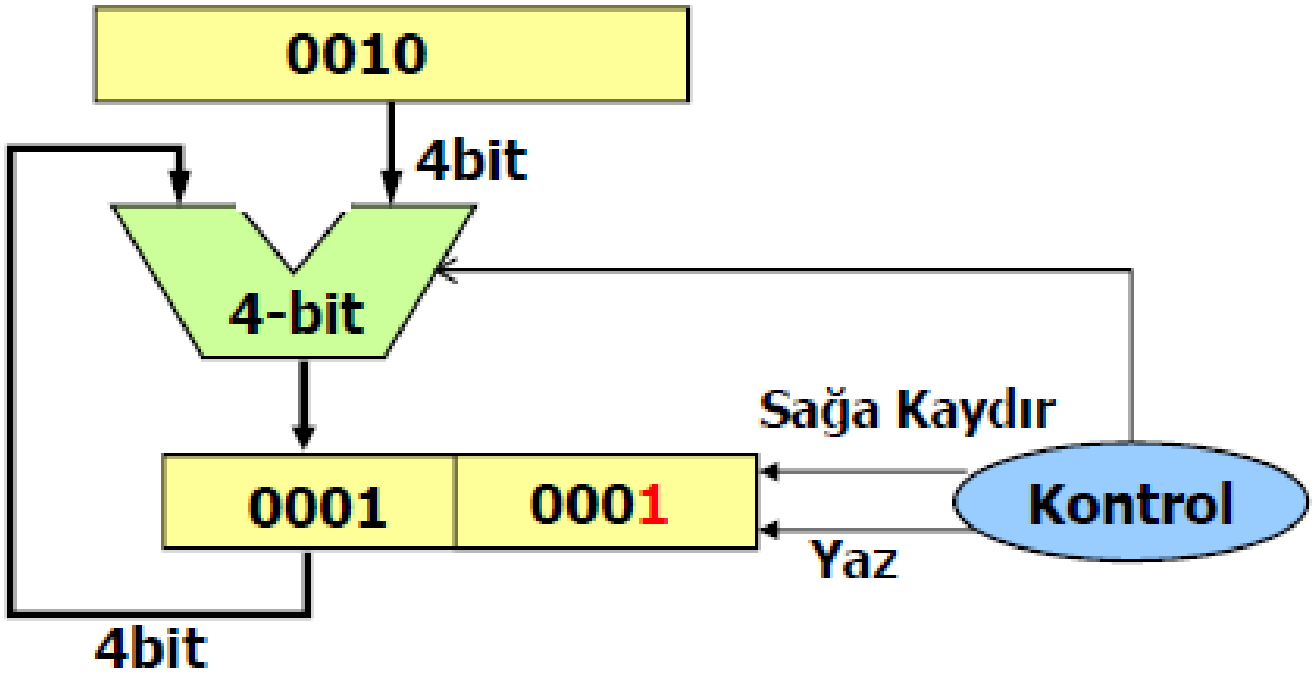



# Çarpma İşlemi (3.Yöntem) 2a






- 4 kez tekrarla.
- **if Çarpımın 0. bit değeri ($Q_0$) 1'e eşitse**
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- Çarpımı bir bit sağa kaydır.

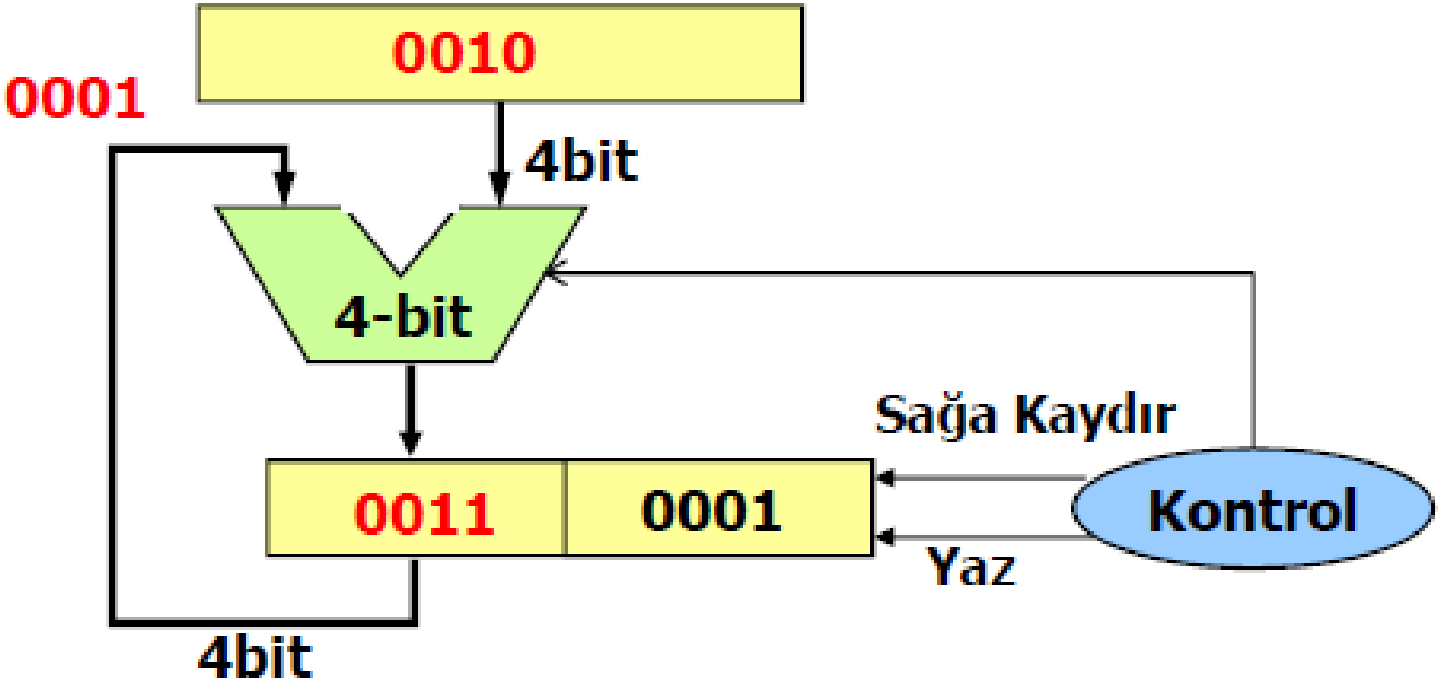



# Çarpma İşlemi (3.Yöntem) 2b






- 4 kez tekrarla.
- if Çarpımın 0. bit değeri ($Q_0$) 1'e eşitse
- **Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.**
- Çarpımı bir bit sağa kaydır.

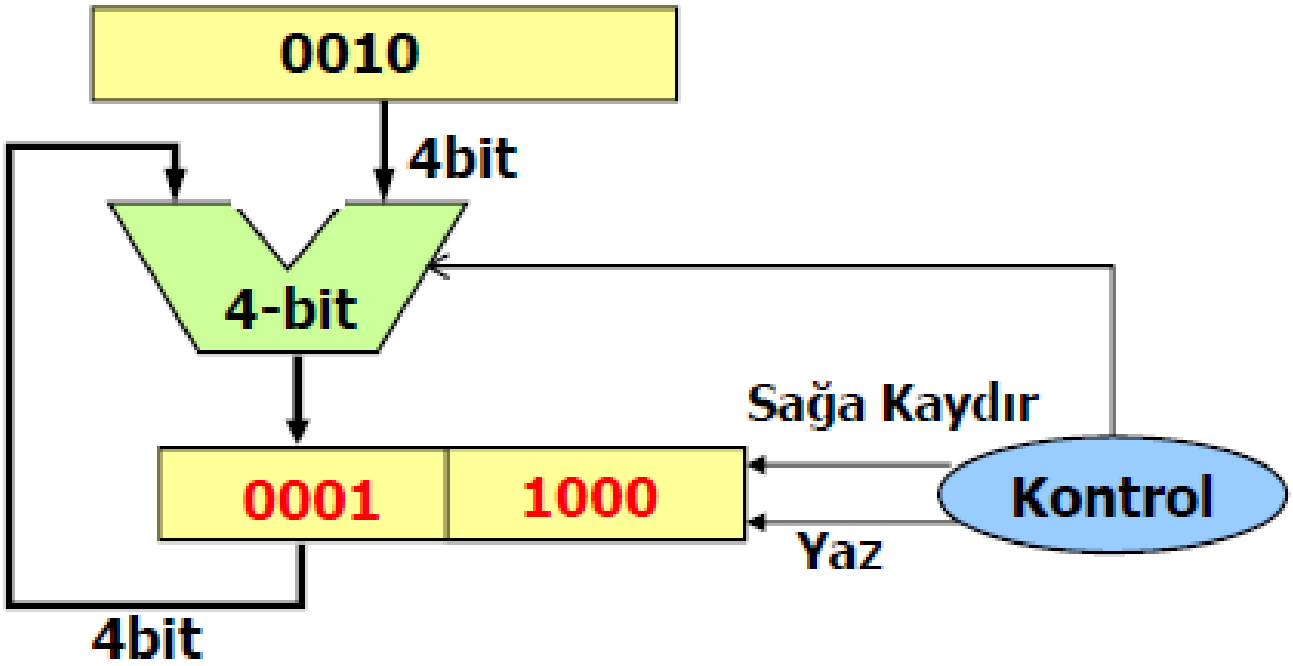



# Çarpma İşlemi (3.Yöntem) 2c





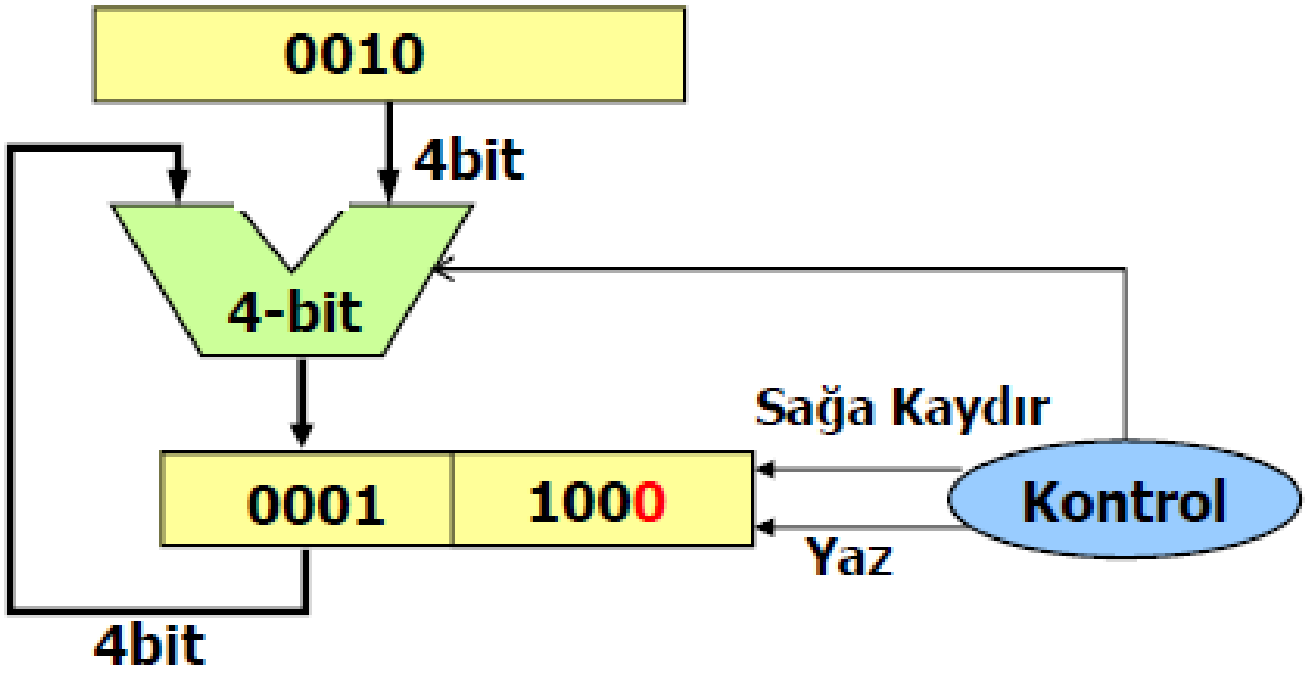

- 4 kez tekrarla.
- if Çarpımın 0. bit değeri (Q0) 1'e eşitse
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- **Çarpımı bir bit sağa kaydır.**




# Çarpma İşlemi (3.Yöntem) 3a



- 4 kez tekrarla.
- **if Çarpımın 0. bit değeri ($Q_0$) 1'e eşitse**
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- Çarpımı bir bit sağa kaydır.

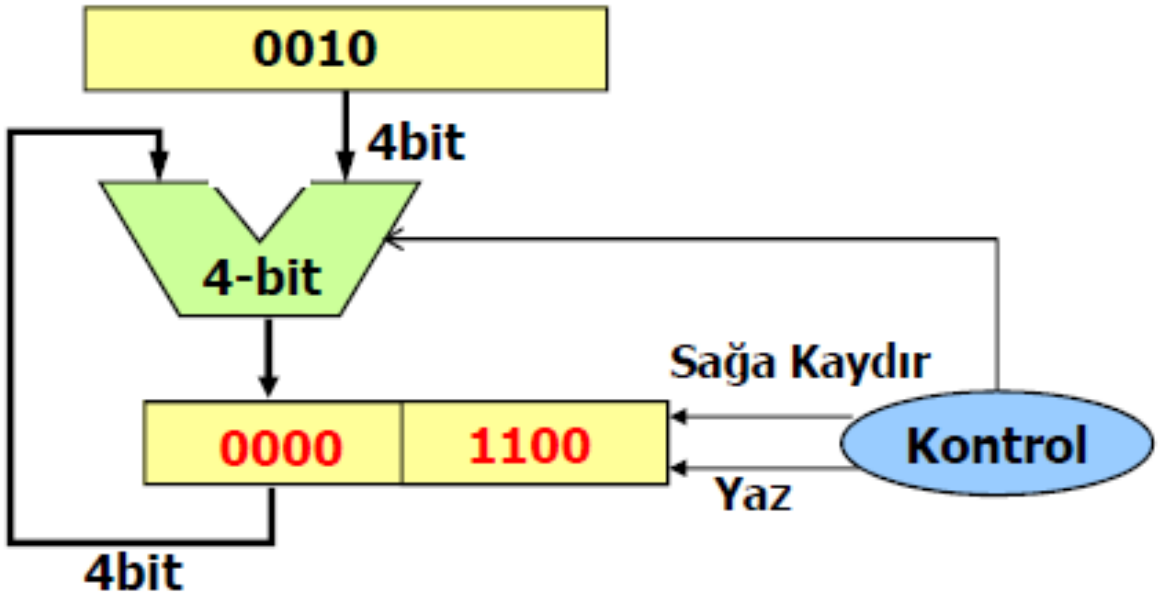



# Çarpma İşlemi (3.Yöntem) 3b






- 4 kez tekrarla.
- if Çarpımın 0. bit değeri (Q0) 1'e eşitse
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- **Çarpımı bir bit sağa kaydır.**



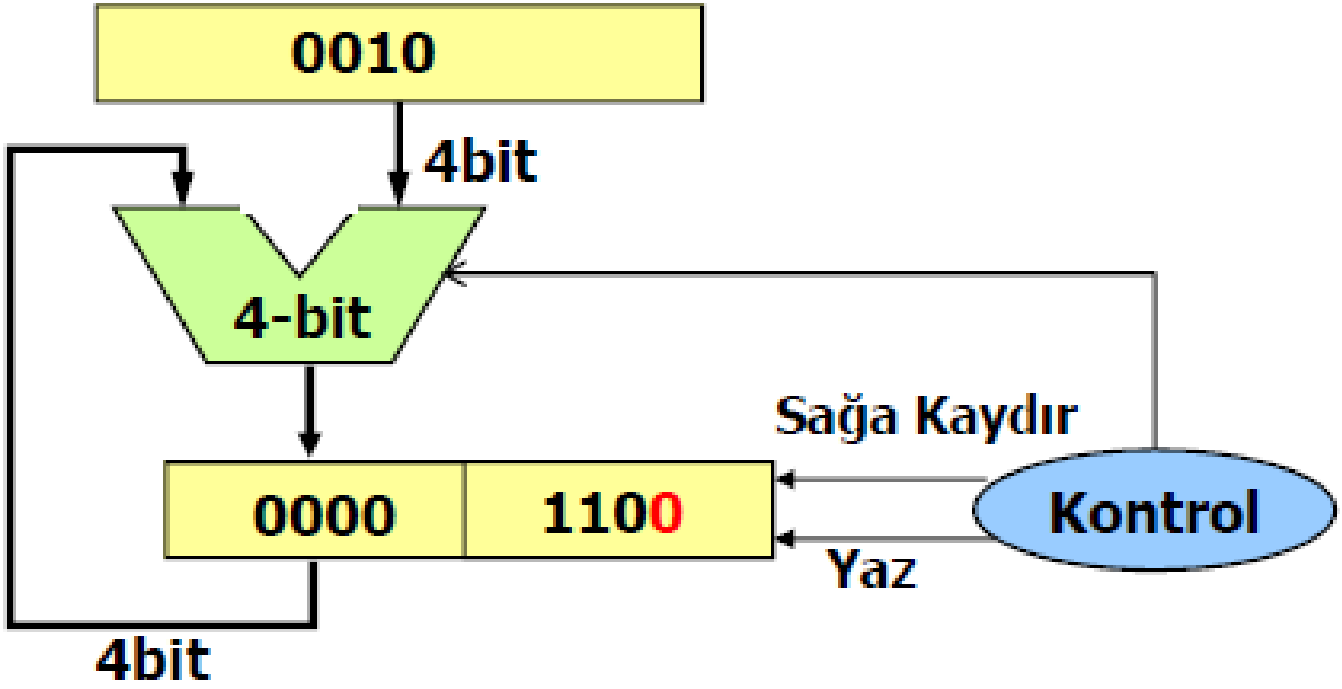

# Çarpma İşlemi (3.Yöntem) 4a







- 4 kez tekrarla.
- **if Çarpımın 0. bit değeri ($Q_0$) 1'e eşitse**
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- Çarpımı bir bit sağa kaydır.

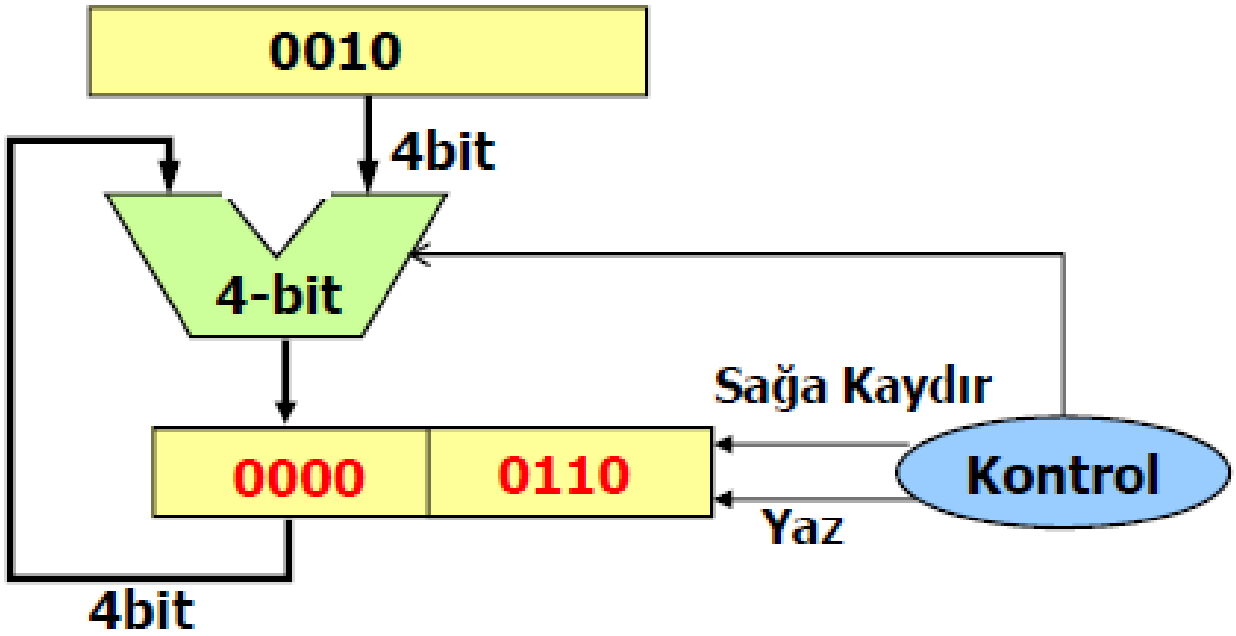



# Çarpma İşlemi (3.Yöntem) 4b






- 4 kez tekrarla.
- if Çarpımın 0. bit değeri (Q0) 1'e eşitse
- Çarpılan ile çarpımın sol yarısını topla ve çarpımın sol yarısına yaz.
- **Çarpımı bir bit sağa kaydır.**




# İşaretli Sayıların Çarpımı

- Şimdiye kadar anlatılan yöntemler işaretli sayılar için doğrudan kullanılamamaktadır.
  - 1101 × 0110 (–3 × 6)
  - Sonuç= 1110 1110 (–18) olmalı, 0100 1110 (+78) olmamalı
- Bu yöntemler ile iki işaretli sayı çarpılırken çarpma işlemi normal bir şekilde yapılır ve sonucun işaretine göre tümleyen aritmetiği kullanılabilir.
- Ancak işaretlerin tespit edilmesi ve sonucun duruma göre terslenmesi ekstra işlem yükü ve devre gerektirmektedir.
- Booth's algoritması işaretli sayıların çarpılabilmesi için geliştirilmiş verimli bir yöntemdir.
  - İşaretli ikiye tümleyen sayıları üzerinde çalışır.
  - Çarpma işleminde gereken toplama sayısını düşürmektedir.



# Booth Algoritmasının Genelleştirilmesi



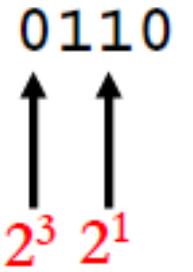
MK



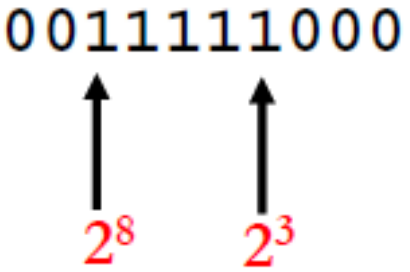
- Yöntem çarpandaki 1'lerin sırasına göre genelleştirilebilir.
  - $(2^i–2^j)x= 2^ix–2^jx, [ i > j ]$
- 00101×01110 (5×14) işlemi düşünüldüğünde normalde 3 toplama gerektirmektedir.
  - İfade (00101 ×10000) – (00101 ×00010) şeklinde yeniden yazılabilir.
  - On'lu olarak, 5×14 =(5×16) – (5×2).
  - Bir adet toplama ve bir adet çıkarma işlemi
- Çarpandaki art arda gelen 1'lerin sayısı arttıkça daha fazla toplama işlemi elimine edilebilir.

# Booth Algoritmasının Örnekleri

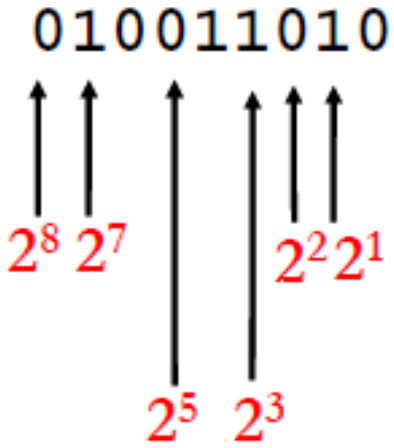





$2^3-2^1 \longrightarrow (8-2=6)$




$2^8-2^3 \longrightarrow (256-8 = 248)$




$2^8-2^7 + 2^5-2^3 + 2^2-2^1$

$256-128 + 32-8 + 4-2$



# Booth Algoritmasında 1'lerin Sırası






- Booth algoritması çarpandaki 1'lerin sırasına bakmaktadır. Bu yöntemde çarpanda sağdan sola art arta gelen iki bit aranmaktadır.
- Toplam 4 durum vardır.

| Bit i | Bit i-1 | Anlamı | Örnek |
|---|---|---|---|
| 1 | 0 | 1'lerin başlangıcı | 0000111**10**00 |
| 1 | 1 | 1'lerin ortası | 0000**11**1000 |
| 0 | 1 | 1'lerin sonu | 0000**1**111000 |
| 0 | 0 | 0'ların ortası | 0000**0**1111000 |

- 1'lerin başlangıcında çıkarma yapılır ($-2^j x$).
- 1'lerin sonunda toplama yapılır ($2^i x$).
- Algoritmaya başlayabilmek için orijinal çarpanın en sağdaki bitine bit-1 olarak adlandırılan ekstra bir bit eklenir.

# Booth Algoritması






- Booth algoritması 1'lerin ve 0'ların ortasında iken herhangi bir toplama ya da çıkarma işlemine gerek duymadığı için hızlıdır.
- İşaretli sayıların çarpımında Çarpımın işareti, sağdaki kaymalı kaydedicide tutulmalıdır.
- Bu işlem aritmetik kaydırma olarak da adlandırılır.

```
Başlangıç
  Çarpım=0, ve bit –1 = 0
 n kere tekrar et
      if Çarpanın bit 0 ve bit –1 'i 10'a eşitse
           Çarpılanı çarpımın sol yarısından çıkar
      else if  çarpanın bit 0 ve bit –1'i 01'a eşitse
           Çarpılanı Çarpımın sol yarısına ekle
      İşareti koruyarak Çarpımı bir bit sağa kaydır
      Bit -1 de dahil olmak üzere Çarpanı bir bit sağa kaydır
```

# Booth Algoritması için 4-Bitlik Örnek






- 1101×0110 (–3×6) ifadesi Booth algoritmasına göre çarpılmak isteniyor.
- Sonuç=1110 1110(–18) olmalı

```
Başlangıç
  Çarpım=0, ve bit –1 = 0
 n kere tekrar et
      if Çarpanın bit 0 ve bit –1 'i 10'a eşitse
           Çarpılanı çarpımın sol yarısından çıkar
      else if  çarpanın bit 0 ve bit –1'i 01'a eşitse
           Çarpılanı Çarpımın sol yarısına ekle
      İşareti koruyarak Çarpımı bir bit sağa kaydır
      Bit -1 de dahil olmak üzere Çarpanı bir bit sağa kaydır
```

| Çarpılan | Çarpım | Çarpan |
|---|---|---|
| 1101 | 0000 0000 | 0110 0 |