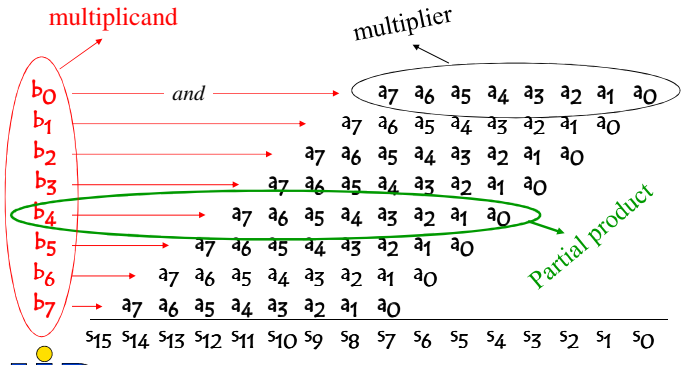
Outline

- Digital CMOS Design
- Arithmetic Operators
  - Adders
  - Comparators
  - Shifters
  - Multipliers

Multipliers



Multipliers

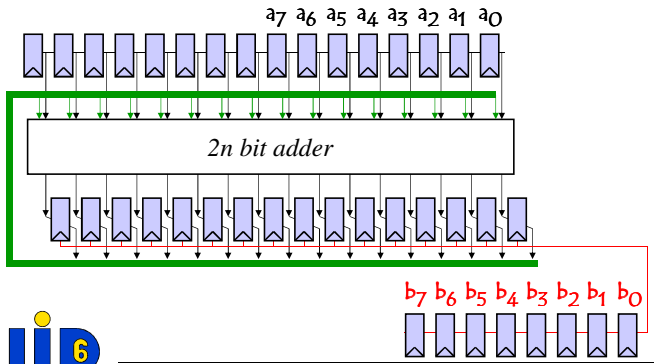
Two natural numbers A and B coded on n bits

the result of A × B is coded on 2n bits

Classic method as learned in primary school

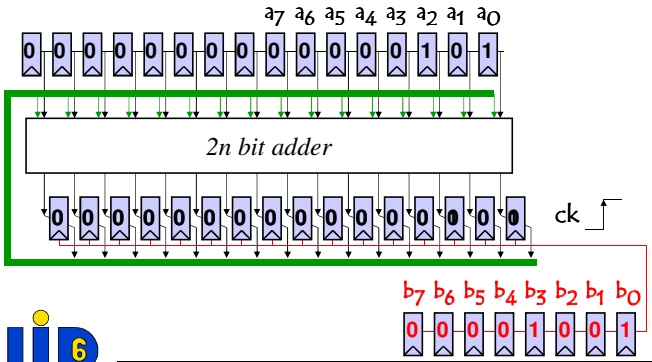
Multipliers

Implementation : sequential multiplier



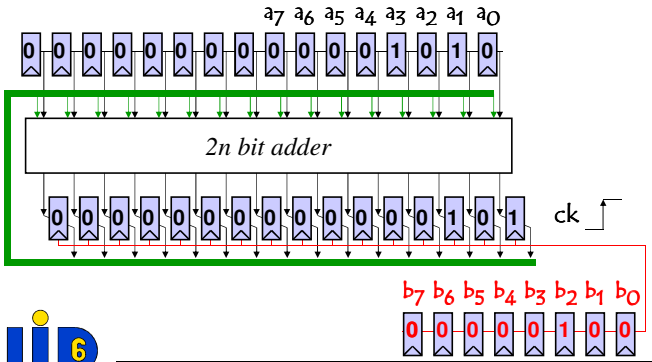
Multipliers

Implementation : sequential multiplier



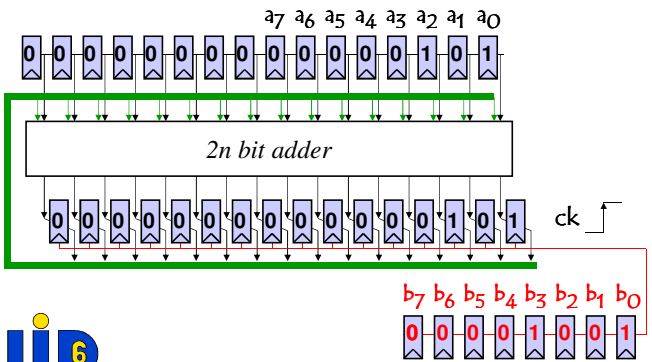
Multipliers

Implementation : sequential multiplier



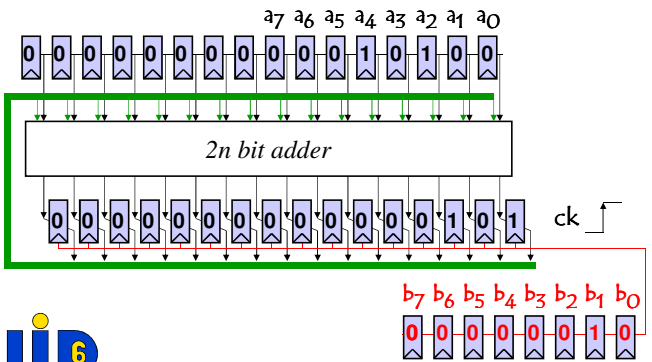
Multipliers

Implementation : sequential multiplier



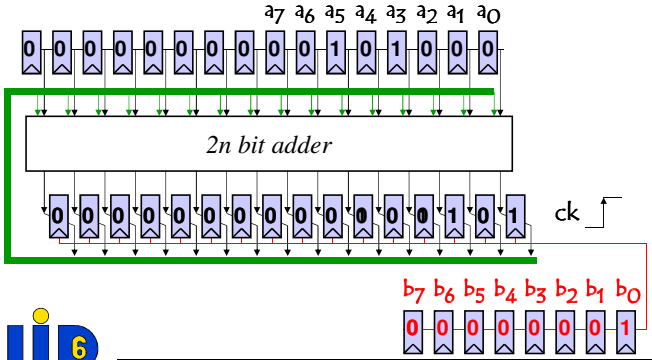
Multipliers

Implementation : sequential multiplier



Multipliers

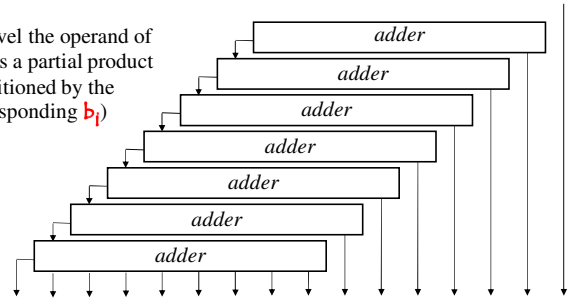
Implementation : sequential multiplier



Multipliers

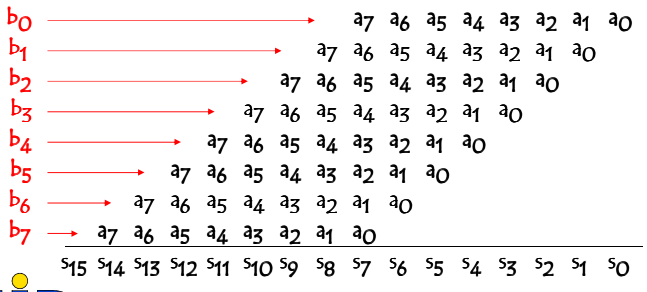
Implementation : parallel multiplier

At each level the operand of the adder is a partial product (conditioned by the corresponding b_i)



Multipliers

Implementation : parallel multiplier



Multipliers

Implementation : parallel multiplier

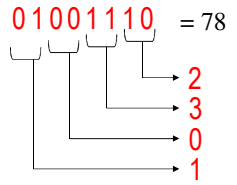
Improvement : Reduce the number of partial products

$$B = \sum_{i=0}^{n-1} b_i \times 2^i \qquad b_i \in \{0,1\}$$

$$B = \sum_{i=0}^{n/2-1} (b_i + 2b_{i+1}) \times 2^{2i}$$

half less terms

→ half less partial products



$$B = 2 \times 2^0 + 3 \times 2^2 + 0 \times 2^4 + 1 \times 2^6 = 78$$

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$B = \sum_{i=0}^{n-1} b_i \times 2^i$$

$$B = b_0 \times 2^0 + b_1 \times 2^1 + b_2 \times 2^2 + b_3 \times 2^3 + b_4 \times 2^4 + b_5 \times 2^5 + b_6 \times 2^6 + b_7 \times 2^7$$

$$B = \sum_{i=0}^{n/2-1} (b_i + 2b_{i+1}) \times 2^{2i}$$

$$B = (b_0 + 2b_1) \times 2^0 + (b_2 + 2b_3) \times 2^2 + (b_4 + 2b_5) \times 2^4 + (b_6 + 2b_7) \times 2^6$$

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$B = \sum_{i=0}^{n/2-1} (b_i + 4b_{i+1} - 2b_{i+1}) \times 2^{2i}$$

$$B = (b_0 + 4b_1 - 2b_1) \times 2^0 + (b_2 + 4b_3 - 2b_3) \times 2^2 + (b_4 + 4b_5 - 2b_5) \times 2^4 + (b_6 + 4b_7 - 2b_7) \times 2^6$$

$$B = (b_0 + 0 - 2b_1) \times 2^0 + (b_2 + b_1 - 2b_3) \times 2^2 + (b_4 + b_3 - 2b_5) \times 2^4 + (b_6 + b_5 - 2b_7) \times 2^6 + (\quad b_7 \quad) \times 2^8$$

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$B = \sum_{i=0}^{n-1} b_i \times 2^i \qquad b_i \in \{0, 1\}$$

$$B = \sum_{i=0}^{n/2-1} (b_i + 2b_{i+1}) \times 2^{2i} \qquad \in \{0, 1, 2, \mathbf{3}\}$$

$2 = 4 - 2$!!

$$B = \sum_{i=0}^{n/2-1} (b_i + 4b_{i+1} - 2b_{i+1}) \times 2^{2i}$$

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$B = \sum_{i=0}^{n-1} b_i \times 2^i \qquad b_i \in \{0, 1\}$$

$$B = \sum_{i=0}^{n/2-1} (b_i + 4b_{i+1} - 2b_{i+1}) \times 2^{2i}$$

next weight

previous weight

$$B = \sum_{i=0}^{n/2} (b_{i-1} + b_i - 2b_{i+1}) \times 2^{2i}$$

$$B = \sum_{i=0}^{n/2} b'_i \times 2^{2i} \qquad b'_i \in \{-2, -1, 0, 1, 2\}$$

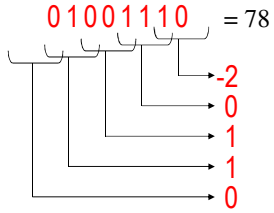
Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$B = \sum_{i=0}^{n/2} (b_{i-1} + b_i - 2b_{i+1}) \times 2^{2i}$$

Booth encoding



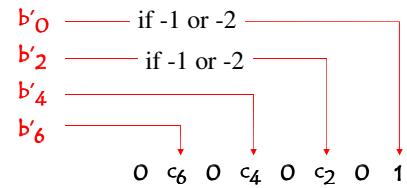
$$B = -2 \times 2^0 + 0 \times 2^2 + 1 \times 2^4 + 1 \times 2^6 + 0 \times 2^8 = 78$$

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

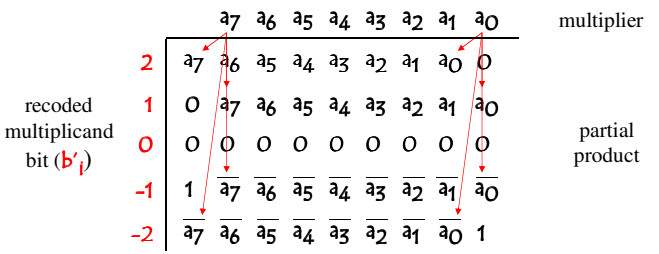
An additional partial product is generated to take into account the input carry in case of subtraction



Multipliers

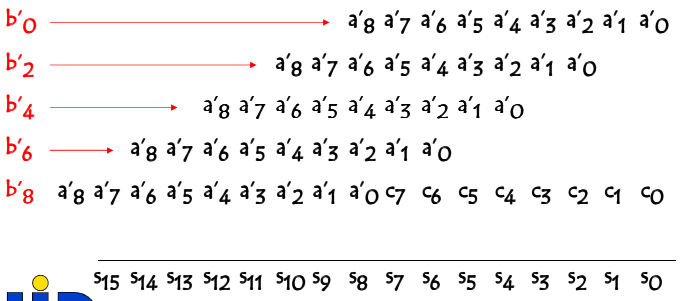
Implementation : parallel multiplier

Improvement : Reduce the number of partial products



Multipliers

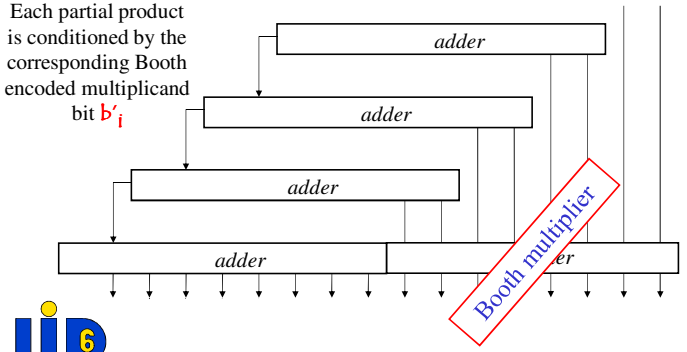
Implementation : parallel multiplier



Multipliers

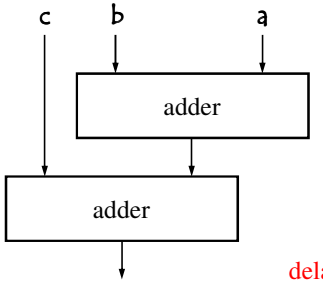
Implementation : parallel multiplier

Each partial product is conditioned by the corresponding Booth encoded multiplicand bit b'_i



Multipliers

Adding three natural numbers



delay

Multipliers

Implementation : fast parallel multiplier

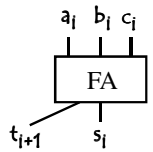
Basically for a n×n multiplication, we have to add n partial products (2n-bit numbers)

Multipliers

Adding three natural numbers

$s_i = a_i \oplus b_i \oplus c_i$ $\quad c_{i+1} = a_i.b_i + a_i.c_i + b_i.c_i$

the expressions are symmetrical in regard of a, b and c

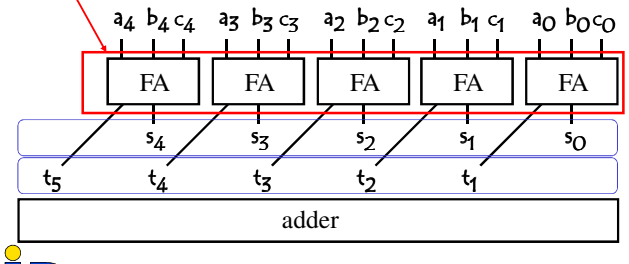


A full adder creates 2 numbers from 3

Multipliers

Adding three natural numbers

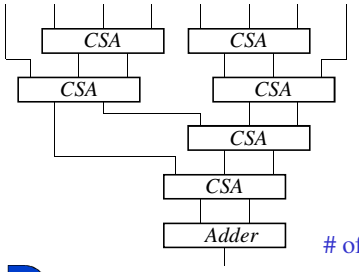
Carry Save Adder (CSA)



Multipliers

Implementation : fast parallel multiplier

8 partial products



Wallace multiplier

# of CSA layers ≈ $\log_{3/2}$ (n/2)

Multipliers

Implementation : fast parallel multiplier

Basically for a n×n multiplication,
we have to add n partial products
(2n-bit numbers)

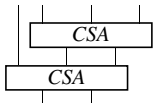
⇨ Use CSA (Carry Save Adder)
to reduce 3 partial products
into 2

Multipliers

Implementation : fast parallel multiplier

32×32 bits multiplier 32 → 22 → 15 → 10 → 7 → 5 → 4 → 3 → 2

Using 4 → 2 reduction leads to a more
regular hardware implementation



32×32 bits multiplier 32 → 16 → 8 → 4 → 2

# of CSA layers ≈ 2 log (n/2)

Multipliers

Implementation : fast parallel multiplier

32×32 bits multiplier 32 → 22 → 15 → 10 → 7 → 5 → 4 → 3 → 2

2 → 3 → 4 → 6 → 9 → 13 → 19 → 28 → 42

The margin on the number of 'partial products' can be used to extended the functionality of the multiplier

Multipliers

Multiply and add

$P = A \times B + C$

if $B < 0$?

$P = A \times B + C = (-A) \times (-B) + C$

$P = (\bar{A} + 1) \times (\bar{B} + 1) + C$

$P = \bar{A} \times \bar{B} + \bar{A} + \bar{B} + C + 1$

Multipliers

Two relative numbers A and B coded on n bits

$P = A \times B$

How it works if $B < 0$?

$P = A \times B = (-A) \times (-B) = (\bar{A} + 1) \times (\bar{B} + 1)$

$P = \bar{A} \times \bar{B} + \bar{A} + \bar{B} + 1$

Multipliers

Multiply and subtract

$P = C - A \times B$

$P = C + A \times (-B)$

if $B > 0$?

$P = C + A \times (-B) = C + A \times \bar{B} + A$