

2384-8

**ICTP Latin-American Advanced Course on FPGA Design for Scientific  
Instrumentation**

*19 November - 7 December, 2012*

**Digital Design  
(Part 8)**

BAZARGAN SABET Pirouz  
*Universite Pierre Et Marie Curie (Vi)  
Lip6, Departement Asim  
4, Place Jussieu  
75252 Paris Cedex 05  
FRANCE*

# Outline

- Digital CMOS Design

- Arithmetic Operators

  - Adders

  - Comparators

  - Shifters

  - Multipliers



# Multipliers

Two natural numbers  $a$  and  $b$  coded on  $n$  bits

the result of  $a \times b$  is coded on  $2n$  bits

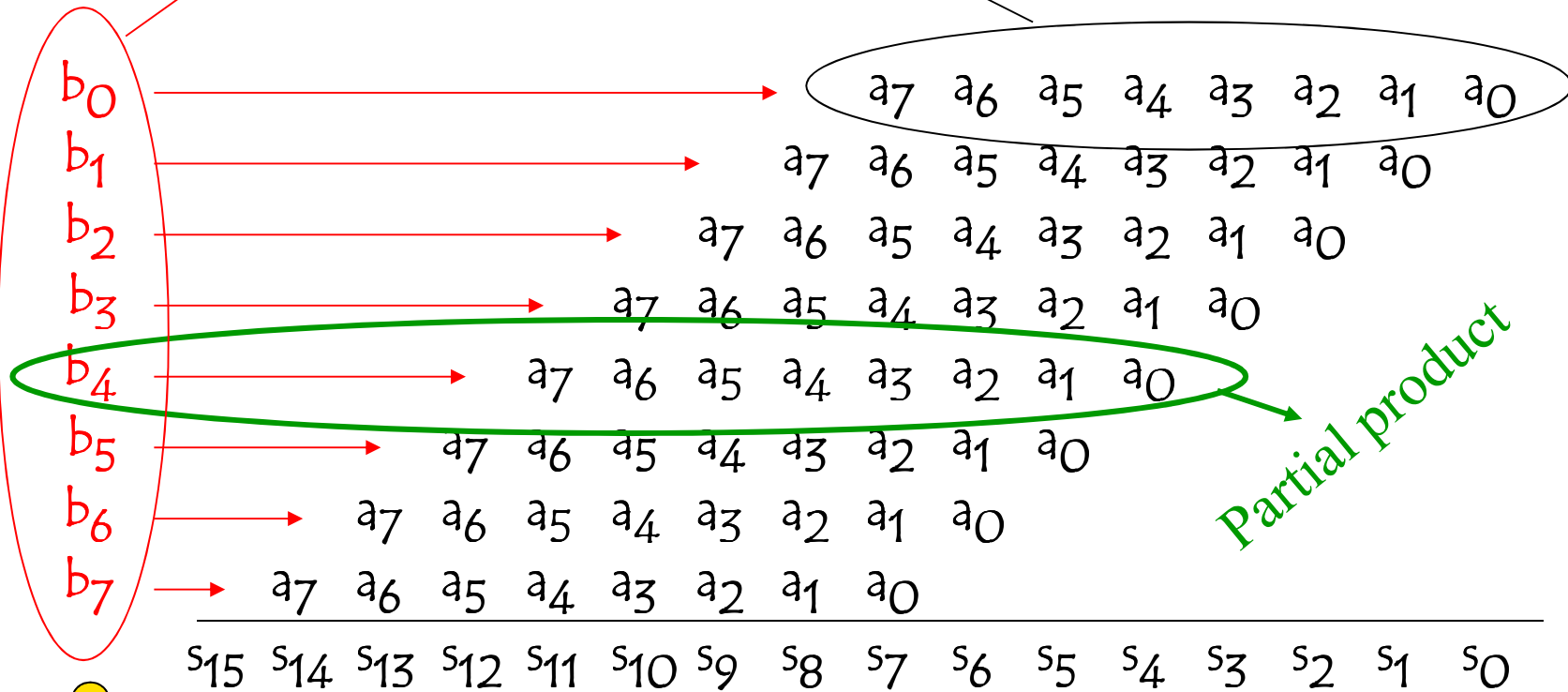
Classic scholar multiplication



# Multipliers

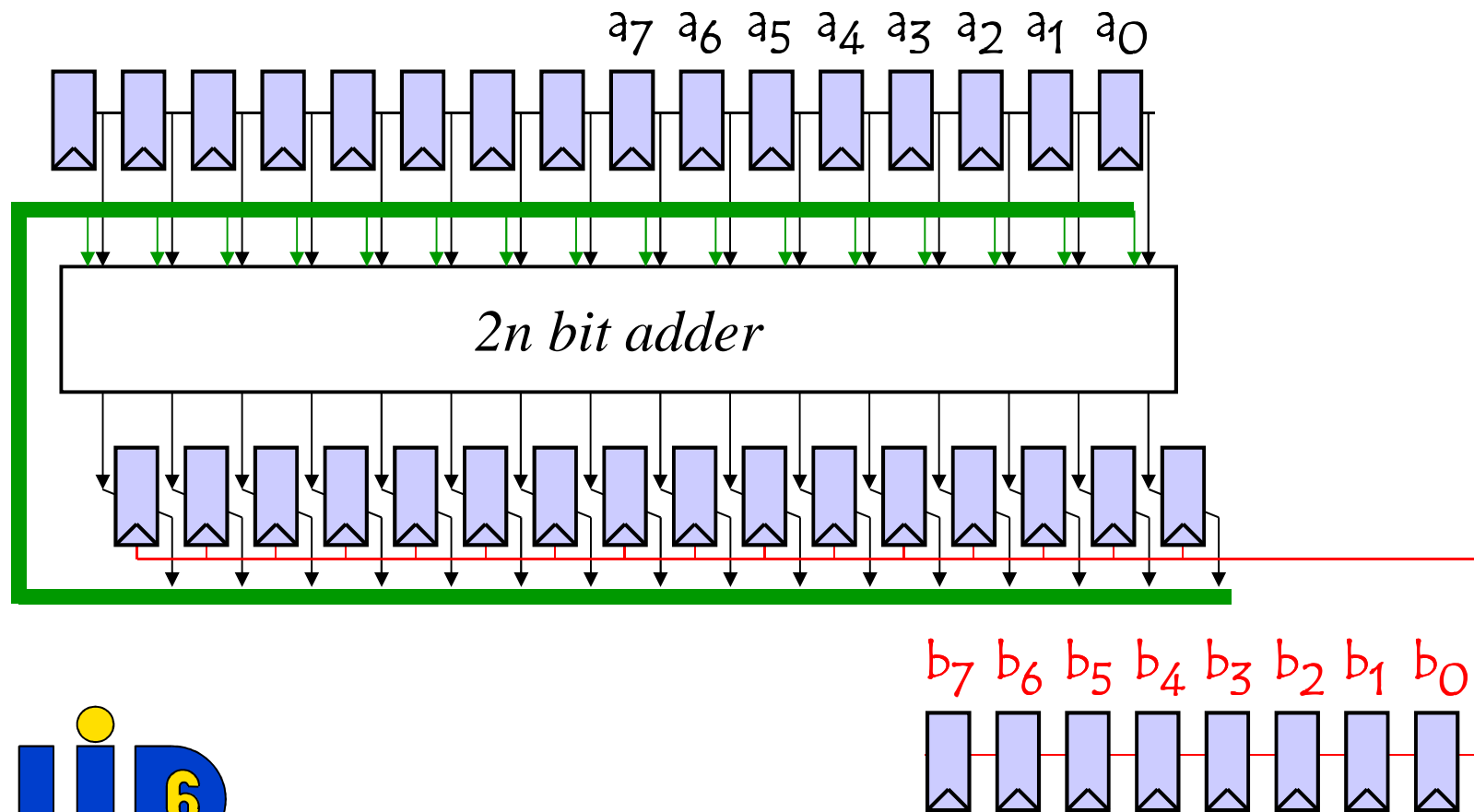
multiplicand

multiplier



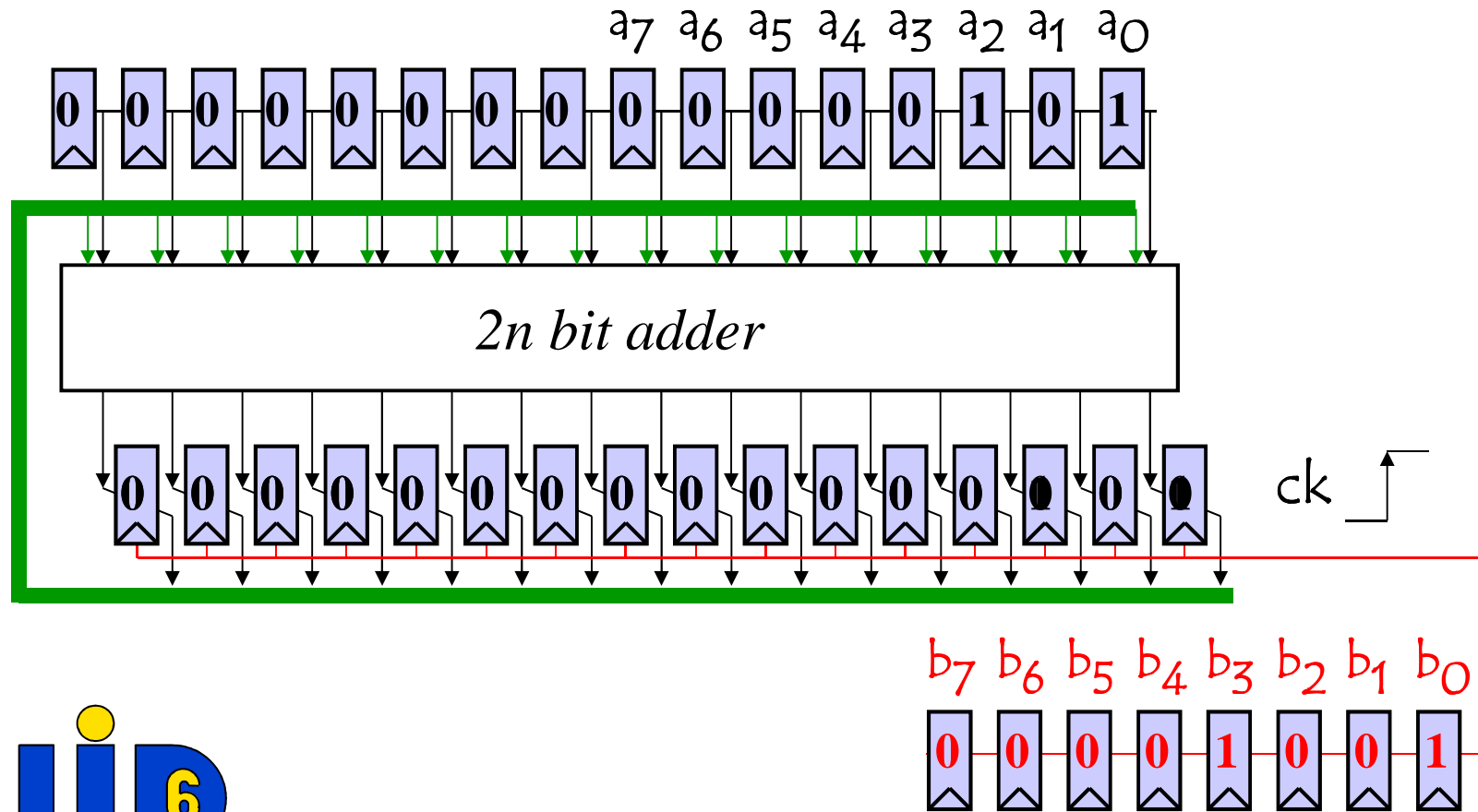
# Multipliers

Implementation : sequential multiplier



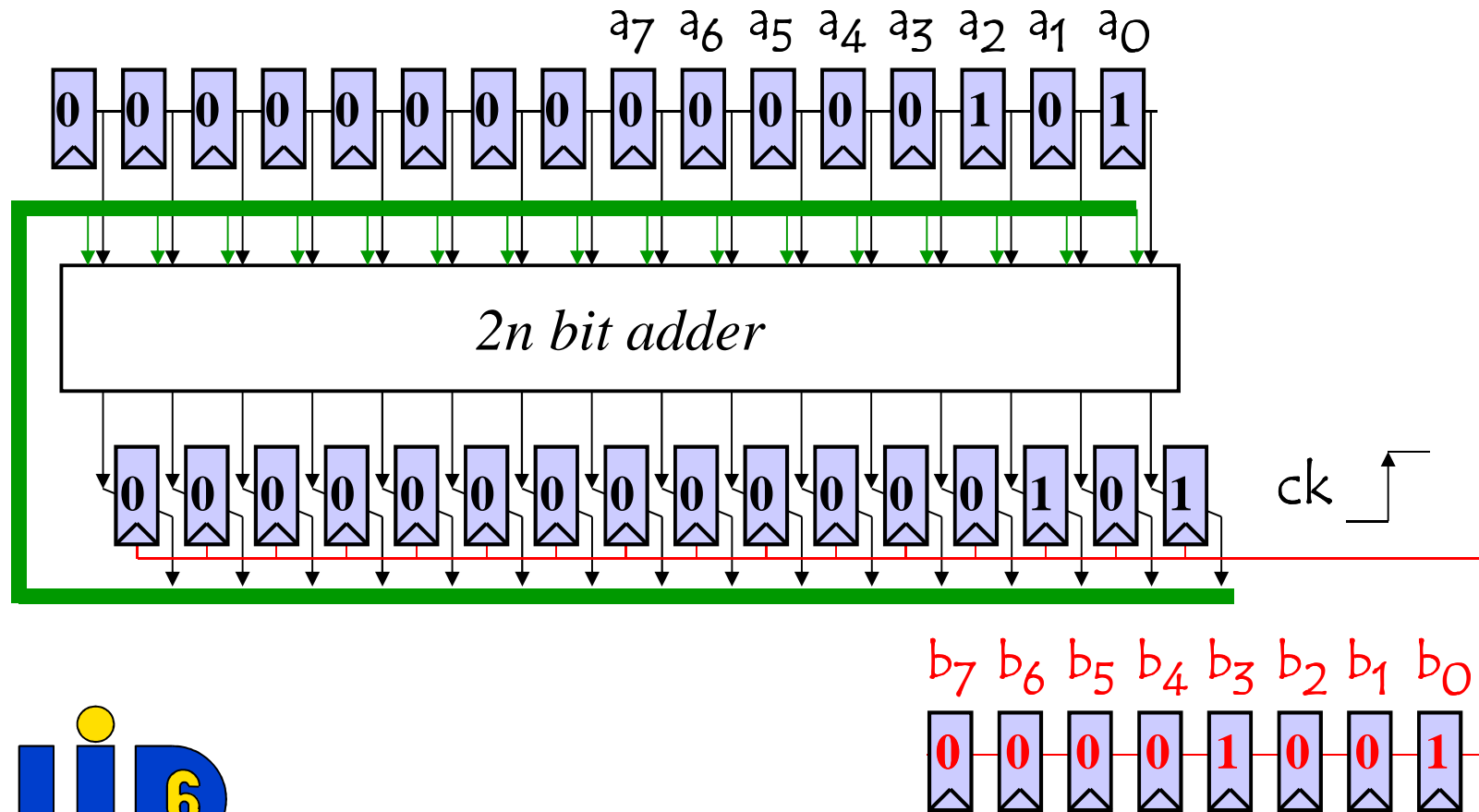
# Multipliers

Implementation : sequential multiplier



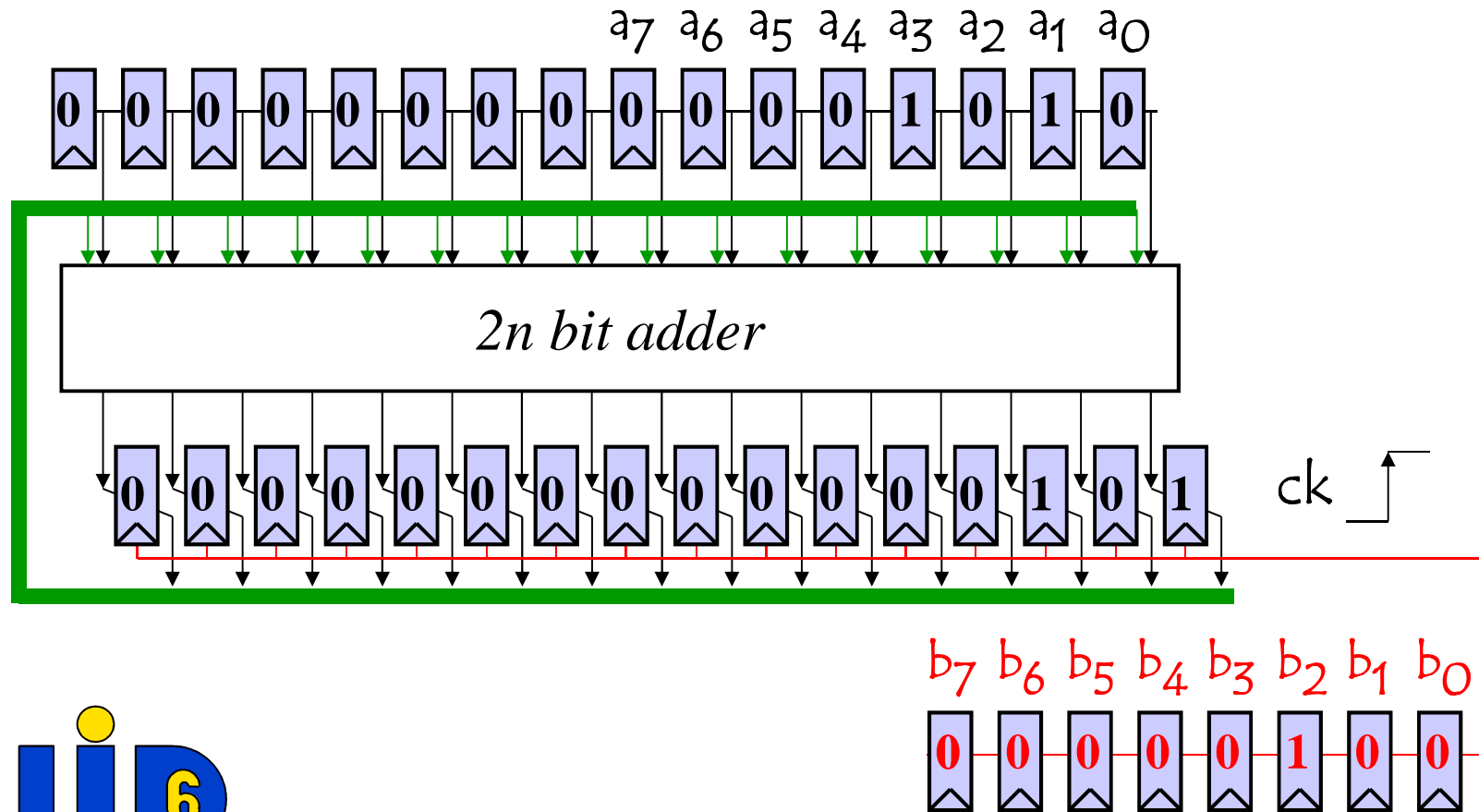
# Multipliers

Implementation : sequential multiplier



# Multipliers

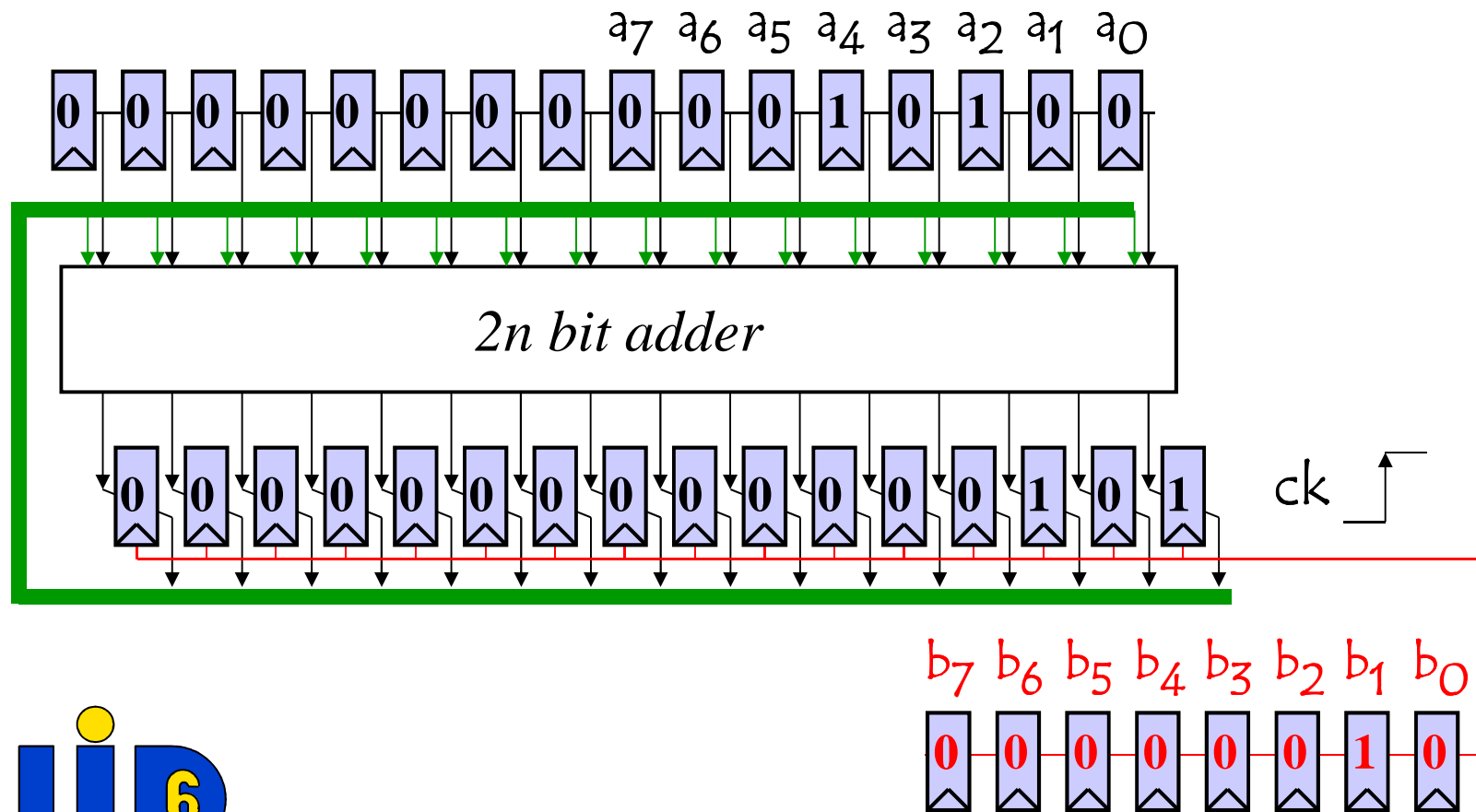
Implementation : sequential multiplier





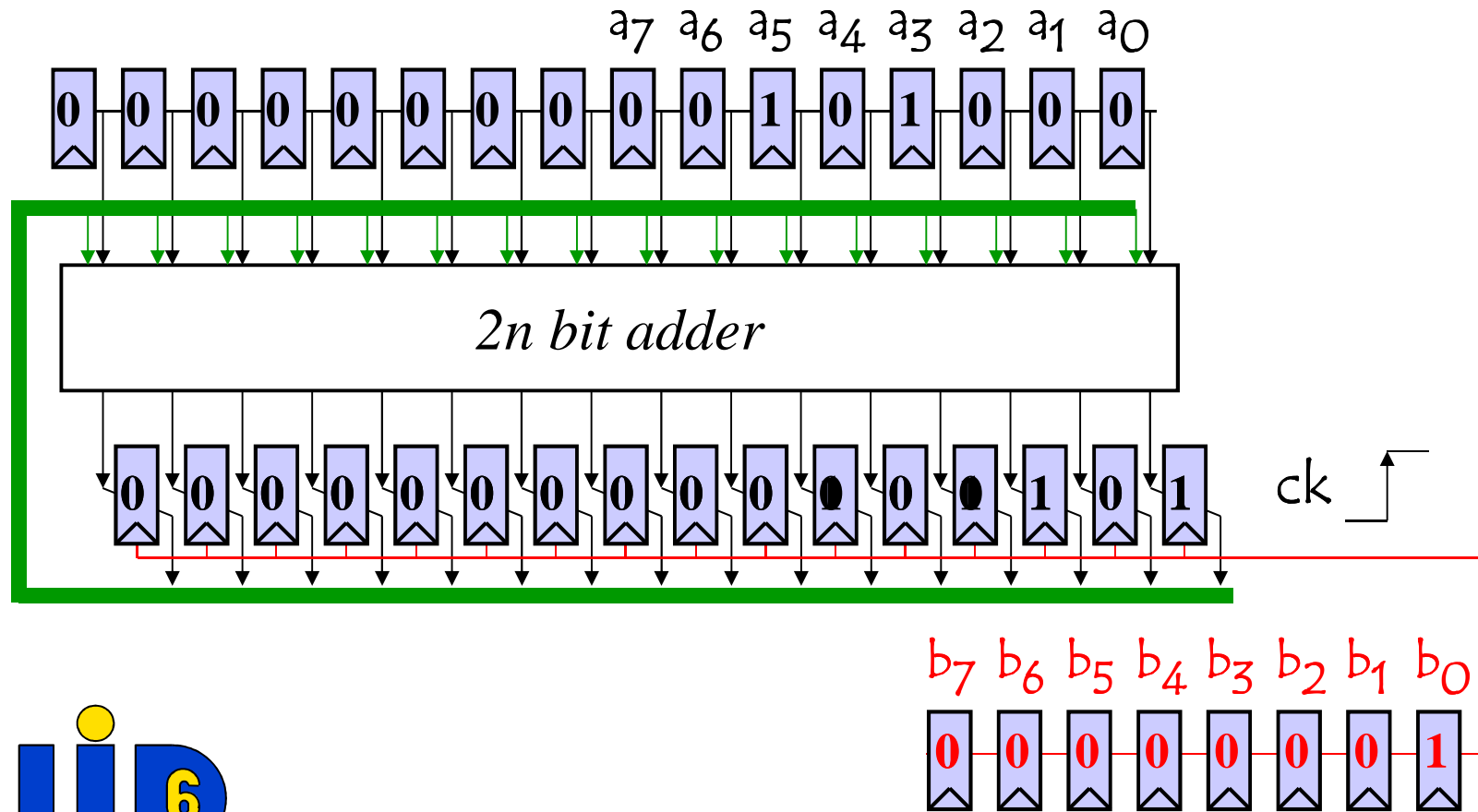
# Multipliers

Implementation : sequential multiplier



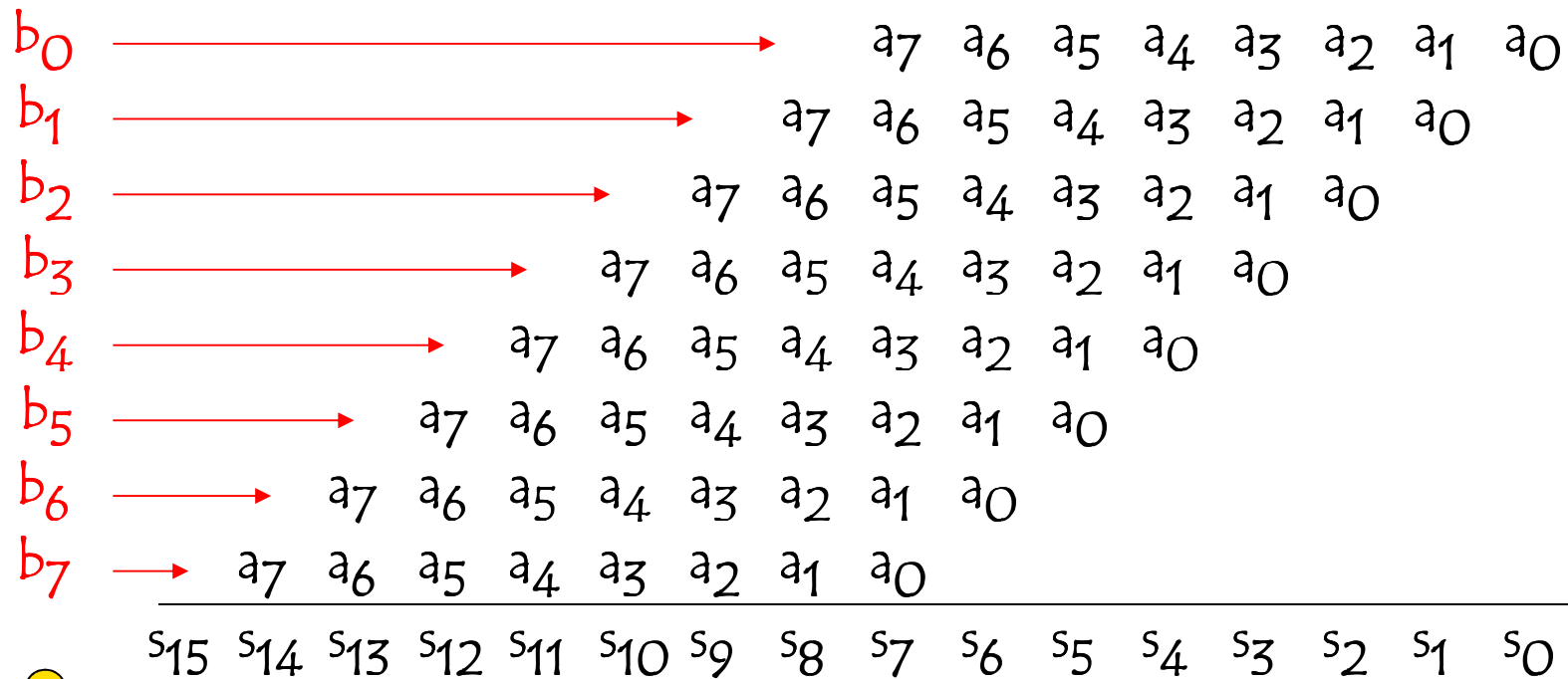
# Multipliers

Implementation : sequential multiplier



# Multipliers

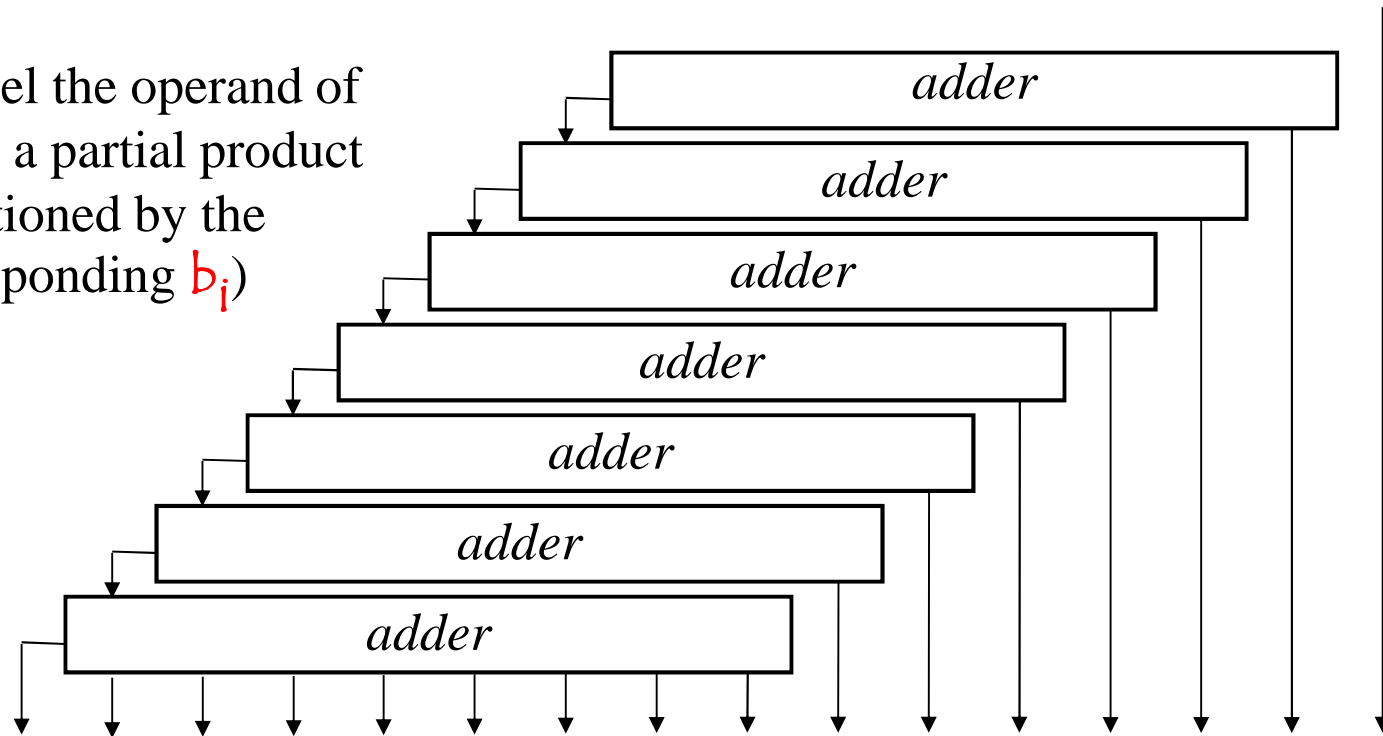
Implementation : parallel multiplier



# Multipliers

## Implementation : parallel multiplier

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



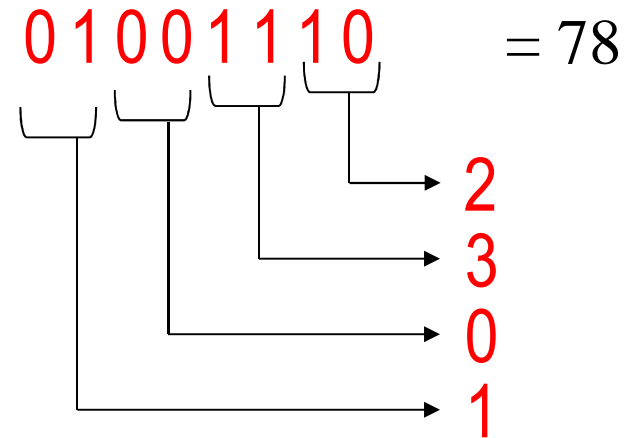
# Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$x = \sum_{i=0}^{n-1} x_i \times 2^i \quad x_i \in \{0, 1\}$$

$$x = \sum_{i=0}^{n/2-1} (x_{2i} + 2x_{2i+1}) \times 2^{2i}$$



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



# Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$x = \sum_{i=0}^{n-1} x_i \times 2^i \quad x_i \in \{0, 1\}$$

$$x = \sum_{i=0}^{n/2-1} (x_{2i} + 2x_{2i+1}) \times 2^{2i} \quad \in \{0, 1, 2, 3\}$$

$$2 = 4 - 2$$

$$x = \sum_{i=0}^{n/2-1} (x_{2i} + 4x_{2i+1} - 2x_{2i+1}) \times 2^{2i}$$



# Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$x = \sum_{i=0}^{n/2-1} (x_{2i} + 4x_{2i+1} - 2x_{2i+2}) \times 2^{2i}$$

$$\begin{aligned} x = & (x_0 + 4x_1 - 2x_2) \quad + \\ & (x_4 + 4x_5 - 2x_6) \times 4 + \\ & (x_8 + 4x_9 - 2x_{10}) \times 16 + \dots \end{aligned}$$

$$\begin{aligned} x = & (x_0 + 0 - 2x_1) \quad + \\ & (x_4 + x_5 - 2x_6) \times 4 + \\ & (x_8 + x_9 - 2x_{10}) \times 16 + \dots \end{aligned}$$



# Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$x = \sum_{i=0}^{n-1} x_i \times 2^i \quad x_i \in \{0, 1\}$$

$$x = \sum_{i=0}^{n/2-1} (x_{2i} + 4x_{2i+1} - 2x_{2i+2}) \times 2^{2i} \quad \text{next weight}$$

$$x = \sum_{i=0}^{n/2-1} (x_{2i-1} + x_{2i} - 2x_{2i+1}) \times 2^{2i}$$

$$x = \sum_{i=0}^{n/2-1} x'_{2i} \times 2^{2i} \quad x'_i \in \{-2, -1, 0, 1, 2\}$$





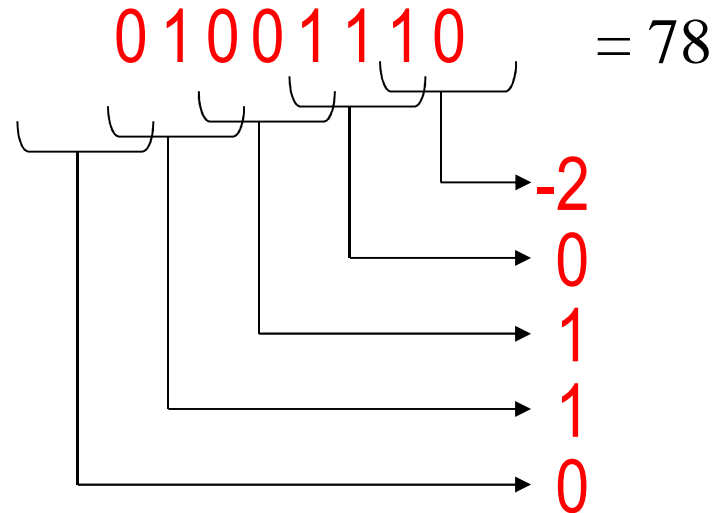
# Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

$$x = \sum_{i=0}^{n/2-1} (x_{2i-1} + x_{2i} - 2x_{2i+1}) \times 2^{2i}$$

Booth encoding

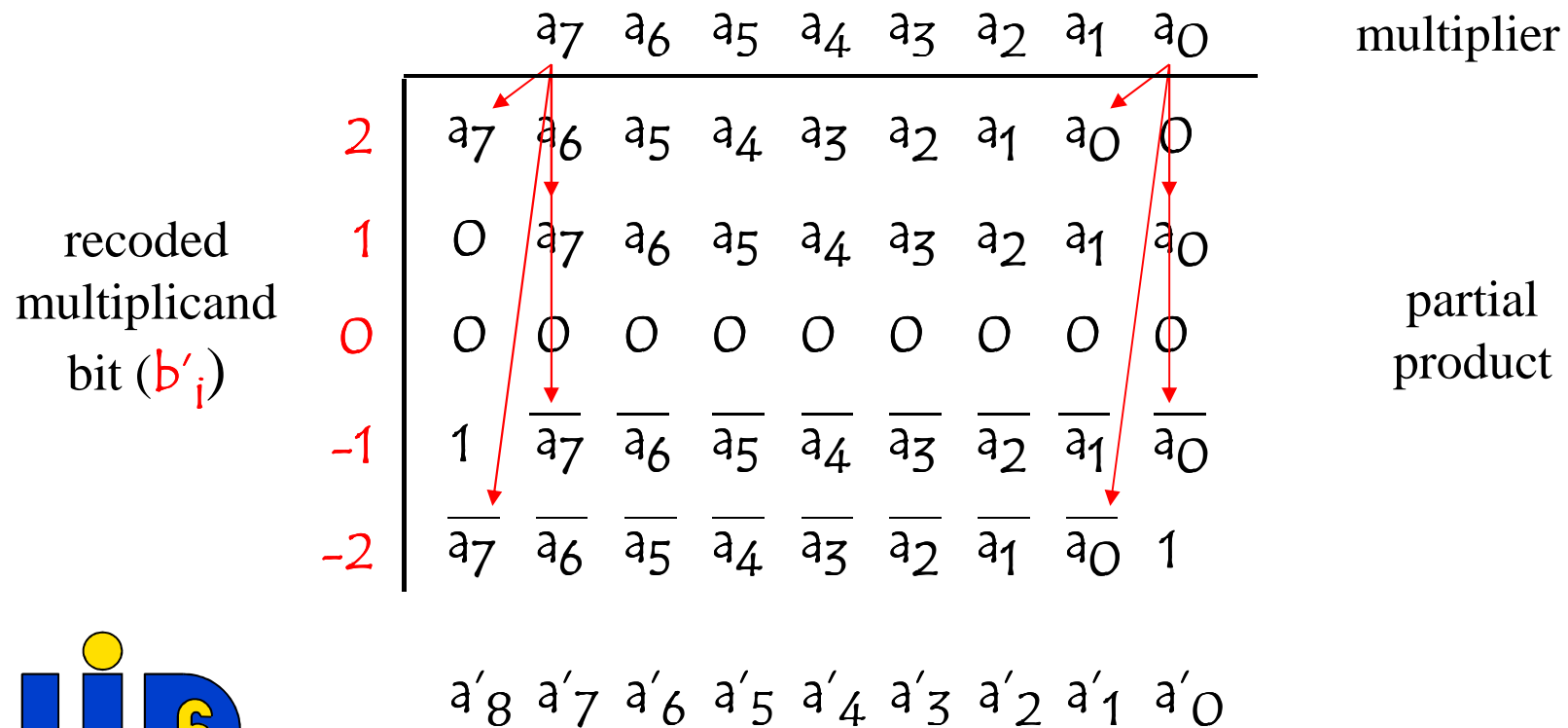


$$x = -2 \times 2^0 + 0 \times 2^2 + 1 \times 2^4 + 1 \times 2^6 + 0 \times 2^8 = 78$$

# Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

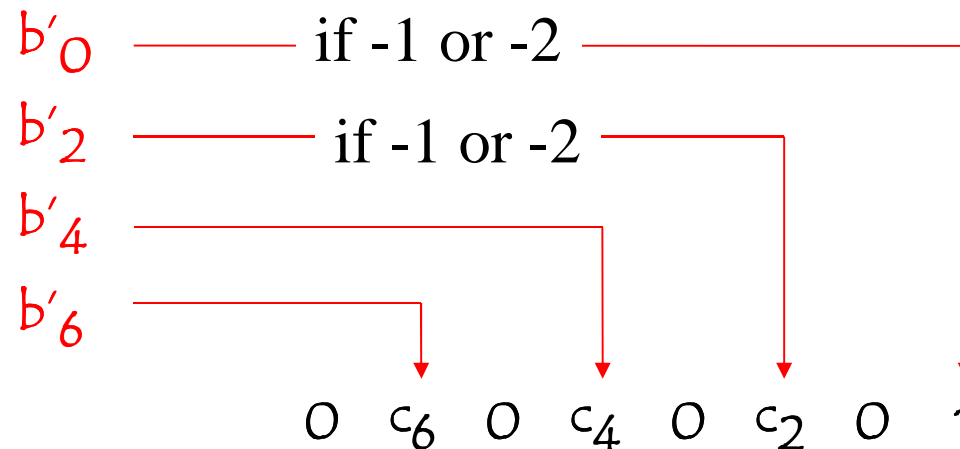


# Multipliers

Implementation : parallel multiplier

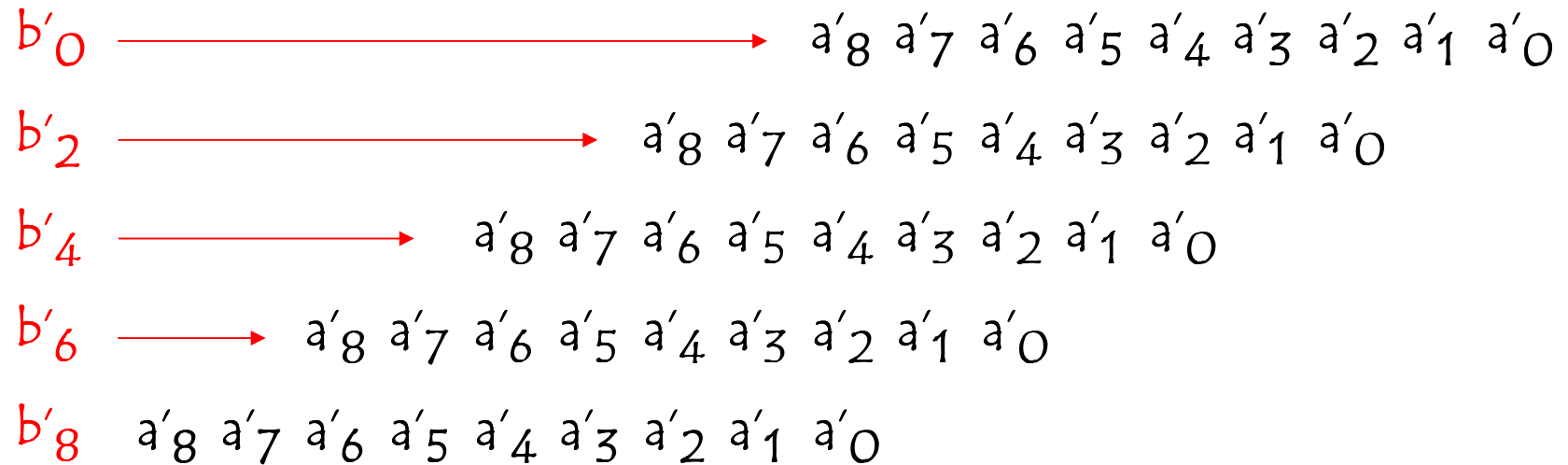
Improvement : Reduce the number of partial products

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



# Multipliers

## Implementation : parallel multiplier



$c_7 \quad c_6 \quad c_5 \quad c_4 \quad c_3 \quad c_2 \quad c_1 \quad c_0$

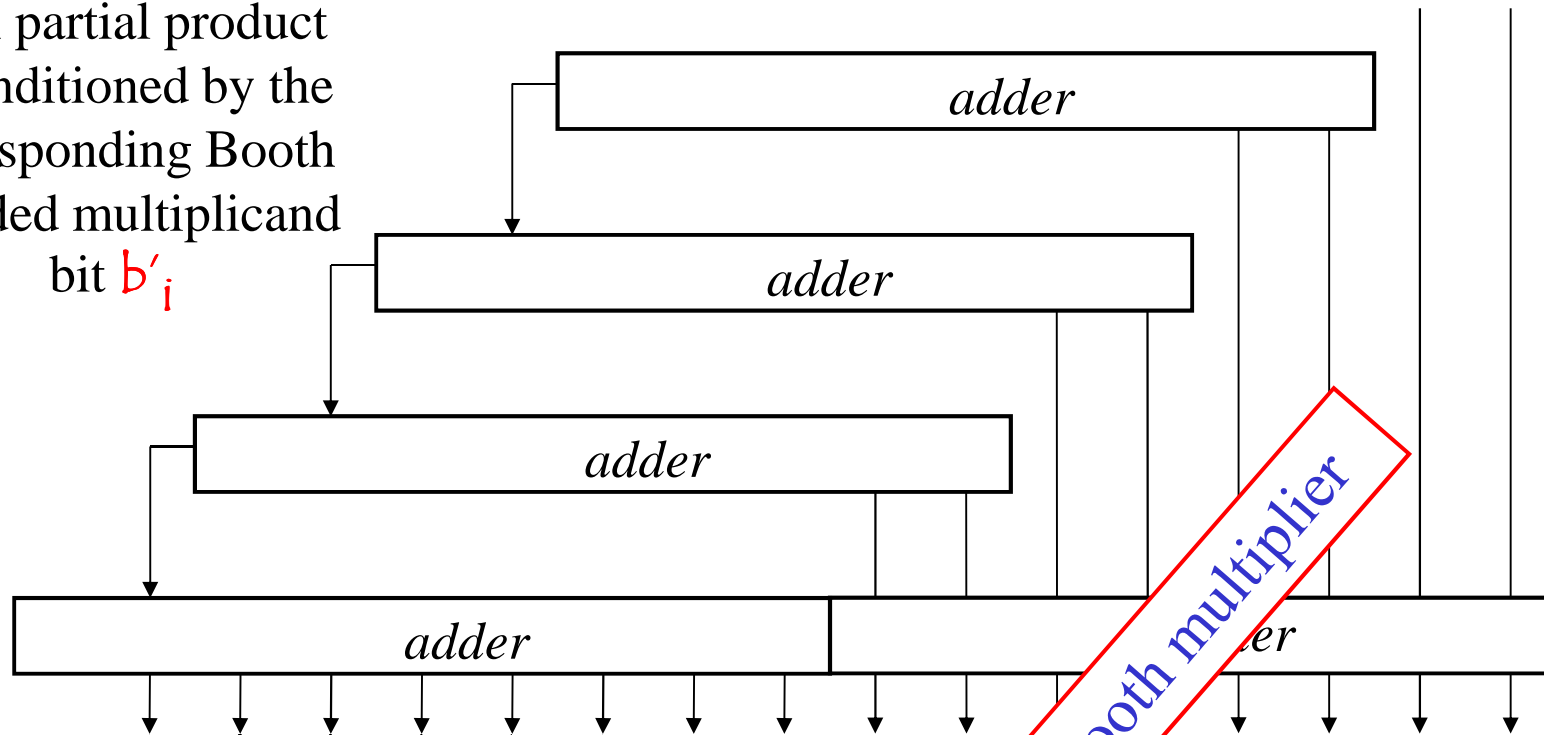
$s_{15} \quad s_{14} \quad s_{13} \quad s_{12} \quad s_{11} \quad s_{10} \quad s_9 \quad s_8 \quad s_7 \quad s_6 \quad s_5 \quad s_4 \quad s_3 \quad s_2 \quad s_1 \quad s_0$



# Multipliers

## Implementation : parallel multiplier

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



# Multipliers

Implementation : fast parallel multiplier

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



Redundant coding

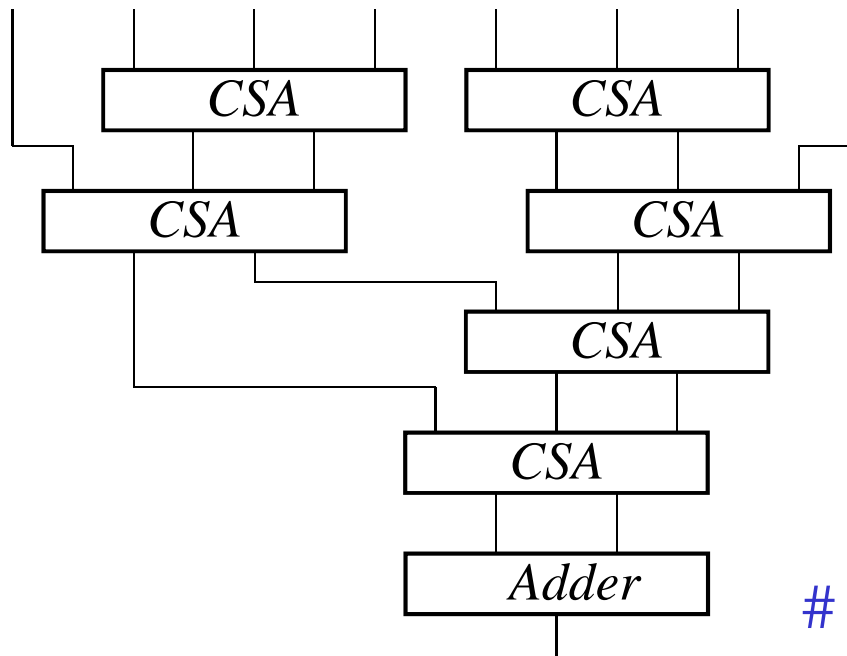
Use CSA (Carry Save Adder)  
to reduce 3 partial products  
into 2



# Multipliers

Implementation : fast parallel multiplier

8 partial products



Wallace multiplier

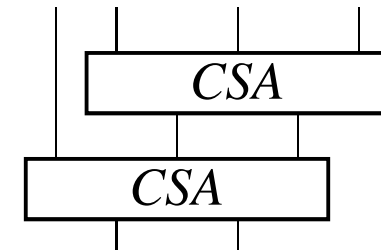
# of CSA layers  $\approx \log_{3/2} (n/2)$

# Multipliers

Implementation : fast parallel multiplier

32×32 bits multiplier  $32 \rightarrow 22 \rightarrow 15 \rightarrow 10 \rightarrow 7 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2$

Using  $4 \rightarrow 2$  reduction leads to a more regular hardware implementation



32×32 bits multiplier  $32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2$

# of CSA layers  $\approx 2 \log (n/2)$





# 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 extra 'partial products' can be used to extended the functionality of the multiplier



# Multipliers

Two relative numbers  $a$  and  $b$  coded on  $n$  bits

$$p = a \times b$$

How it works if  $b < 0$  ?

$$\begin{aligned} p = a \times b = (-a) \times (-b) &= (\bar{a} + 1) \times (\bar{b} + 1) \\ &= \bar{a} \times \bar{b} + \bar{a} + \bar{b} + 1 \end{aligned}$$



# Multipliers

Multiply and add

$$p = a \times b + c$$

$$\begin{aligned} p = a \times b + c &= (-a) \times (-b) + c = (\bar{a} + 1) \times (\bar{b} + 1) + c \\ &= \bar{a} \times \bar{b} + \bar{a} + \bar{b} + 1 + c \end{aligned}$$

