The Abdus Salam International Centre for Theoretical Physics

2499-18

International Training Workshop on FPGA Design for Scientific Instrumentation and Computing

11 - 22 November 2013

Digital CMOS Design
Multipliers

Pirouz Bazargan-Sabet
Department ASIM, LIP6
University Pierre & Marie Curie (VI), 4 place Jussieu, 75252 Paris Cedex 05
France

Outline

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

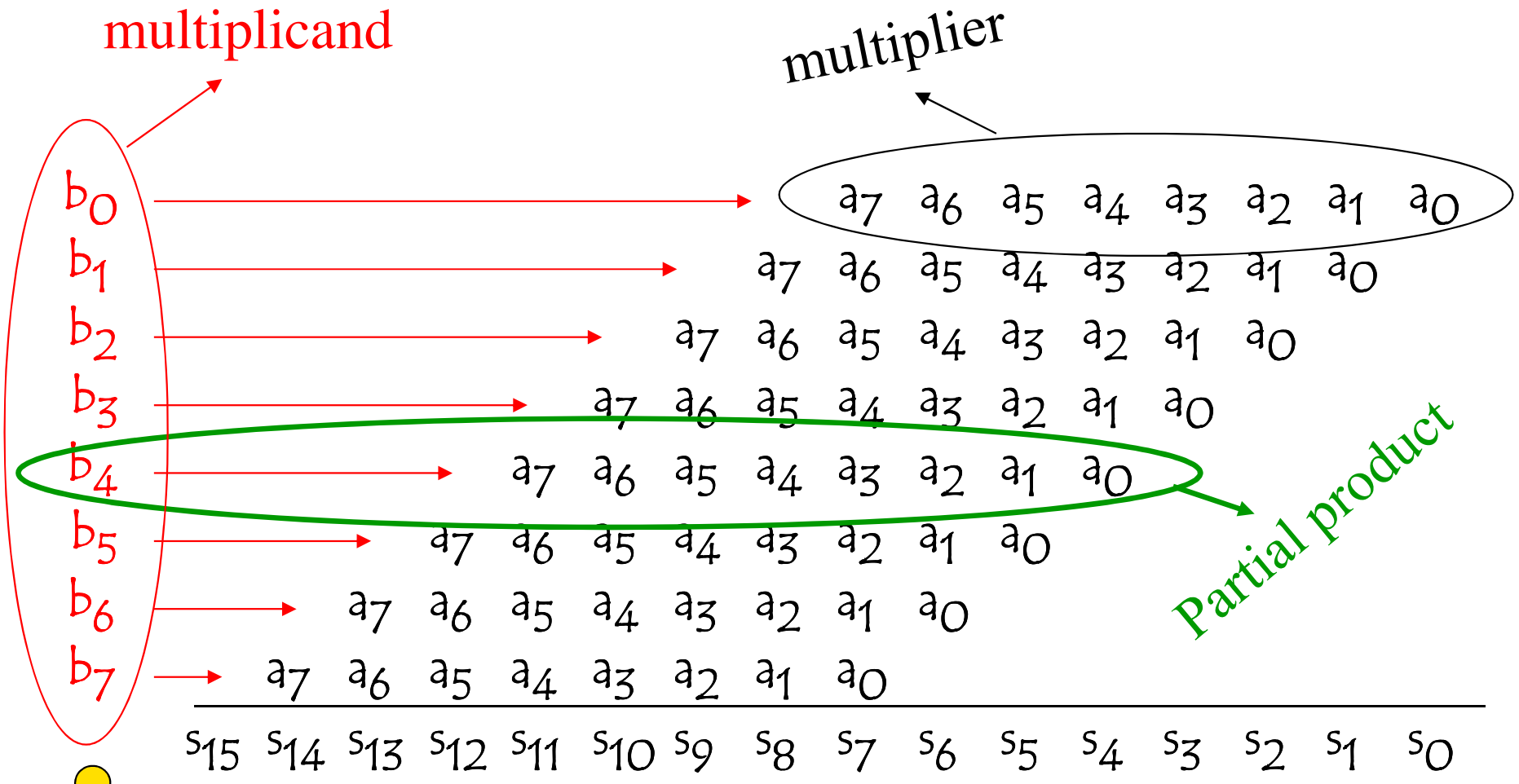
Multipliers

Two natural numbers a and b coded on n bits

the result of a × b is coded on 2n bits

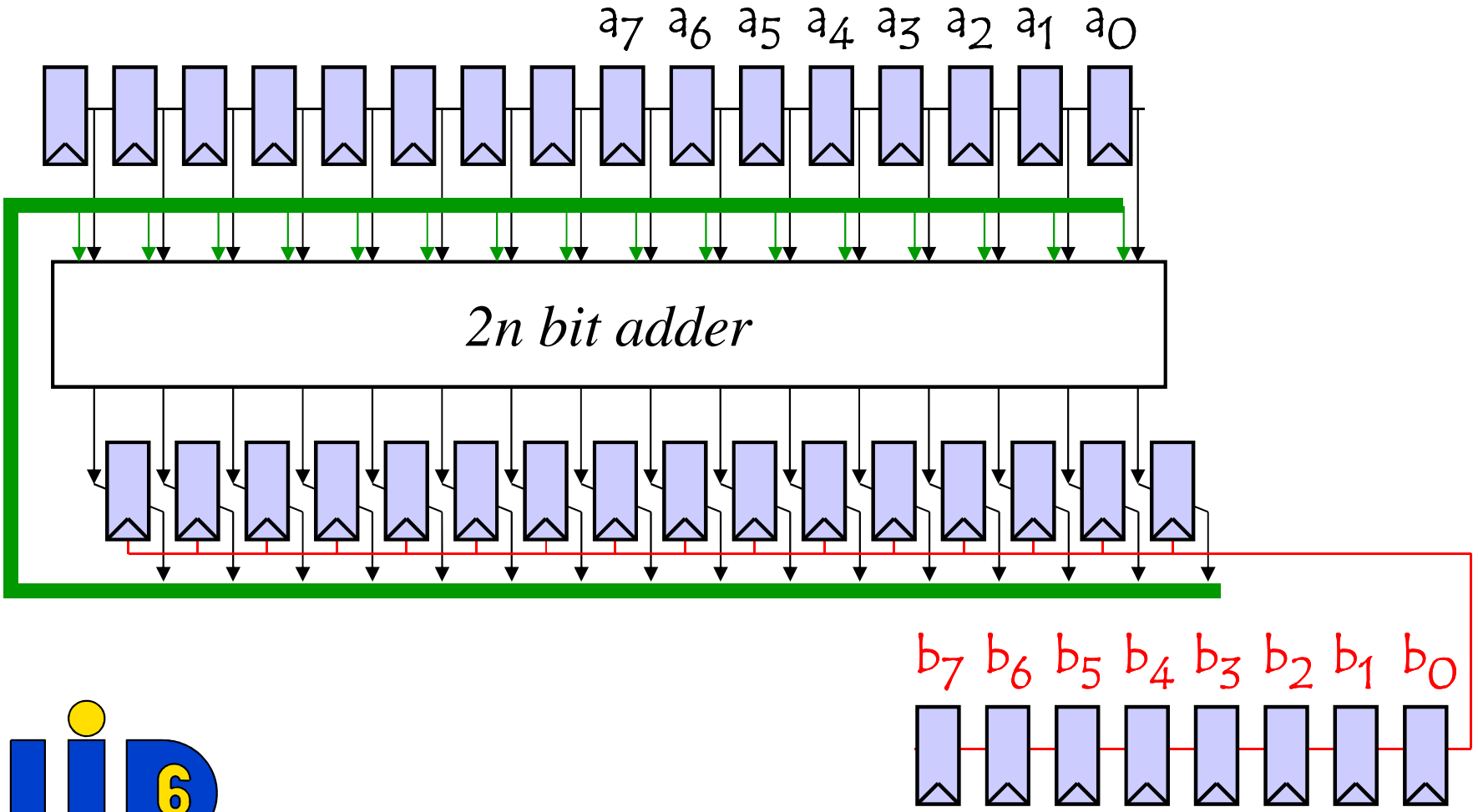
Classic scholar multiplication

Multipliers



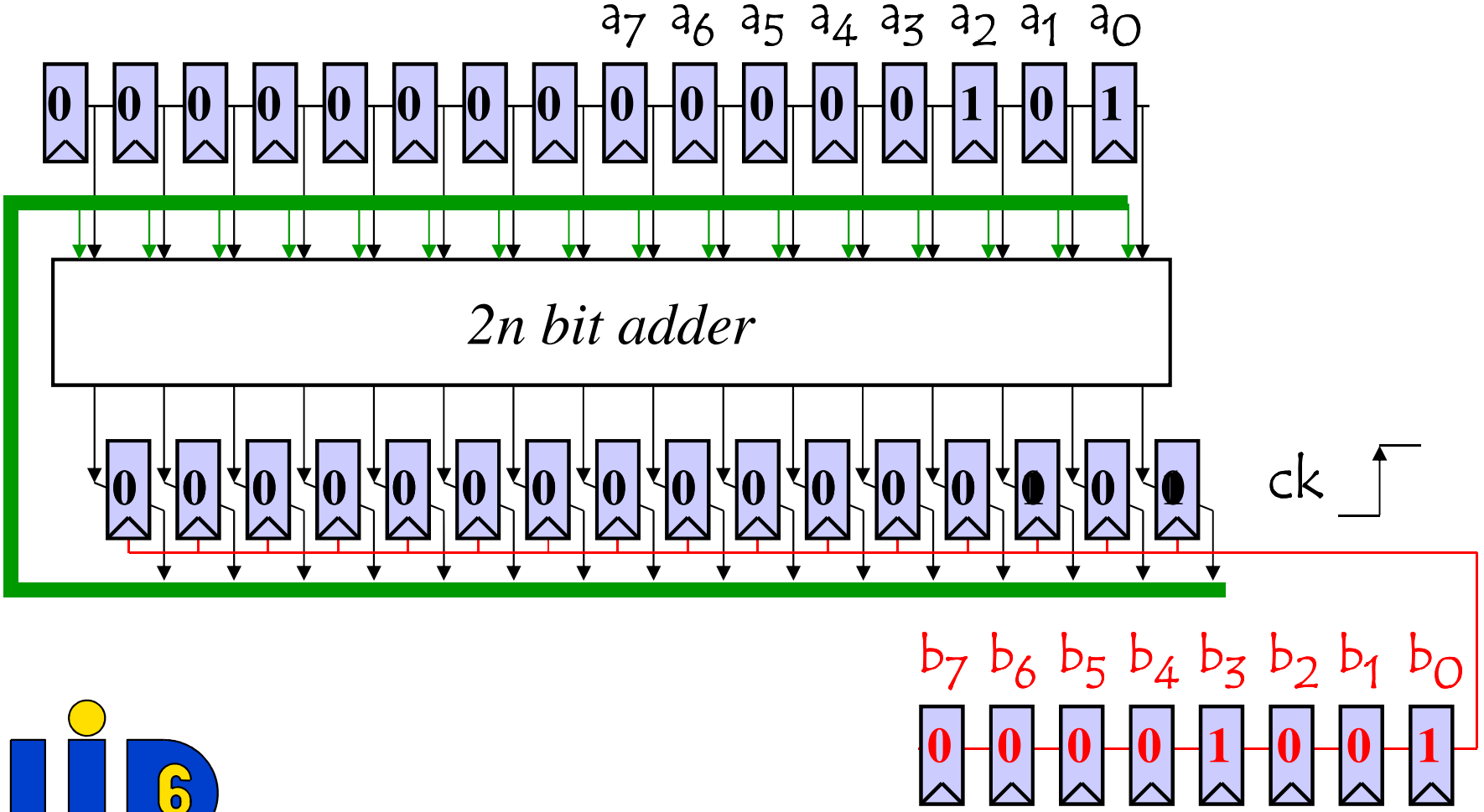
Multipliers

Implementation : sequential multiplier



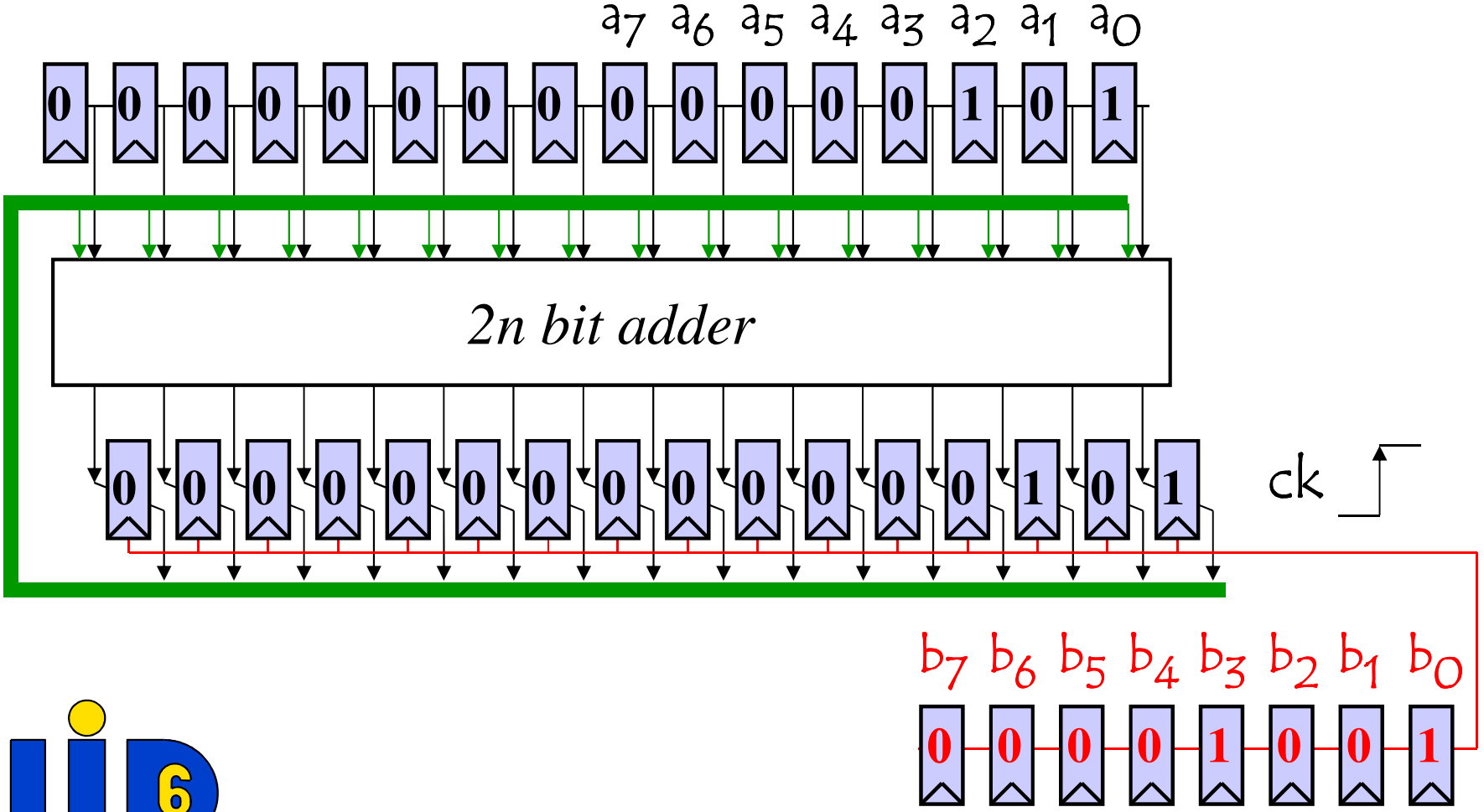
Multipliers

Implementation : sequential multiplier



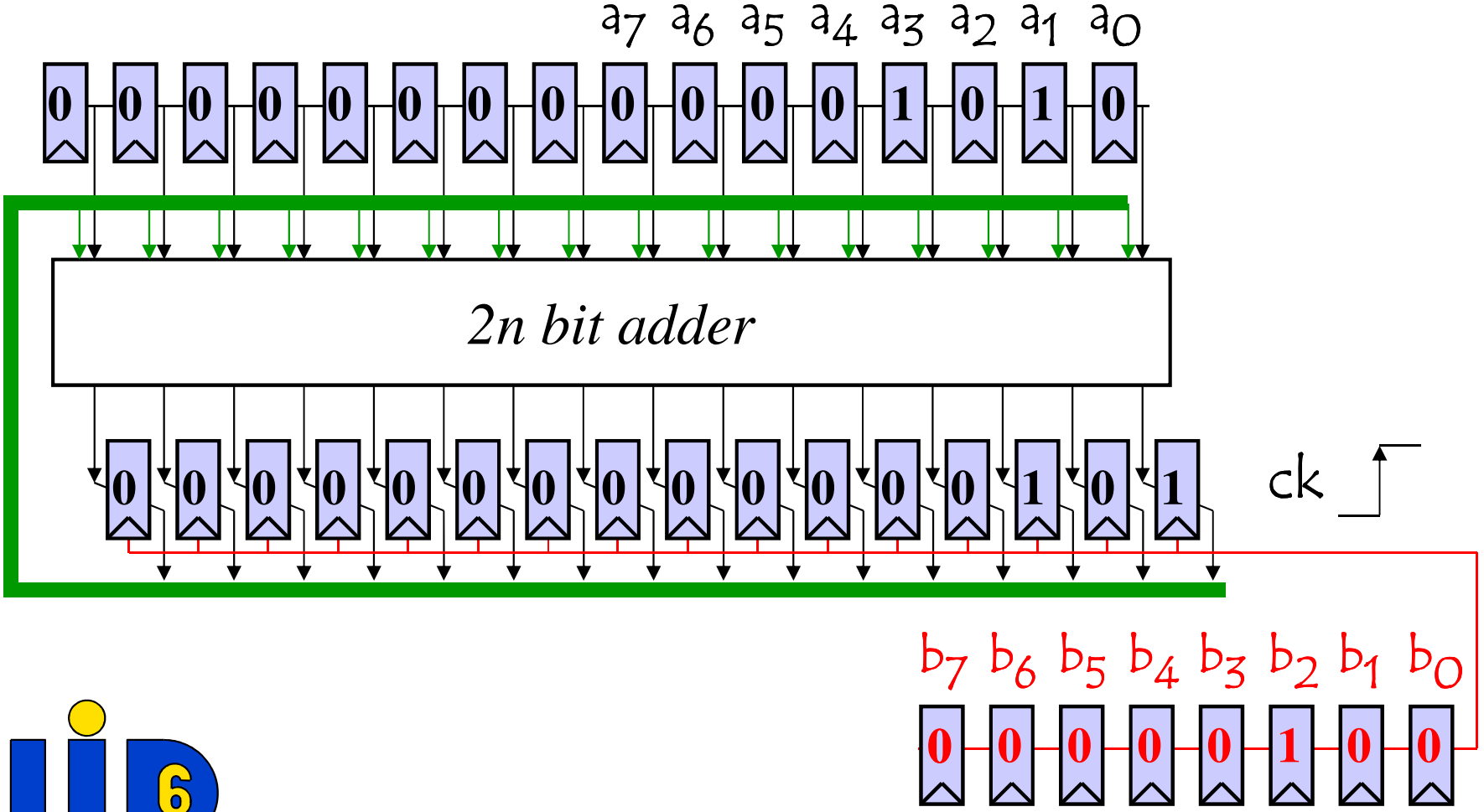
Multipliers

Implementation : sequential multiplier



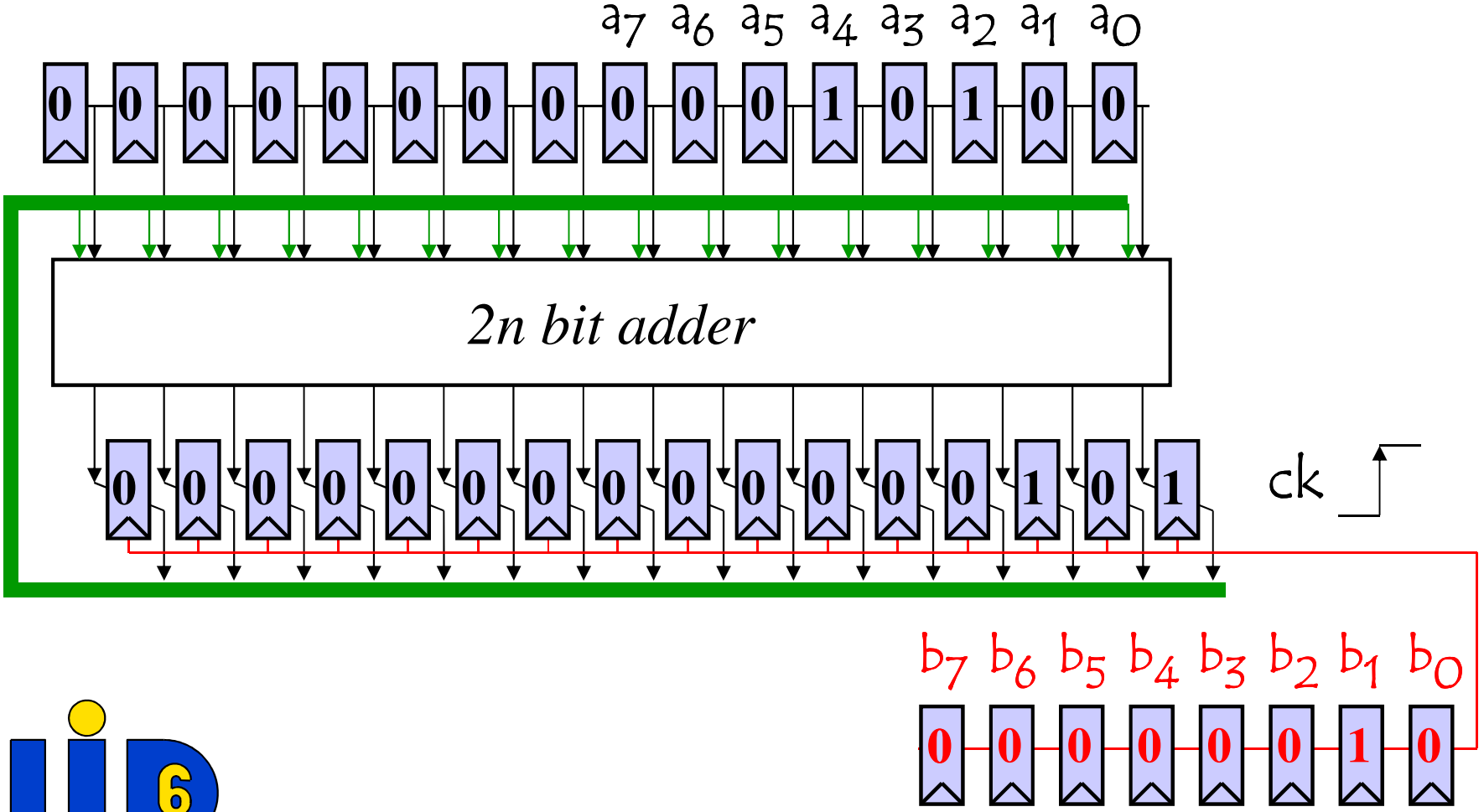
Multipliers

Implementation : sequential multiplier



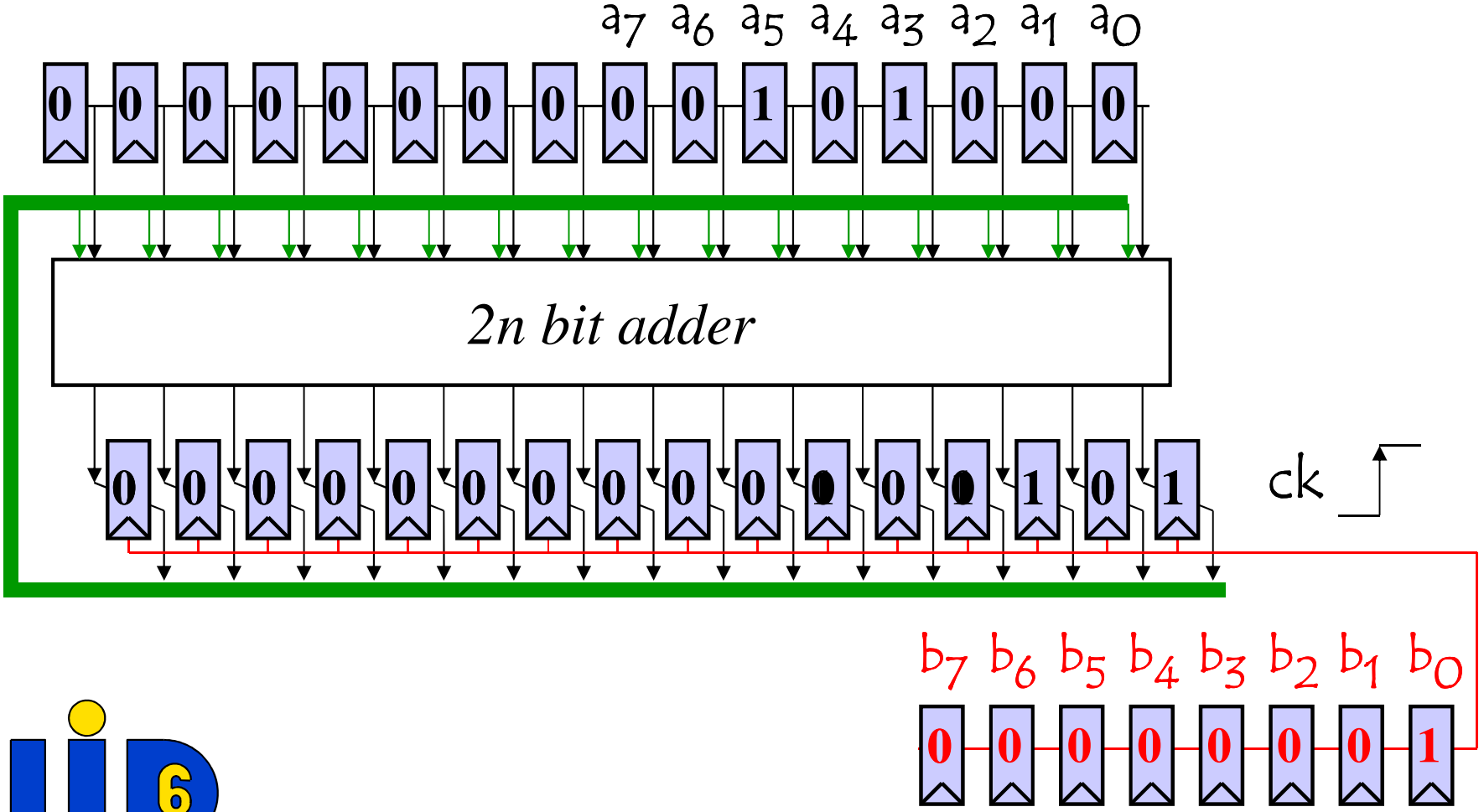
Multipliers

Implementation : sequential multiplier



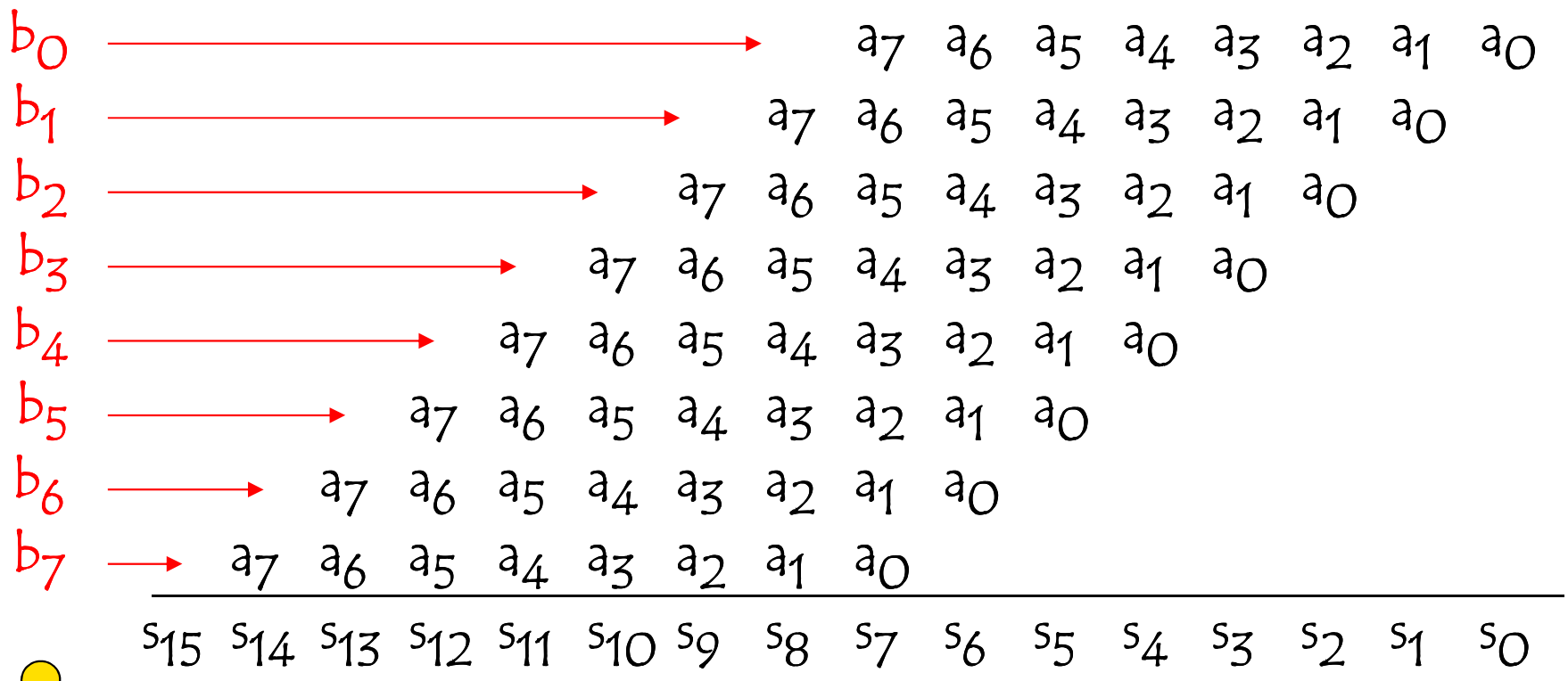
Multipliers

Implementation : sequential multiplier



Multipliers

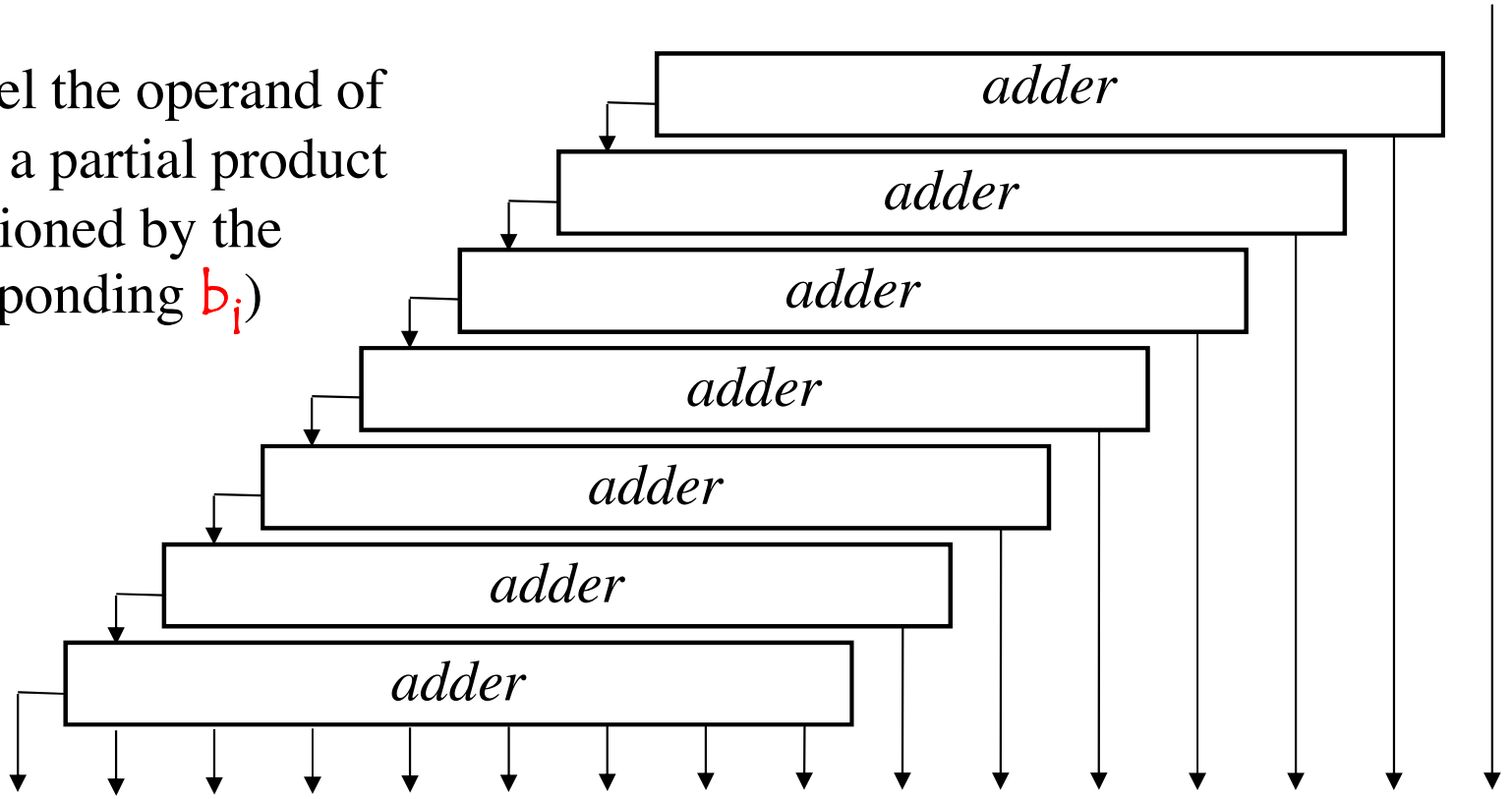
Implementation : parallel multiplier



Multipliers

Implementation : parallel multiplier

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



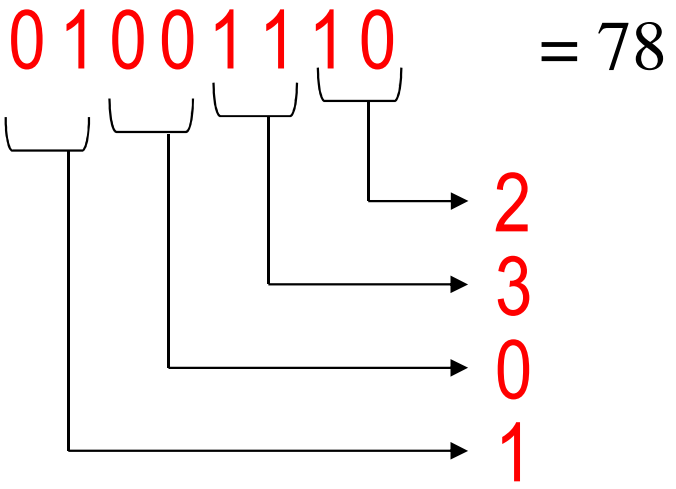
Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

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

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



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

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

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

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

$$2 = 4 - 2$$

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

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

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

$$x = (x_0 + 4x_1 - 2x_1) \quad + \\ (x_2 + 4x_3 - 2x_3) \times 4 + \\ (x_4 + 4x_5 - 2x_5) \times 16 + \cdots$$

$$x = (x_0 + 0 - 2x_1) \quad + \\ (x_2 + x_1 - 2x_3) \times 4 + \\ (x_4 + x_3 - 2x_5) \times 16 + \cdots$$

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

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

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

next weight

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

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

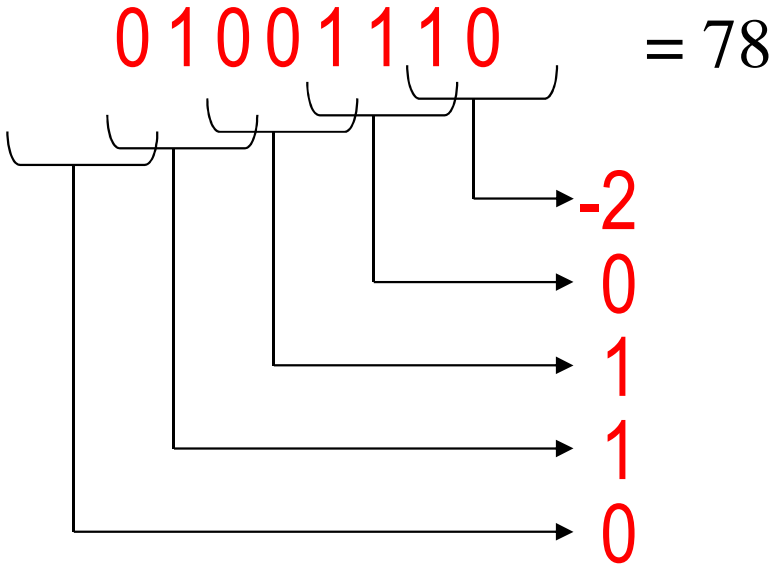
Multipliers

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

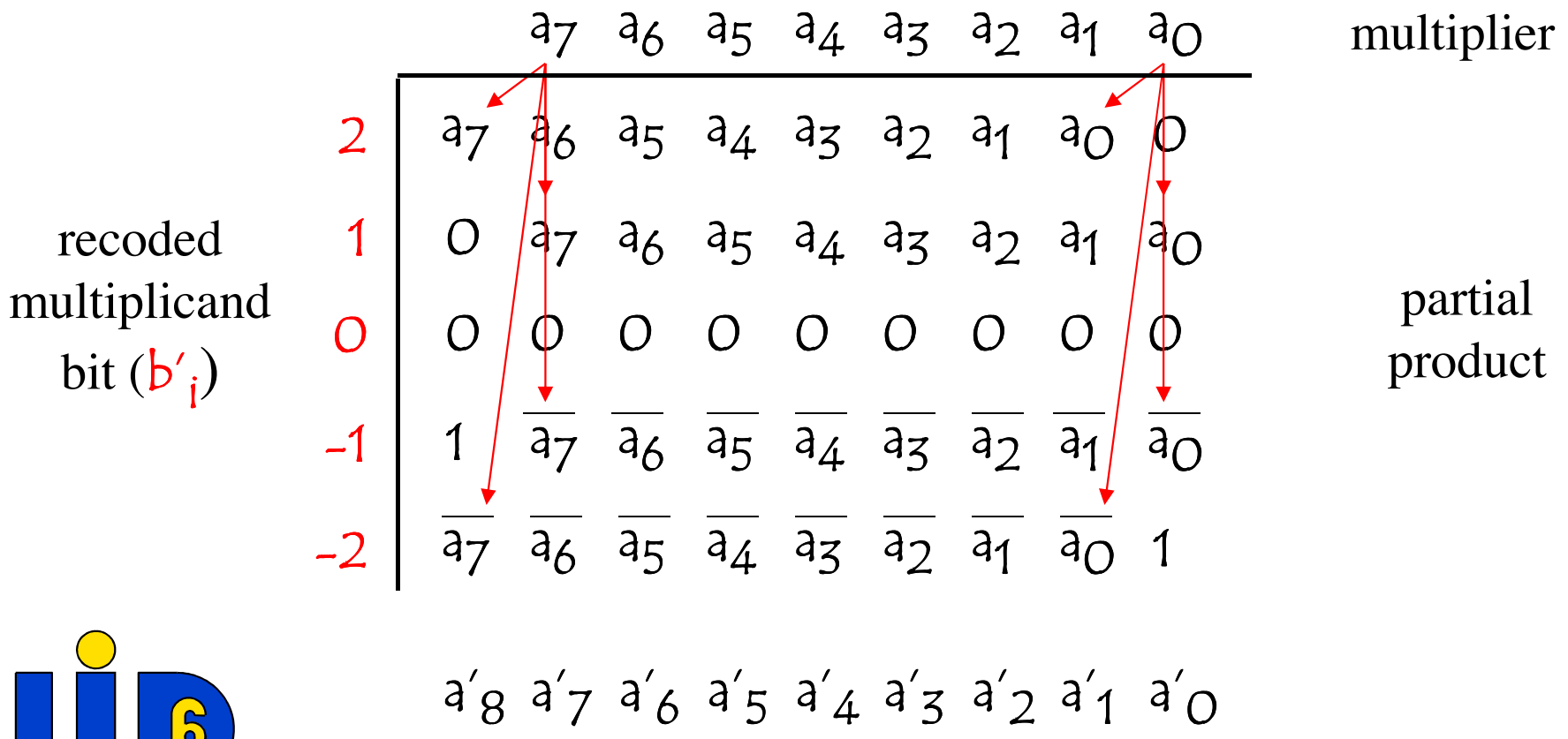


$$x = -2 \times 2^0 + 0 \times 2^2 + 1 \times 2^4 + 1 \times 2^6 + 0 \times 2^8 = 78$$

Multipliers

Implementation : parallel multiplier

Improvement : Reduce the number of partial products

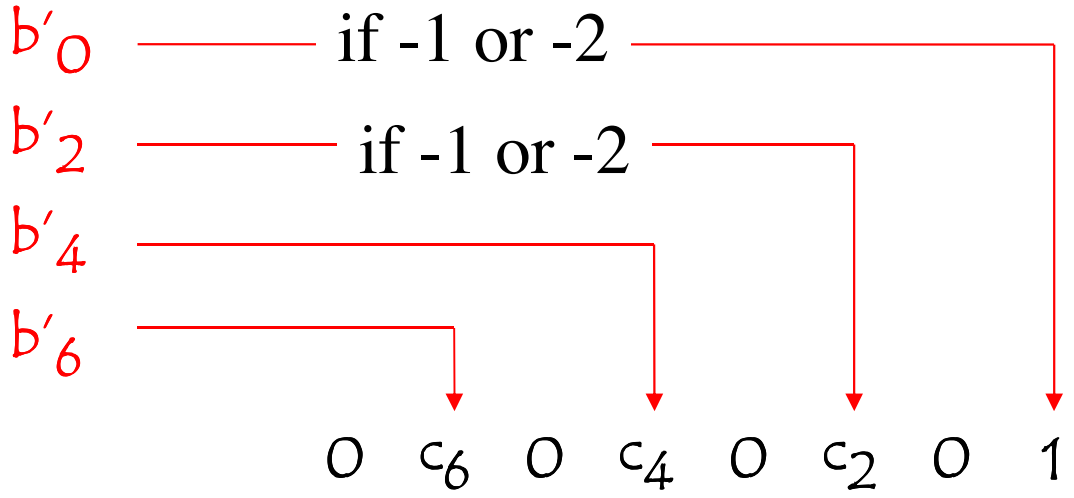


Multipliers

Implementation : parallel multiplier

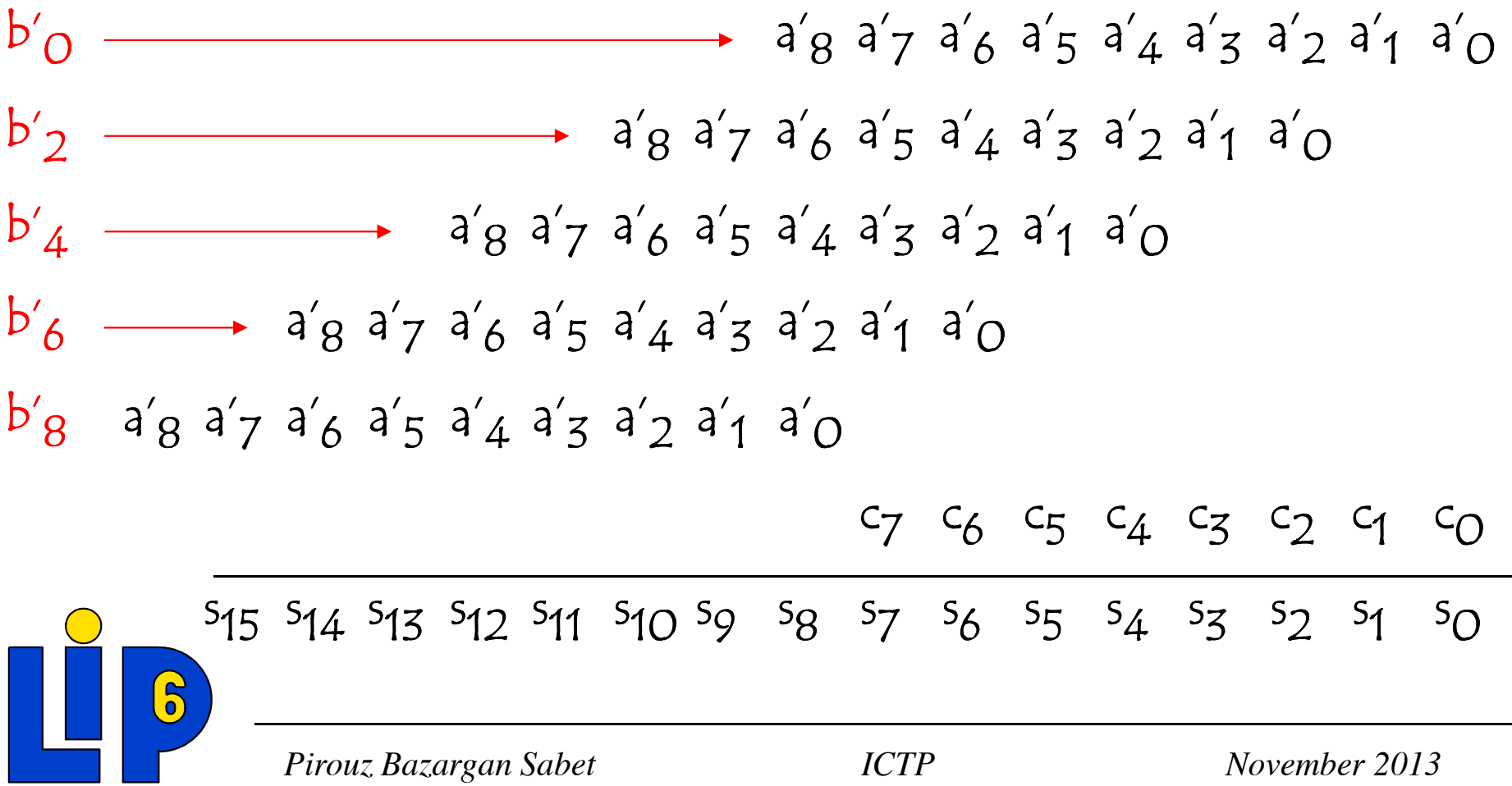
Improvement : Reduce the number of partial products

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



Multipliers

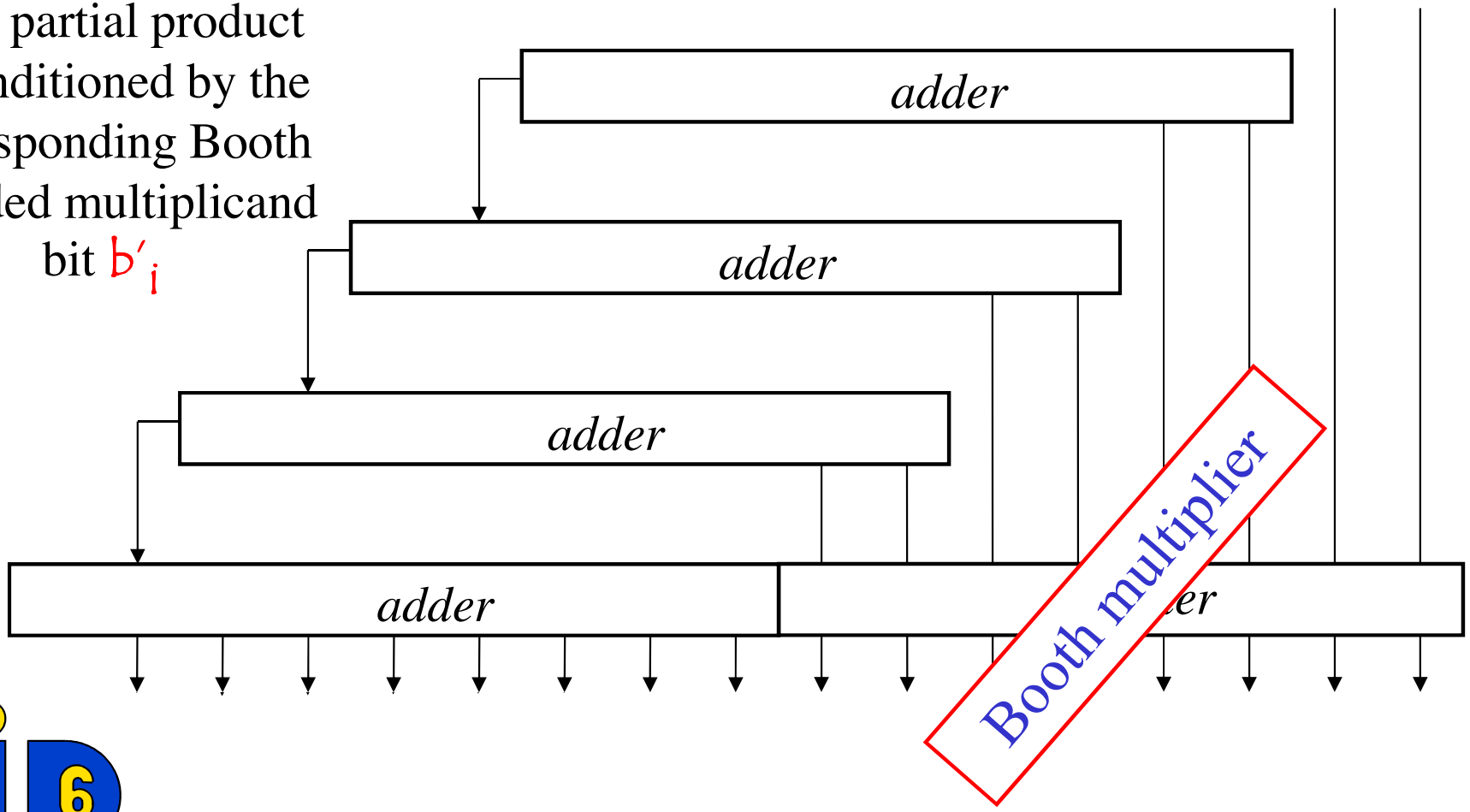
Implementation : parallel multiplier



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

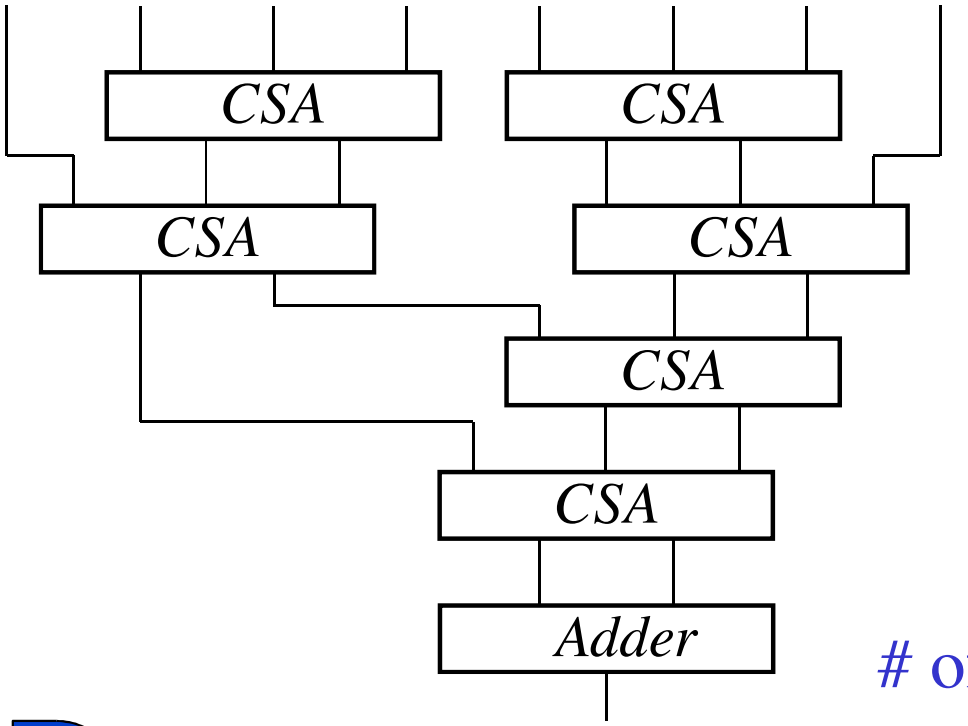
Redundant coding

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

Multipliers

Implementation : fast parallel multiplier

8 partial products



Wallace multiplier

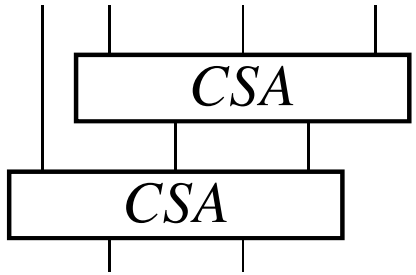
# of CSA layers ≈ $\log_{3/2}(n/2)$

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 ≈ 2 log (n/2)

Multipliers

Implementation : fast parallel multiplier

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

$2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 13 \rightarrow 19 \rightarrow 28 \rightarrow 42$

The additional ‘rooms’ may 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 ?

$$p = a \times b = (-a) \times (-b) = (\overline{a} + 1) \times (\overline{b} + 1)$$
$$= \overline{a} \times \overline{b} + \overline{a} + \overline{b} + 1$$

Multipliers

Multiply and add

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

and if b < 0

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

$$= \bar{a} \times \bar{b} + \bar{a} + \bar{b} + 1 + c$$