

Opérateurs Arithmétiques

Adders

Outline

- Digital CMOS design
- Arithmetic operators
 - Adders
 - Comparators
 - Shifters
 - Multipliers



Adders

Adding two natural numbers

At each stage, I need to sum 3 single bit numbers a_i, b_i, c_i
 The carry out of the stage i is the input carry of the next stage

$$\begin{array}{r} c_{i+1}c_i \\ a_i \\ + \\ b_i \\ \hline s_i \end{array}$$

s_i and c_{i+1} are Boolean functions of a_i, b_i, c_i



Adders

Adding two natural numbers

Let consider two natural numbers A and B
 coded on 8 bits using Natural Binary Code

$$\begin{array}{r} c_7 c_6 c_5 c_4 c_3 c_2 c_1 \\ a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 + \\ b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 \\ \hline s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0 \end{array}$$



Adders

Adding two natural numbers

	00	01	11	10
0	0	1	0	1
1	1	0	1	0

c_i s_i

$$s_i = a_i \oplus b_i \oplus c_i$$

$$\begin{aligned} s_i &= \bar{a}_i \cdot \bar{b}_i \cdot c_i + \bar{a}_i \cdot b_i \cdot \bar{c}_i + \\ &\quad a_i \cdot b_i \cdot c_i + a_i \cdot \bar{b}_i \cdot \bar{c}_i \\ s_i &= \bar{a}_i (\bar{b}_i \cdot c_i + b_i \cdot \bar{c}_i) + \\ &\quad a_i (b_i \cdot c_i + \bar{b}_i \cdot \bar{c}_i) \end{aligned}$$

$$s_i = \bar{a}_i (\bar{b}_i \oplus c_i) + a_i (b_i \oplus c_i)$$



Adders

Adding two natural numbers

	00	01	11	10
0	0	1	0	1
1	1	0	1	0

c_i

s_i

$$s_i = a_i \oplus b_i \oplus c_i$$

	00	01	11	10
0	0	0	1	0
1	0	1	1	1

c_i

c_{i+1}

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i$$



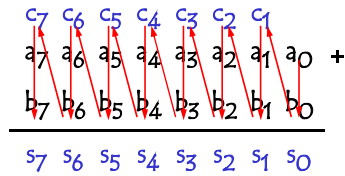
Pirouz Bazargan Sabet

Digital Design

February 2010

Adders

Adding two natural numbers



Pirouz Bazargan Sabet

Digital Design

February 2010

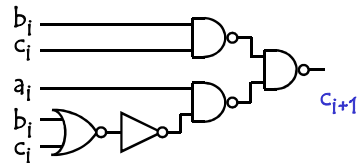
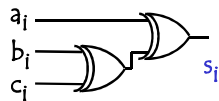
Adders

Adding two natural numbers

$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i$$

$$c_{i+1} = a_i \cdot (b_i + c_i) + b_i \cdot c_i$$



Pirouz Bazargan Sabet

Digital Design

February 2010

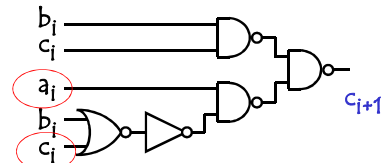
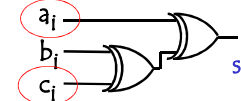
Adders

Adding two natural numbers

$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i$$

$$c_{i+1} = a_i \cdot (b_i + c_i) + b_i \cdot c_i$$



Addition delay depends on the delay of c_i to c_{i+1}

Pirouz Bazargan Sabet

Digital Design

February 2010

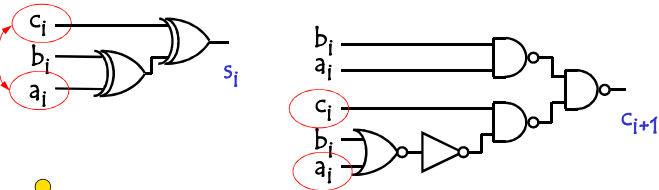
Adders

Adding two natural numbers

$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i$$

$$c_{i+1} = a_i \cdot b_i + (a_i + b_i) \cdot c_i$$



Pirouz Bazargan Sabet

Digital Design

February 2010

Adders

Adding two natural numbers

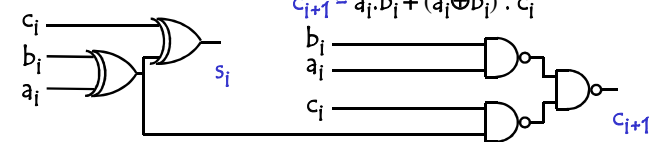
$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i$$

$$c_{i+1} = a_i \cdot b_i + (a_i + b_i) \cdot c_i$$

$$c_{i+1} = a_i \cdot b_i + (a_i \oplus b_i + a_i \cdot b_i) \cdot c_i$$

$$c_{i+1} = a_i \cdot b_i + (a_i \oplus b_i) \cdot c_i$$



Pirouz Bazargan Sabet

Digital Design

February 2010

Adders

Adding two natural numbers

	0	1
0	0	1
1	1	0

a_i

b_i

$a_i \oplus b_i$

	0	1
0	0	1
1	1	1

a_i

b_i

$a_i + b_i$

$$a_i + b_i = a_i \oplus b_i + a_i \cdot b_i$$



Pirouz Bazargan Sabet

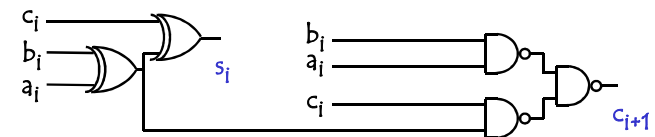
Digital Design

February 2010

Adders

Adding two natural numbers

The circuit generating s_i and c_{i+1} is called a Full Adder (FA)



Pirouz Bazargan Sabet

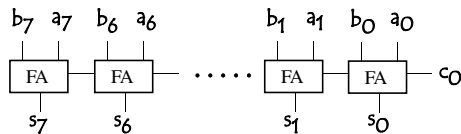
Digital Design

February 2010

Adders

Adding two natural numbers

At each stage, I need to sum 3 single bit numbers $a_i b_i c_i$
The carry out of the stage i is the input carry of the next stage



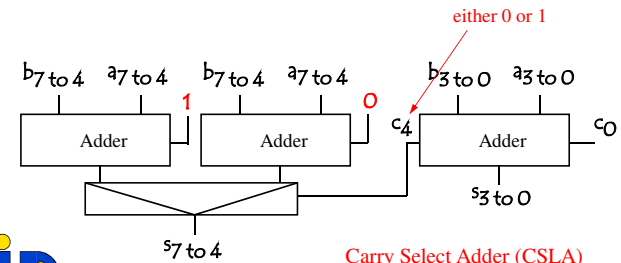
Ripple Carry Adder (RCA)



Adders

Adding two natural numbers

Acceleration technics



Carry Select Adder (CSLA)

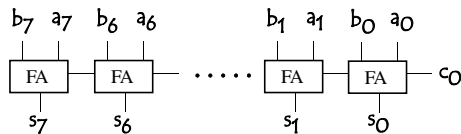


Adders

Adding two natural numbers

Ripple Carry Adder (RCA)

Area $\propto n$ Delay $\propto n$



Timing should be improved

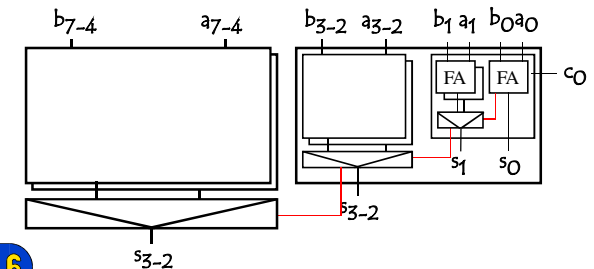


Adders

Adding two natural numbers

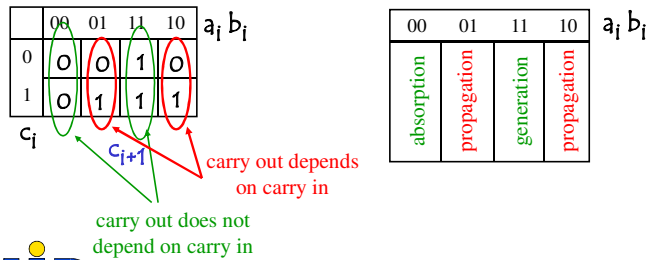
Carry Select Adder (CSLA)

Area $\propto 3^{\log(n)} = n^{\log(3)} = n^{1.585}$ Delay $\propto \log(n)$



Adders

Adding two natural numbers
Acceleration technics



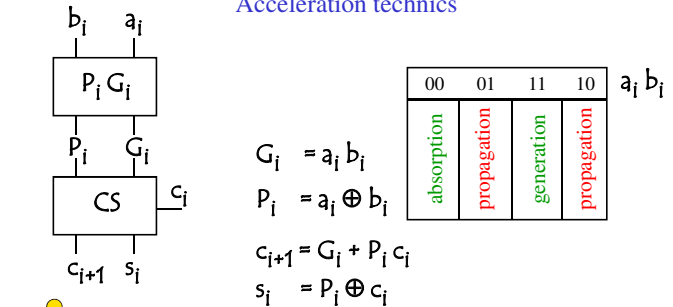
Pirouz Bazargan Sabet

Digital Design

February 2010

Adders

Adding two natural numbers
Acceleration technics



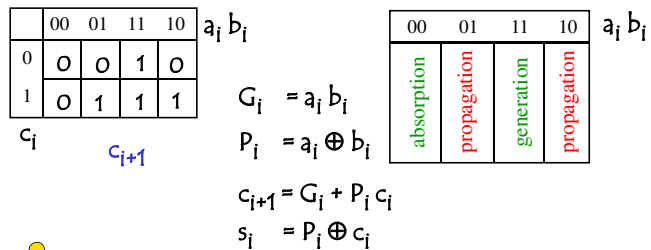
Pirouz Bazargan Sabet

Digital Design

February 2010

Adders

Adding two natural numbers
Acceleration technics



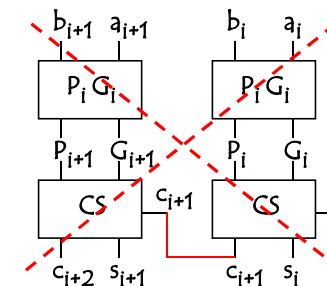
Pirouz Bazargan Sabet

Digital Design

February 2010

Adders

Adding two natural numbers
Acceleration technics



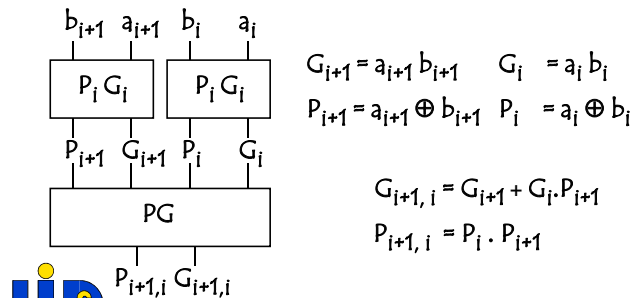
Pirouz Bazargan Sabet

Digital Design

February 2010

Adders

Adding two natural numbers
Acceleration technics

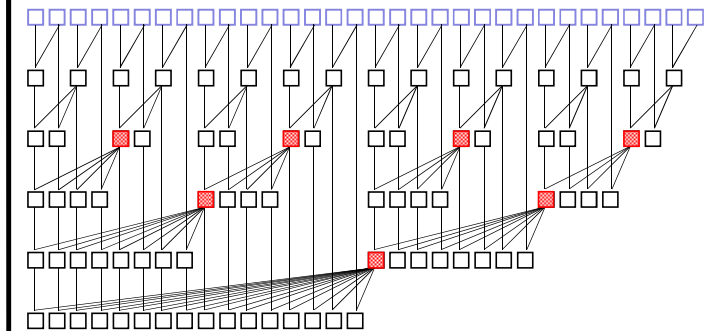


Pirouz Bazargan Sabet

Digital Design

February 2010

Adders



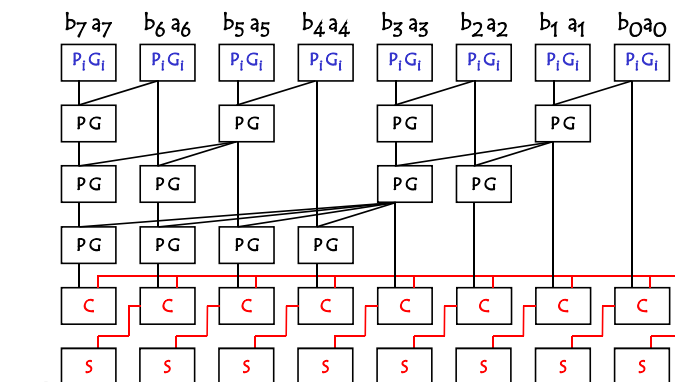
Pirouz Bazargan Sabet

Digital Design

February 2010

Slansky Adder

Adders

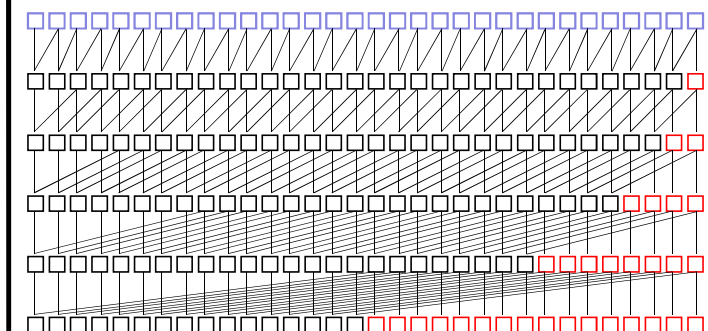


Pirouz Bazargan Sabet

Digital Design

February 2010

Adders



Pirouz Bazargan Sabet

Digital Design

February 2010

Kogge-Stone Adder

Adders

Adding two natural numbers (summary)

	Area	Delay
Ripple Carry (RCA)	n	n
Carry Select (CSLA)	$3 \log(n)$	$\log(n)$
Carry Lookahead (CLA)	$n \log(n)$	$\log(n)$



Opérateurs Arithmétiques

Comparators

Outline

□ Digital CMOS design

□ Arithmetic operators

- Adders
- **Comparators**
- Shifters
- Multipliers



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing a natural number to zero : =

Boolean function

Null = 1 if

$$\bar{a}_7 \cdot \bar{a}_6 \cdot \bar{a}_5 \cdot \bar{a}_4 \cdot \bar{a}_3 \cdot \bar{a}_2 \cdot \bar{a}_1 \cdot \bar{a}_0 = 1$$

$$\text{Null} = a_7 + a_6 + a_5 + a_4 + a_3 + a_2 + a_1 + a_0$$



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing a natural number to a constant : =

Let consider a natural number **A** coded on 8 bits using Natural Binary Code

$$\begin{array}{cccccccc} a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & = ? & & & \\ & & & & \downarrow & & & \\ & & & & 0/1 & & & \end{array}$$



Pirouz Bazargan Sabet

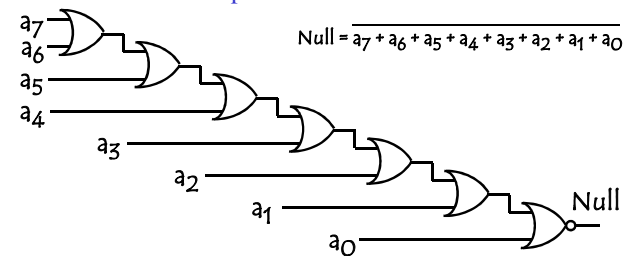
Digital Design

February 2010

Comparators

Comparing a natural number to zero : =

Implementation



Pirouz Bazargan Sabet

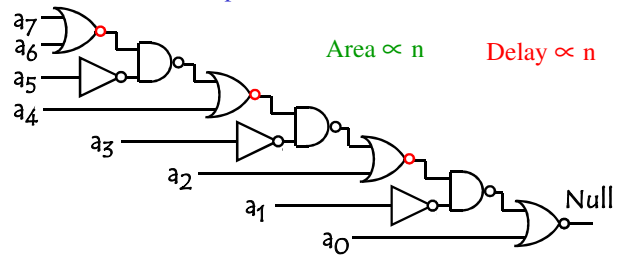
Digital Design

February 2010

Comparators

Comparing a natural number to zero : =

Implementation



Pirouz Bazargan Sabet

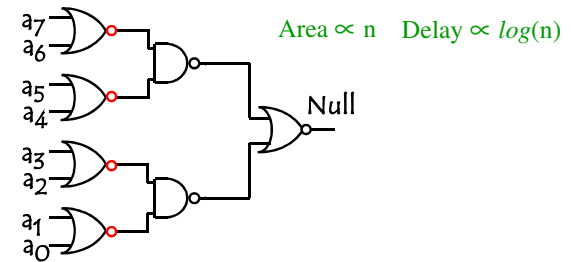
Digital Design

February 2010

Comparators

Comparing a natural number to zero : =

Implementation improvement



Pirouz Bazargan Sabet

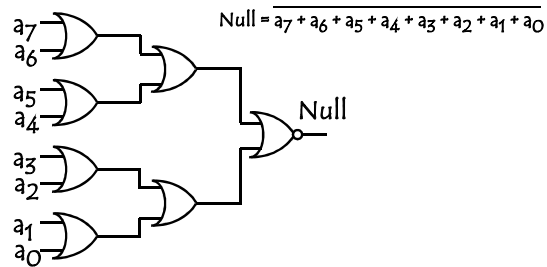
Digital Design

February 2010

Comparators

Comparing a natural number to zero : =

Implementation improvement



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing two natural numbers : =

Let consider two natural numbers **A** and **B** coded on 8 bits using Natural Binary Code

$$\begin{array}{cccccccc} a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ = ? \\ b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ \downarrow \\ 0/1 \end{array}$$



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing two natural numbers : =

Boolean function

A Equal B if : $a_7=b_7$ and $a_6=b_6$ and ... and $a_0=b_0$

A Equal B if : $\overline{(a_7 \oplus b_7)} \cdot \dots \cdot \overline{(a_0 \oplus b_0)} = 1$

Equal = $\overline{(a_7 \oplus b_7)} + \dots + \overline{(a_0 \oplus b_0)}$

Equal = $(e_7) + \dots + (e_0)$



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing two natural numbers : <

Let consider two natural numbers A and B coded on 8 bits using Natural Binary Code

$a_7 \ a_6 \ a_5 \ a_4 \ a_3 \ a_2 \ a_1 \ a_0$
 $< ?$
 $b_7 \ b_6 \ b_5 \ b_4 \ b_3 \ b_2 \ b_1 \ b_0$
 \downarrow
 $0 / 1$



Pirouz Bazargan Sabet

Digital Design

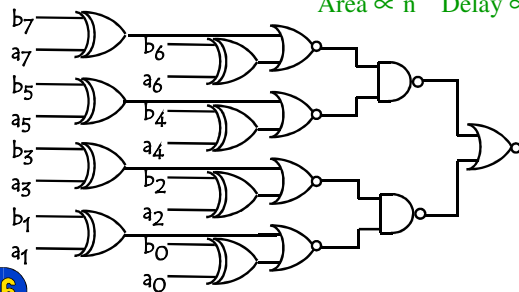
February 2010

Comparators

Comparing two natural numbers : =

Implementation

Area $\propto n$ Delay $\propto \log(n)$



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing two natural numbers : <

Boolean function

$A < B$ if : $a_7 < b_7$ or $(a_7 = b_7 \text{ and } (a_6 < b_6 \text{ or } (a_6 = b_6 \text{ and } \dots)))$

$a_7 \ a_6 \ a_5 \ a_4 \ a_3 \ a_2 \ a_1 \ a_0$
 $< ?$
 $b_7 \ b_6 \ b_5 \ b_4 \ b_3 \ b_2 \ b_1 \ b_0$
 \downarrow
 $0 / 1$



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing two natural numbers : <

Boolean function

$A < B$ if : $a_7 < b_7$ or $(a_7 = b_7 \text{ and } (a_6 < b_6 \text{ or } (a_6 = b_6 \text{ and } \dots)))$

$A < B$ if : $\bar{a}_7 b_7 + ((a_7 \oplus b_7) \cdot (\bar{a}_6 b_6 + ((a_6 \oplus b_6) \cdot \dots)))$



Pirouz Bazargan Sabet

Digital Design

February 2010

Comparators

Comparing two natural numbers : <

Implementation Improvement

$A < B$ if : $a_7 < b_7$ or $(a_7 = b_7 \text{ and } (a_6 < b_6 \text{ or } (a_6 = b_6 \text{ and } \dots)))$

$A < B$ if : $\bar{a}_7 b_7 + ((a_7 \oplus b_7) \cdot (\bar{a}_6 b_6 + ((a_6 \oplus b_6) \cdot \dots)))$

$\bar{a}_i b_i + (\bar{a}_i \oplus b_i) \cdot I_i$

Propagation
Generation



Pirouz Bazargan Sabet

Digital Design

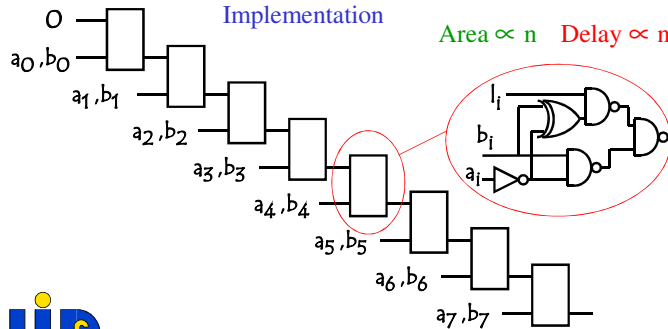
February 2010

Comparators

Comparing two natural numbers : <

Implementation

Area $\propto n$ Delay $\propto n$



Pirouz Bazargan Sabet

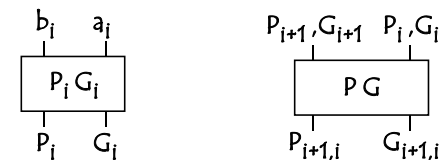
Digital Design

February 2010

Comparators

Comparing two natural numbers : <

Implementation Improvement



$$G_i = \bar{a}_i b_i$$

$$P_i = \bar{a}_i \oplus b_i$$

$$G_{i+1,i} = G_{i+1} + G_i \cdot P_{i+1}$$

$$P_{i+1,i} = P_i \cdot P_{i+1}$$

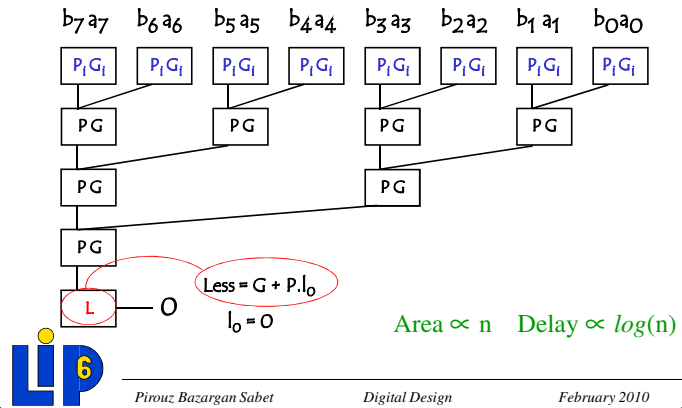


Pirouz Bazargan Sabet

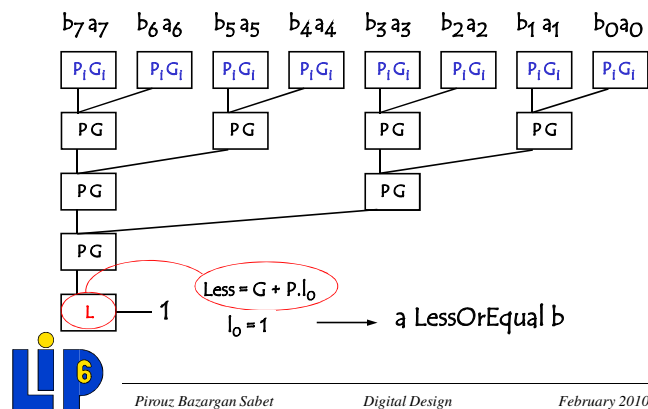
Digital Design

February 2010

Comparators



Comparators



Opérateurs Arithmétiques

Shifters

Outline

□ Digital CMOS design

□ Arithmetic operators

○ Adders

○ Comparators

○ Shifters

○ Multipliers



Pirouz Bazargan Sabet

Digital Design

February 2010

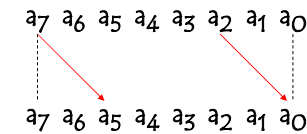
Shifters

Shifting a value

Let consider a value A coded on 8 bits

A can be shifted to the right by n positions ($0 \leq n < 8$)

logic



For a natural number, shift right is a division by 2^n



Pirouz Bazargan Sabet

Digital Design

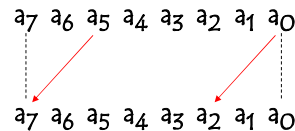
February 2010

Shifters

Shifting a value

Let consider a value A coded on 8 bits

A can be shifted to the left by n positions ($0 \leq n < 8$)



For a natural number, shift left is a multiplication by 2^n



Pirouz Bazargan Sabet

Digital Design

February 2010

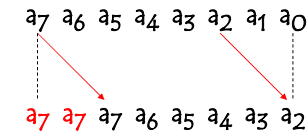
Shifters

Shifting a value

Let consider a value A coded on 8 bits

A can be shifted to the right by n positions ($0 \leq n < 8$)

arithmetic



For a relative number, shift right is a division by 2^n



Pirouz Bazargan Sabet

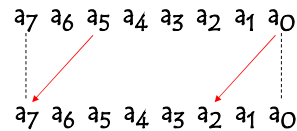
Digital Design

February 2010

Shifters

Shifting a value

Let consider a value A coded on 8 bits
 A can be rotated to the left by n positions ($0 \leq n < 8$)



Pirouz Bazargan Sabet

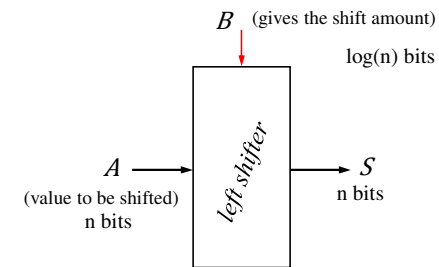
Digital Design

February 2010

Shifters

Shifting a value

Implementation (left shifter)



Pirouz Bazargan Sabet

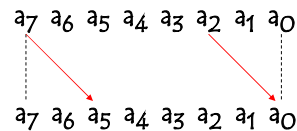
Digital Design

February 2010

Shifters

Shifting a value

Let consider a value A coded on 8 bits
 A can be rotated to the right by n positions ($0 \leq n < 8$)



Pirouz Bazargan Sabet

Digital Design

February 2010

Shifters

Shifting a value by 1 position to the left

Boolean function

B coded on 1 bit

If $b_0 = 0$
 $s_i = a_i$
 else
 $s_i = a_{i-1}$ assuming $a_{-1} = 0$

$$s_i = b_0 \cdot a_{i-1} + \overline{b_0} \cdot a_i \quad \Rightarrow \quad 2\text{-input Multiplexer}$$



Pirouz Bazargan Sabet

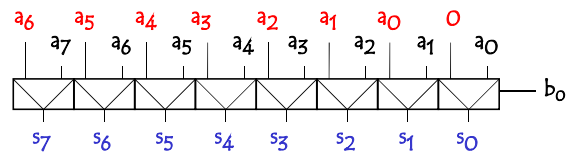
Digital Design

February 2010

Shifters

Shifting a value by 1 position to the left

Implementation



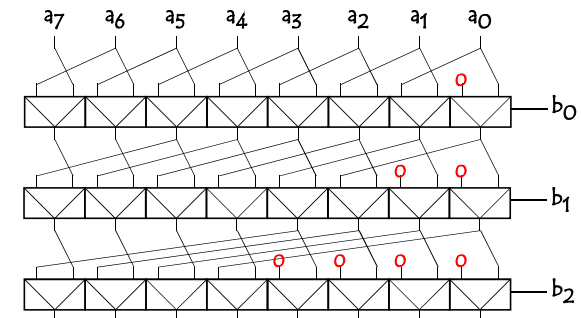
Pirouz Bazargan Sabet

Digital Design

February 2010

Shifters

Shifting a value to the left



Pirouz Bazargan Sabet

Area $\propto n \log(n)$ Delay $\propto \log(n)$

Shifters

Shifting a value by B positions to the left

B may be

- 0 = 000
- 1 = 001
- 2 = 010
- 3 = 011
- 4 = 100
- 5 = 101
- 6 = 110
- 7 = 111

A combination of $2^2, 2^1, 2^0$



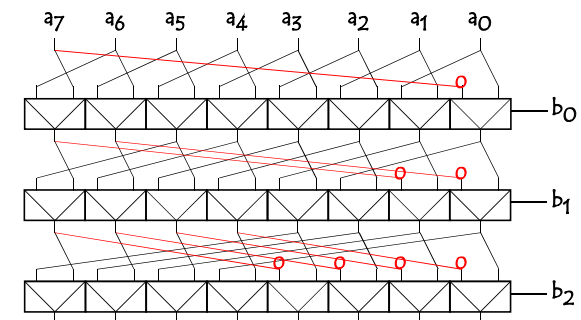
Pirouz Bazargan Sabet

Digital Design

February 2010

Shifters

Rotate a value to the left

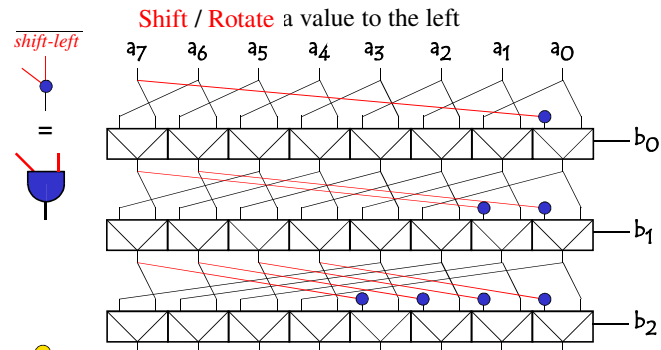


Pirouz Bazargan Sabet

Digital Design

February 2010

Shifters



Pirouz Bazargan Sabet

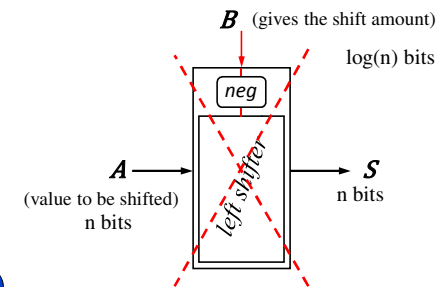
Digital Design

February 2010

Shifters

Shifting a value

Implementation (right shifter)



Pirouz Bazargan Sabet

Digital Design

February 2010

Shifters

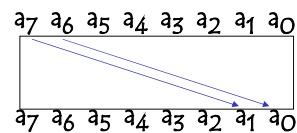
Shifting a value

Rotate left by 2

$6 = -2$

=
Rotate right by 6

$110 = \overline{010} + 1$



Pirouz Bazargan Sabet

Digital Design

February 2010

Shifters

Shifting a value

Rotate right by B

=
Rotate left by $-B$

=
Rotate left by $(\overline{B} + 1)$

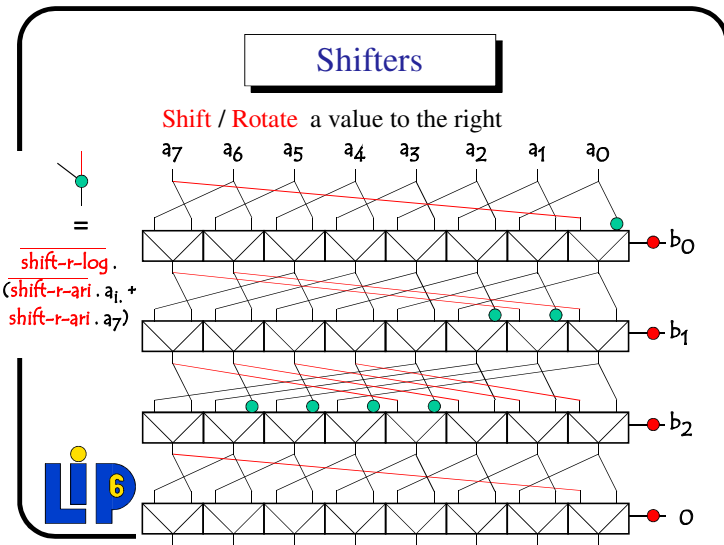
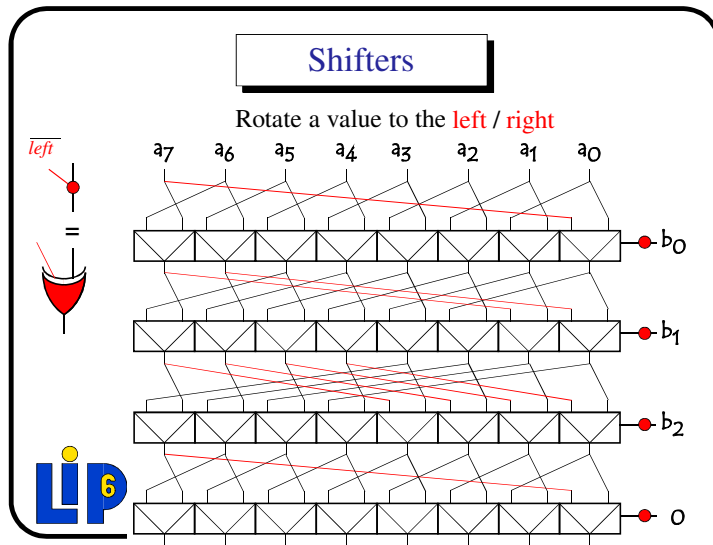
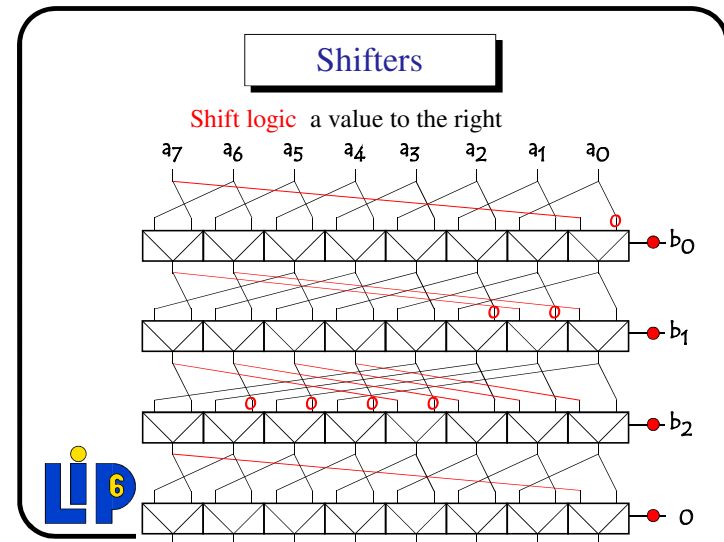
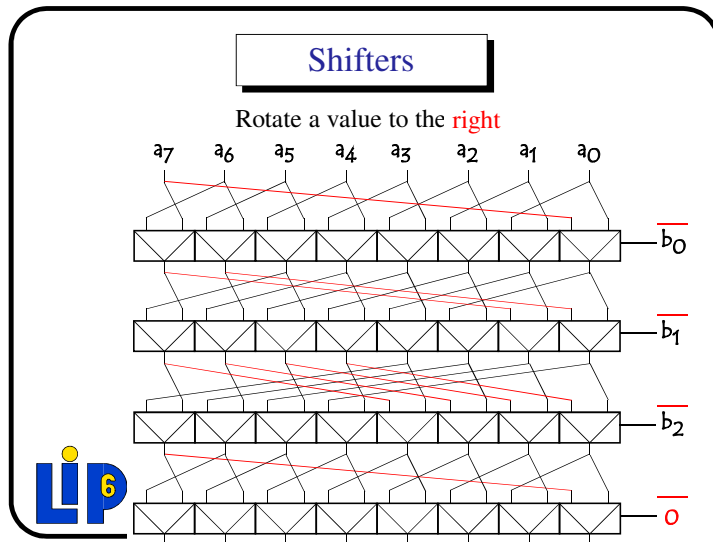
=
Rotate left by \overline{B} followed by a 1 position rotate



Pirouz Bazargan Sabet

Digital Design

February 2010



Shifters

Shift / Rotate a value to the left / right

The diagram illustrates a 4-bit barrel shifter. The input is an 8-bit value $a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0$. The output is a 4-bit value $b_3 b_2 b_1 b_0$. The shifter consists of four stages, each performing a shift operation. The stages are labeled b_0 , b_1 , b_2 , and b_3 . The input is shifted by 1, 2, 3, and 4 positions respectively in each stage. The output of the fourth stage is b_3 . A dashed arrow indicates a feedback loop from the output back to the input.

Shifters

Shift / Rotate a value to the left / right

The diagram illustrates a 4-stage barrel shifter architecture. At the top, a header box contains the title "Shifters" and the instruction "Shift / Rotate a value to the left / right". Below this, the input bits are labeled $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$. The architecture consists of four horizontal rows of logic blocks, each representing a stage of the shifter. Red lines show the data flow from the input bits through the stages. The output bits are labeled $0, b_0, b_1, b_2$ on the right. The first row (stage 0) shows the input bits connected to the output 0 . The second row (stage 1) shows the input bits connected to the output b_0 . The third row (stage 2) shows the input bits connected to the output b_1 . The fourth row (stage 3) shows the input bits connected to the output b_2 . The diagram demonstrates how the shifter can perform a left shift by one position (e.g., $a_7 \rightarrow b_0$) or a right shift by one position (e.g., $a_0 \rightarrow b_2$).

Shifters

Shift / Rotate a value to the left / right

The diagram illustrates a 3-stage barrel shifter. The inputs are labeled a_7 through a_0 from left to right. The outputs are labeled b_0 , b_1 , and b_2 from top to bottom. Red lines show the bit paths: b_0 is a_0 shifted right by 1; b_1 is a_0 shifted right by 2, a_1 shifted right by 1, and a_2 shifted right by 0; b_2 is a_0 shifted right by 3, a_1 shifted right by 2, a_2 shifted right by 1, and a_3 shifted right by 0. Blue and green dots mark the bit positions at each stage.

Shifters

Shift / Rotate a value to the left / right

The diagram illustrates a 4-stage barrel shifter architecture. It consists of four rows of logic blocks, each representing a stage. The input bits are labeled $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$ at the top. The output bits are labeled $0, b_0, b_1, b_2$ on the right. Red lines show the data paths between stages. The first stage shifts the input bits one position to the left. The second stage shifts the result two positions to the left. The third stage shifts the result three positions to the left. The fourth stage shifts the result four positions to the left. The final output is b_2 . The diagram also shows a rotation operation where the output of the fourth stage is shifted back to the original position of the input bit a_0 , indicated by a red dashed circle and arrow.

Shifters

Shift / Rotate a value to the left / right *optimized*

