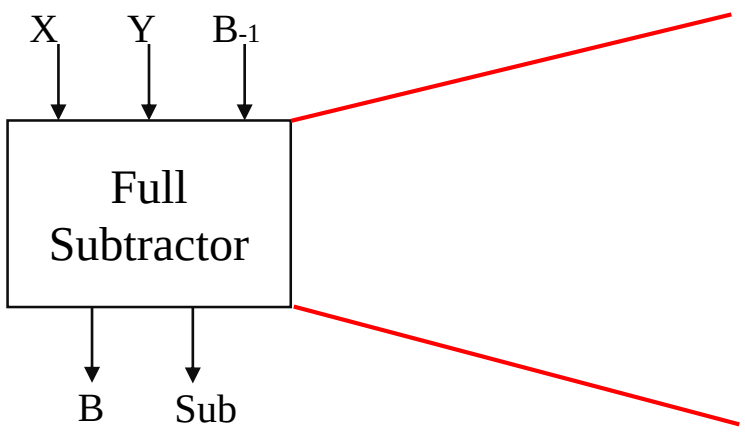
بنام خداوند جان و خرد

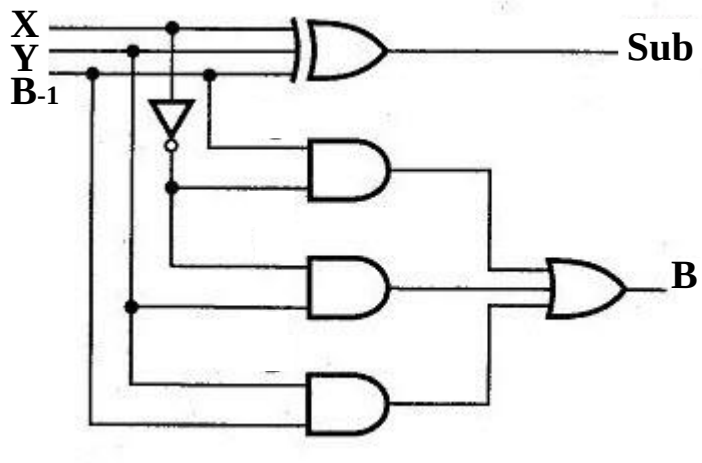
معماری کامپیوتر

جلسه چهاردهم: تفریق کننده-ضرب کننده

تفریق کننده (Subtractor)

- عملیات تفریق توسط جمع‌کننده انجام می‌شود
  - منفی شده عدد دوم (2’s complement) را با عدد اول جمع می‌کنیم
- عملیات تفریق توسط ماژول مجزا انجام شود (Full Subtractor)

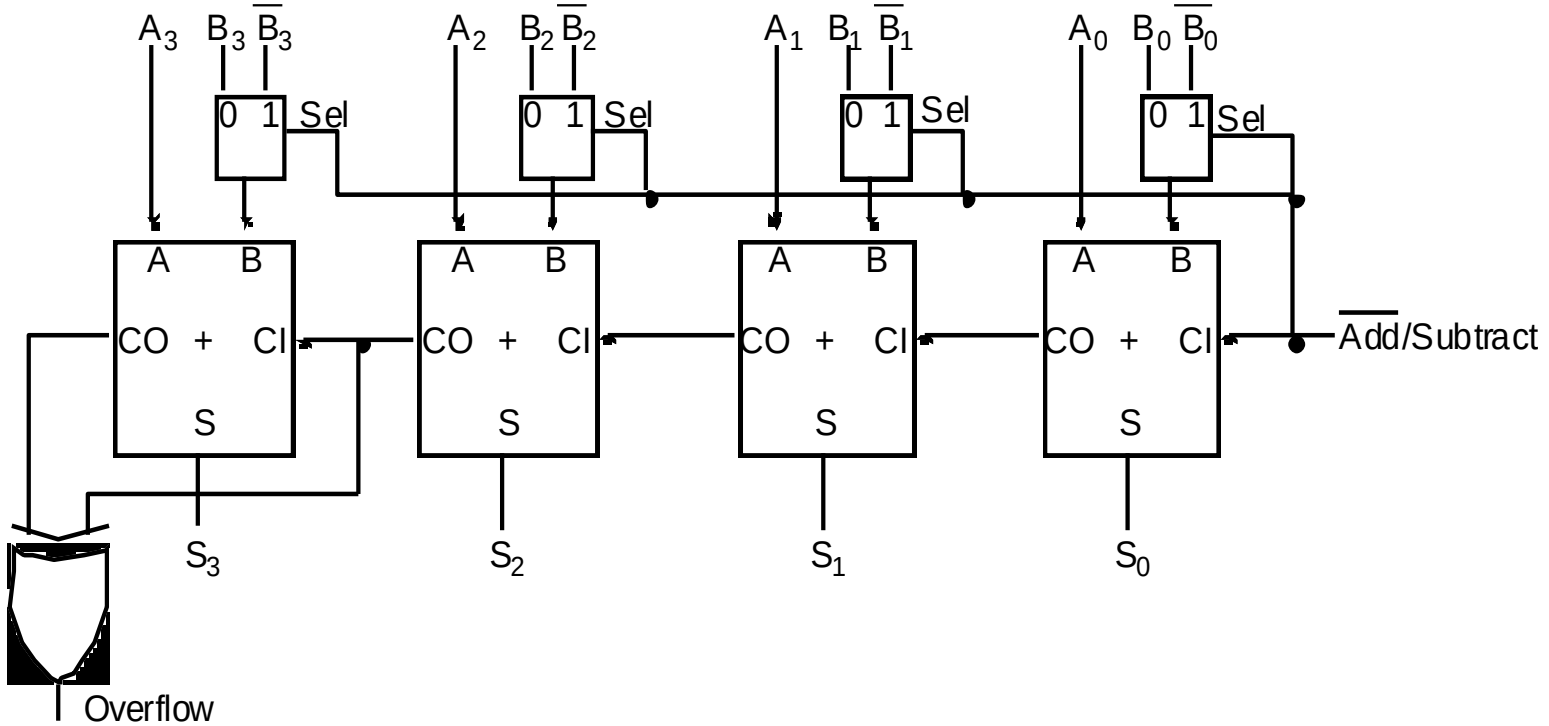




تفریق کننده (Subtractor)

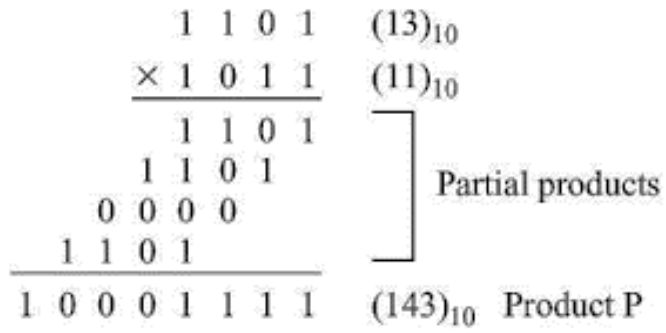
$$A - B = A + (-B) = A + B' + 1$$

- درنظر گرفتن ماژول مجزا برای تفریق مقرون به صرفه نیست
- طراحی واحد جمع/تفریق کننده



ضرب‌کننده (Multiplier)

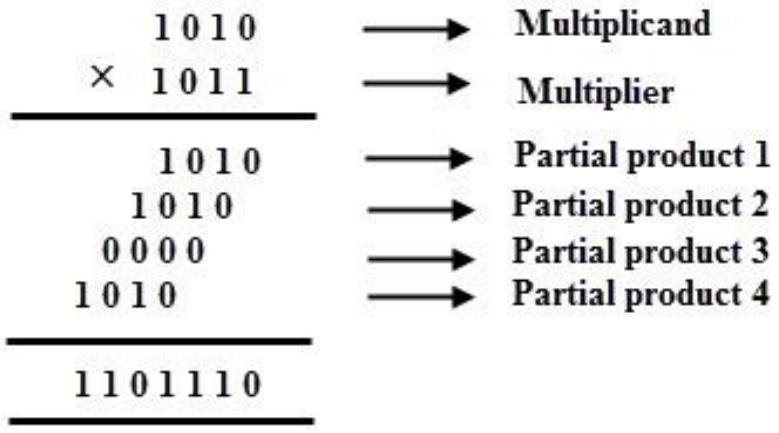
- چنانچه بخواهیم دو عدد n بیتی را درهم ضرب کنیم، بیشینه طول حاصل 2n می‌باشد
  - در هر مرحله از ضرب، یک بیت مضروب‌فیه را در مضروب ضرب می‌کنیم (Partial Product)
    - اگر بیت موردنظر صفر بود، حاصل صفر می‌شود
    - اگر بیت موردنظر یک بود، حاصل برابر مضروب می‌شود.
  - حاصلضرب میانی (partial product) یک بیت به چپ شیفت داده می‌شود



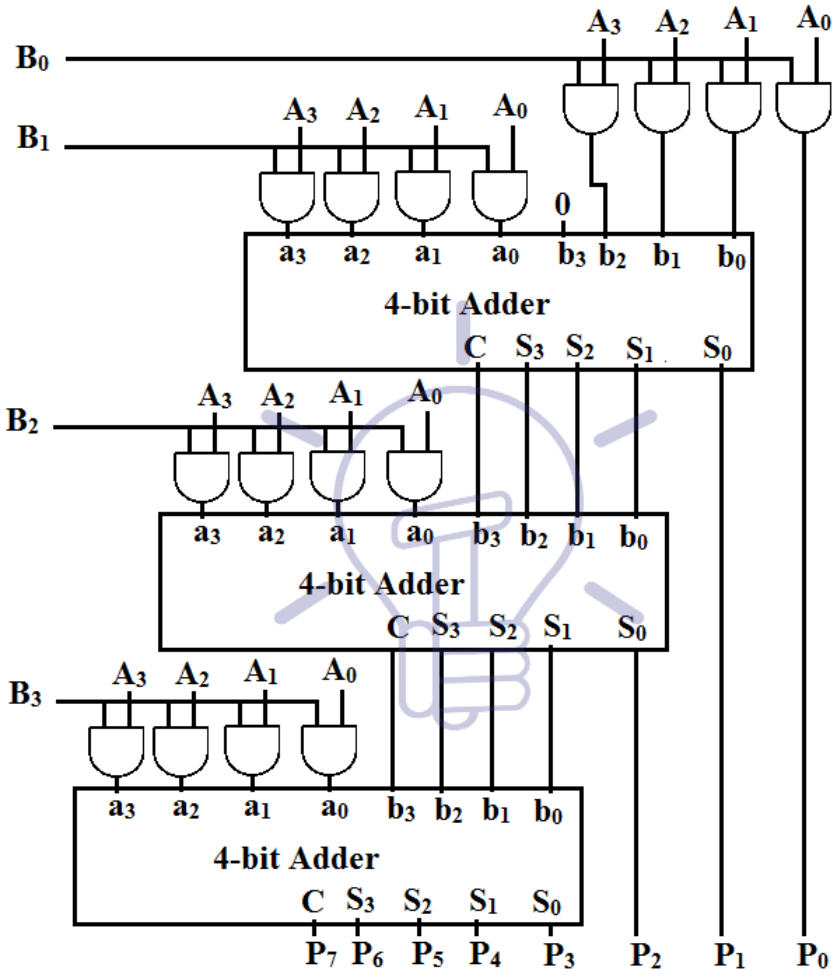
ضرب‌کننده (Multiplier)

- الگوریتم

- استخراج حاصل‌ضرب‌های میانی (ضرب بیت i ام B در A) و شیفت آن‌ها به به اندازه i بیت به چپ
- ذخیره تمامی حاصل‌ضرب‌های میانی
- جمع حاصل‌ضرب‌های میانی و ذخیره حاصل در 2n بیت



ضرب‌کننده (Multiplier)



ضرب‌کننده (Multiplier)

- جمع حاصل‌ضرب‌های میانی با
  - n-1 جمع‌کننده n بیتی (آبشاری)
  - Carry save adder برای جمع n عدد 2nبیتی
  - هزینه سخت‌افزاری و تاخیر زیاد:

HW Cost: (n-1)* Cost(n bit Adder) + n^2 * And Gate

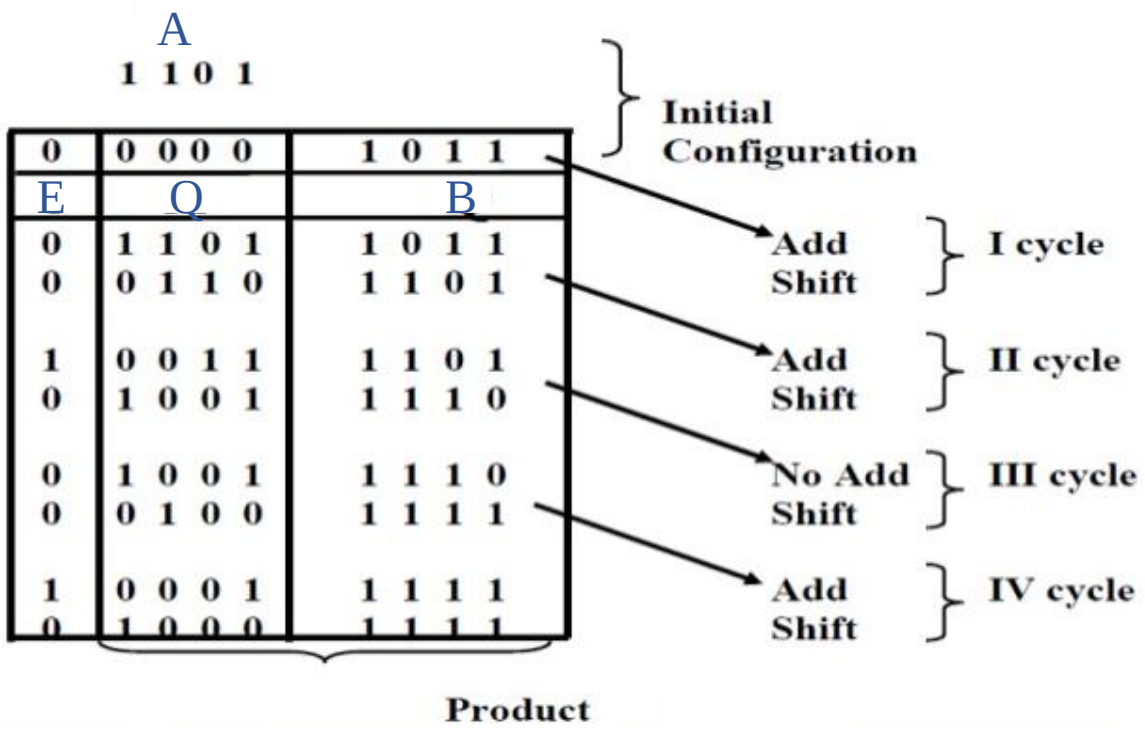
Delay: (n-1) * Delay (n bit Adder) + d

Serial Adders AND Gate

ضرب‌کننده ترتیبی

- **هدف:** کاهش تاخیر جمع‌کننده
- **الگوریتم:**
  - دو ثبات Q و A را به‌طول هریک nبیت به‌صورت متصل درنظر گرفته و مقدارهی اولیه (صفر) می‌کنیم
    - حاصلضرب نهایی در Q:A موجود است
  - هر حاصل‌ضرب میانی یا صفر است یا B که n بیتی است
    - در هر مرحله i لازم است B را i بیت شیفت داده و با بیت‌های متناظرش در Q:A جمع می‌کنیم
      - معادل این است که B را ثابت گرفته و Q:A را i واحد به راست شیفت دهیم و n بیت را با B جمع می‌کنیم
    - محاسبات با جمع‌کننده n بیتی قابل انجام است

ضرب‌کننده ترتیبی



13 * 11 = 143

- الگوریتم:

۱- تنظیم شمارنده به n

۲- چک کردن مقدار B_i

۲-۱- اگر Bi=1

A+Q → EQ

EQB را یک واحد به راست شیفت بده

۲-۲- اگر Bi=0

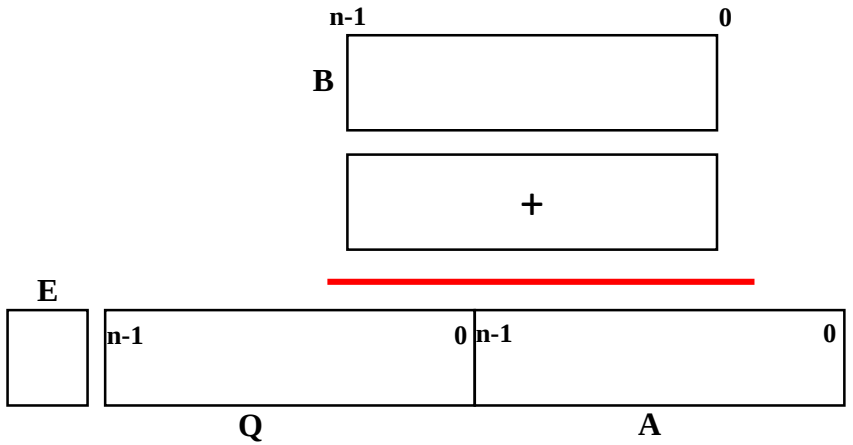
EQB را یک واحد به راست شیفت بده

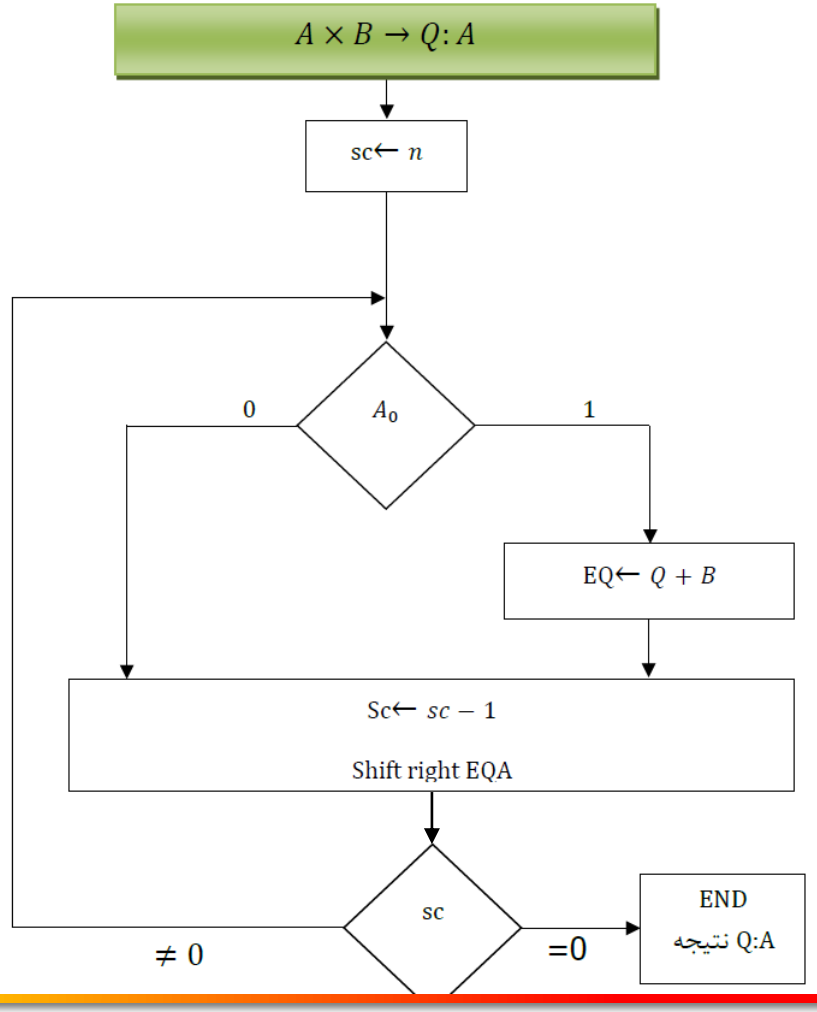
۳- یکی از شمارنده کم کن

۳-۱- اگر شمارنده صفر بود اتمام

۳-۲- اگر شمارنده صفر نبود به مرحله ۲ بازگرد

ضرب‌کننده ترتیبی

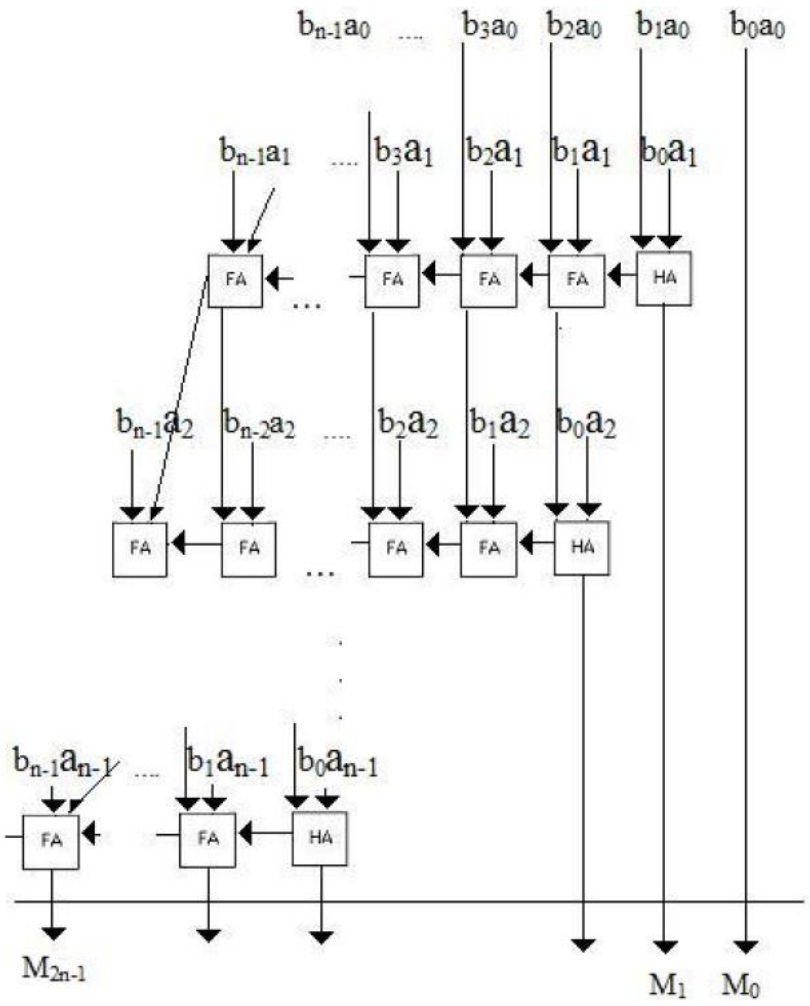




ضرب‌کننده ترتیبی

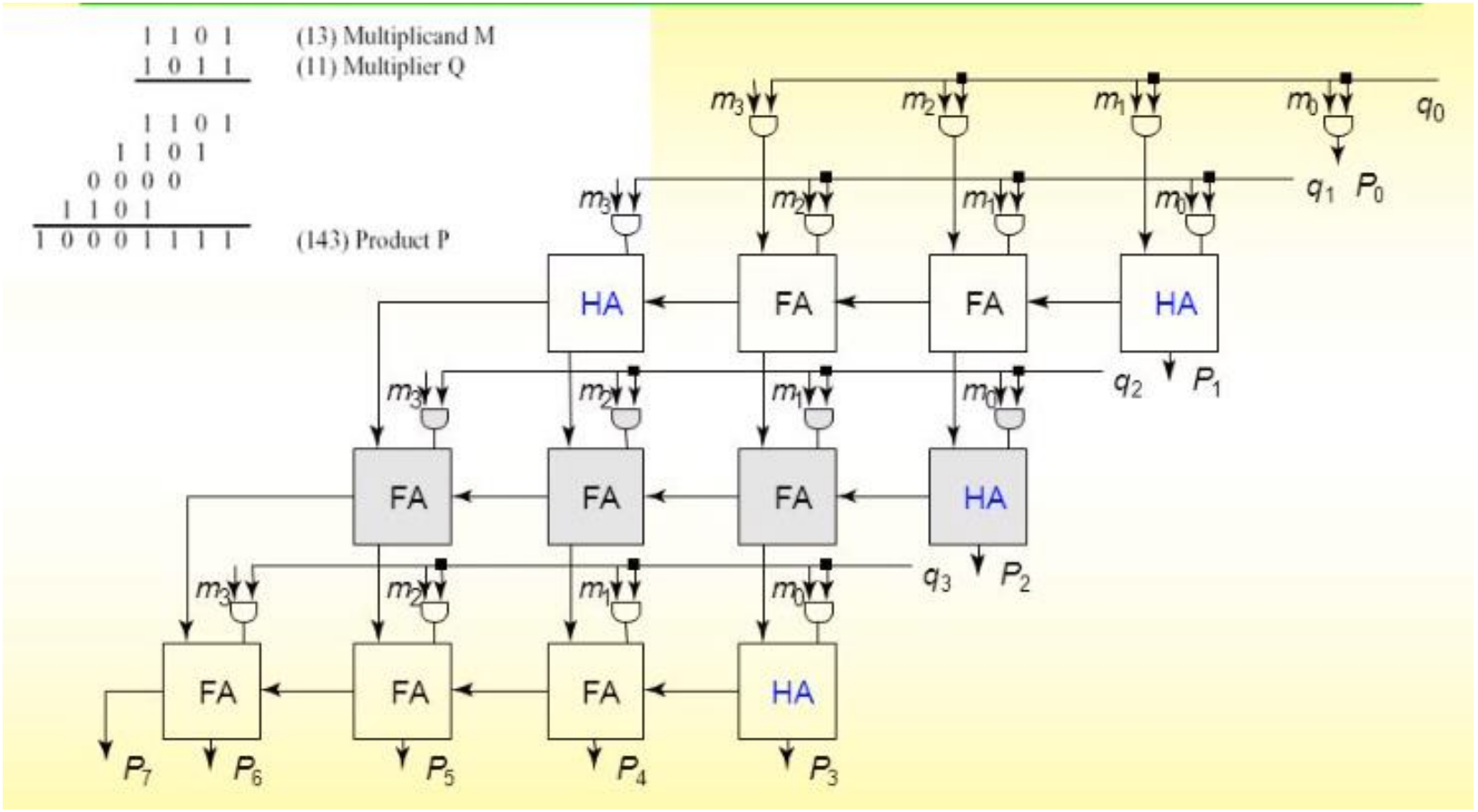
- بدبینانه‌ترین حالت:
  - همه بیت‌های B برابر یک باشند (جمع مداوم و تاخیر زیاد)
- خوش‌بینانه‌ترین حالت:
  - همه بیت‌های B برابر صفر باشند (نیاز به n کلاک برای شمارنده)
- هزینه سخت‌افزاری:
  - یک جمع‌کننده n بیتی، سه ثبات A، B و Q

ضرب‌کننده آرایه‌ای

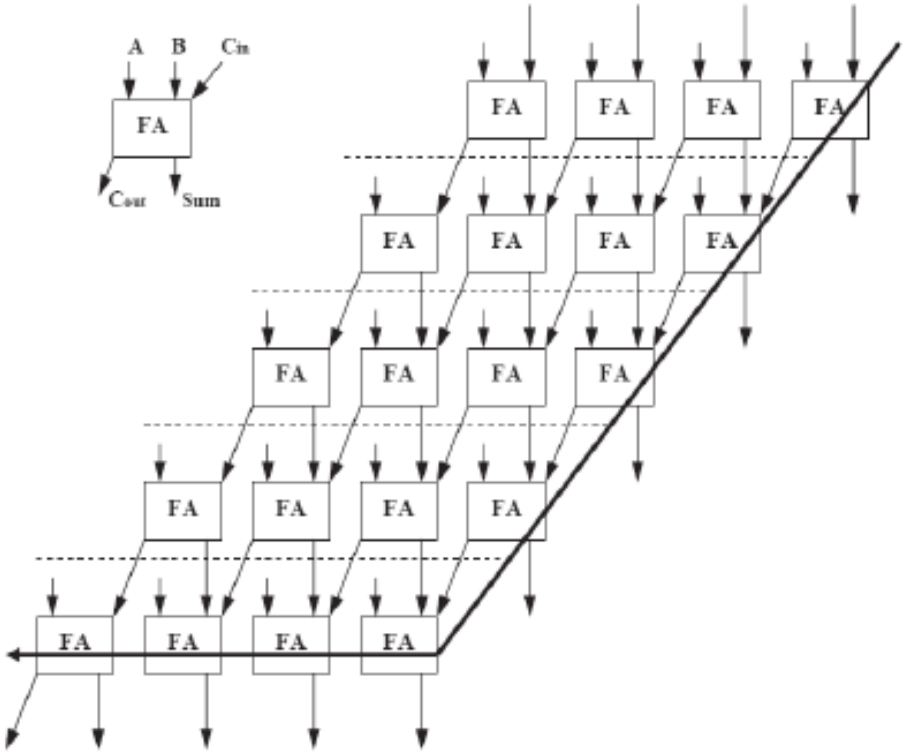


- **هدف:** پیاده‌سازی ضرب‌کننده به‌صورت ترکیبی
  - هرردیف ضرب Ax در بیت‌های B
  - جمع هر ردیف با هم‌ستون ردیف پایینش
    - انتقال sum به ردیف بعدی یا حاصلضرب نهایی
    - انتقال بیت نقلی به به ستون بعدی (جمع‌کننده هم‌ردیف سمت چپ)
  - قابلیت استفاده از نیم‌جمع‌کننده در برخی مکان‌ها

ضرب‌کننده آرایه‌ای



ضرب‌کننده آرایه‌ای

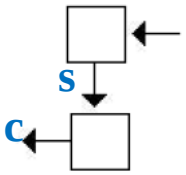


- طراحی دیگر:
  - انتقال بیت نقلی به ستون سمت چپ سطر پایین
    - شرکت دادن بیت نقلی در جمع ستون بعدی در هر دو حالت

هزینه و تاخیر ضرب‌کننده آرایه‌ای

- این ضرب‌کننده متشکل از n-1 جمع‌کننده سریال در سطرهاست
- **هزینه:** در مجموع n-1 سطر که هریک n جمع‌کننده دارند

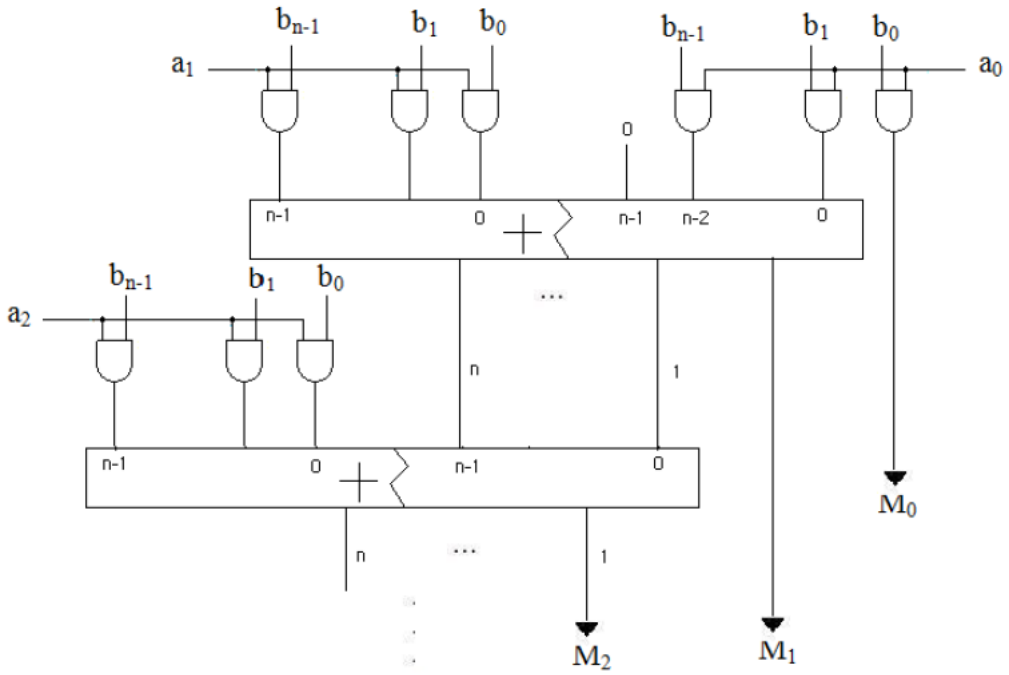
HW Cost = n*(n-1) * 5d + n^2 d ≈ O (n^2 FA)



- **تاخیر:** حرکت بیت‌های جمع و نقلی به‌صورت عمودی و افقی بین جمع‌کننده‌ها

Delay ≈ d +3n*delay (adder) ≈ O (n)

ضرب‌کننده آرایه‌ای



- طراحی دیگر با استفاده از جمع‌کننده‌های سریع
- جایگزین کردن FA های هر سطر با جمع‌کننده n بیتی
- جایگزین کردن جمع‌کننده آبشاری با CLA و ...

ضرب‌کننده بوث (Booth)

- در ضرب‌کننده آرایه‌ای
  - بسیاری از عبارات برابر صفر است و عملیات جمع اضافه باعث افزایش تاخیر می‌شود
- در ضرب‌کننده ترتیبی
  - به تعداد بیت‌های یک عدد دوم عملیات جمع انجام می‌دادیم
- مستقل کردن ضرب از تعداد یک‌ها و کاهش دادن تعداد عملیات جمع
  - معرفی ضرب‌کننده booth

ضرب‌کننده بوث (Booth)

- در این الگوریتم قطع توالی بین رشته‌های یک و صفر هدف است
  - اگر چندین یک پشت سر هم داشته باشیم: عملیات جمع
  - اگر چندین صفر پشت سر هم داشته باشیم: کاری نمی‌کنیم
  - اگر از صفر به یک یا از یک به صفر برویم: تغییر روال
    - درنتیجه دنباله‌های 10 و 01 در مضروب مهم هستند
    - تعداد عملیات جمع در حین ضرب توسط این دنباله‌ها تعیین می‌شوند

ضرب‌کننده بوث (Booth)

- در این الگوریتم سعی بر آن است که عملیات ضرب با شیفت انجام گیرد
- نوشتن اعداد به‌صورت توانی از دو و درجه‌دهی با یک یا منفی یک (کدگذاری بوث)
  - درجه اولین یک از سمت راست عدد را ۱- قرار می‌دهیم
  - تا انتهای توالی یک ادامه داده و درجه اولین صفر را ۱ قرار می‌دهیم
  - تا انتهای توالی صفر پیش رفته و درجه اولین یک بعدی را ۱- قرار می‌دهیم
  - و ...
- نکته: اگر پرارزش‌ترین بیت عدد، یک بود، آن را درجه‌گذاری نمی‌کنیم (علامت عدد)

ضرب‌کننده بوث (Booth)

- نمایش اعداد در کدگذاری بوث

$+15: 00001111 = 2^4 - 2^0$

1 -1

$-10: 11110110 = -2^4 + 2^3 - 2^1$

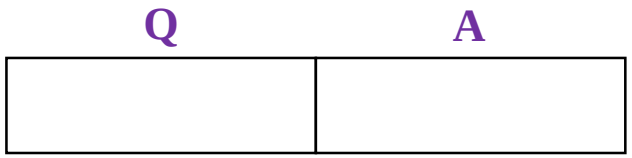
-1 1 -1

$01110101 = (2^7 - 2^4) + (2^3 - 2^2) + (2^2 - 2^0)$

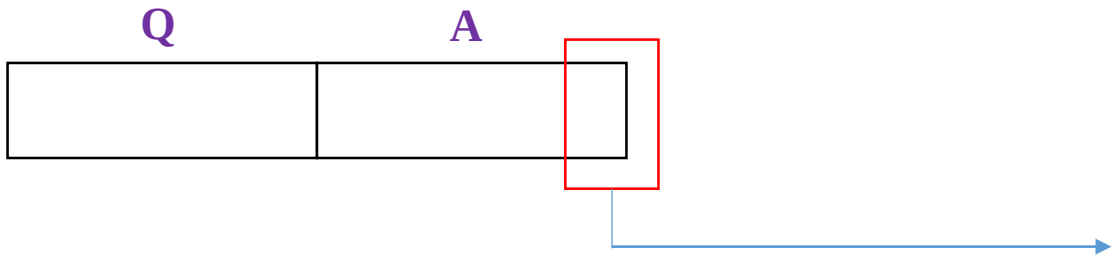
1 -1 1 -1 1 -1

پیاده‌سازی ضرب‌کننده بوث (Booth)

- در این الگوریتم تغییر مقدار مهم است
  - الگوهای 10 یا 01
- با متصل کردن A و Q یک آرایه 2n بیتی می‌سازیم
  - خروجی نهایی در این آرایه ذخیره می‌شود (مشابه ضرب‌کننده ترتیبی)
  - مقداردهی اولیه به صفر



پیاده‌سازی ضرب‌کننده بوث (Booth)

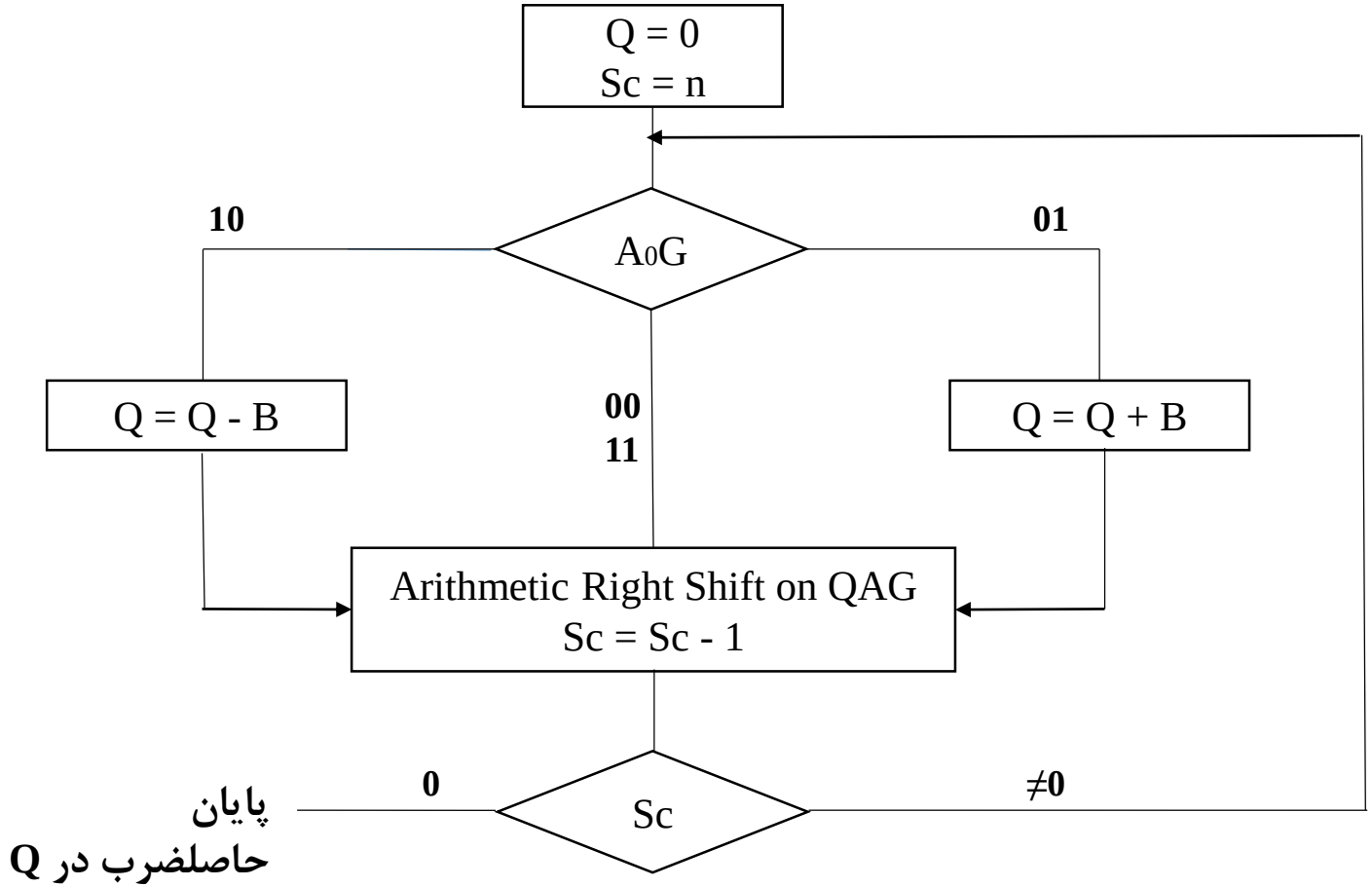


- در هر مرحله از ضرب
  - دو بیت سمت راست A را چک می‌کنیم
    - **00 یا 11:** به سمت راست شیفت می‌دهیم
    - **10:** Q - B → Q.A
    - **01:** Q + B → Q.A

پیاده‌سازی ضرب‌کننده بوث (Booth)

- بدترین حالت در ضرب بوث:
  - رشته عدد بصورت یک در میان صفر و یک باشد (...1010 یا ...0101)
  - مجبوریم همواره عملیات جمع انجام دهیم (از درجه n)
- برای پیاده‌سازی ضرب بوث
  - یک بیت در سمت راست آرایه Q.A قرار می‌دهیم (G)
    - در هر مرحله A_0G را با چک کرده و تصمیم می‌گیریم

الگوریتم ضرب‌کننده بوث (Booth)



الگوریتم ضرب‌کننده بوث (Booth)

مثال: دو عدد روبرو را توسط الگوریتم booth در یکدیگر ضرب کنید.

A = 010101
B = 001110

حل:

QAG = 00000000011100 ,

Sc = 6 → A_0G = 00 → QAG = 00000000001110

Sc = 5 → A_0G = 10 → Q = Q - A = 000000 + 101011 = 101011

QAG = 1010110001110 →1101011000111

Sc = 4 → A_0G = 11 → Q = 1110101100011

Sc = 3 → A_0G = 11→ Q = 1111010110001

Sc = 2 → A_0G = 01 → Q = Q + A = 111101 + 010101 = 010010

QAG = 0100100110001→ 0010010011000

Sc = 1 → A_0G = 00 → QAG = 0001001001100

Sc = 0 → Finish , Q = 001001001100 ≈ 294