Outline

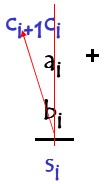
- Digital CMOS design
- Arithmetic operators
  - Adders
  - Comparators
  - Shifters
  - Multipliers

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

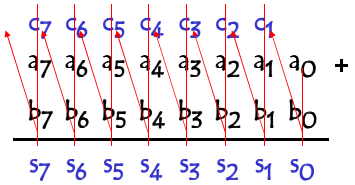


s_i and c_{i+1} are Boolean functions of a_i b_i c_i

Adders

Adding two natural numbers

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



Adders

Adding two natural numbers

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

a_i b_i

c_i

s_i

$s_i = a_i \oplus b_i \oplus c_i$

$s_i = \bar{a}_i.\bar{b}_i.c_i + \bar{a}_i.b_i.\bar{c}_i + a_i.b_i.c_i + a_i.\bar{b}_i.\bar{c}_i$

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

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

Adders

Adding two natural numbers

	00	01	11	10	$a_i\ b_i$
0	0	1	0	1	
1	1	0	1	0	

c_i

s_i

	00	01	11	10	$a_i\ b_i$
0	0	0	1	0	
1	0	1	1	1	

c_i

c_{i+1}

$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i.b_i + a_i.c_i + b_i.c_i$

Adders

Adding two natural numbers

Considering the delay

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

$a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0\ +$

$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$

$s_7\ s_6\ s_5\ s_4\ s_3\ s_2\ s_1\ s_0$

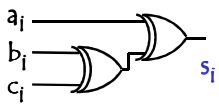
Adders

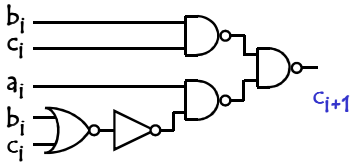
Adding two natural numbers

$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i.b_i + a_i.c_i + b_i.c_i$

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





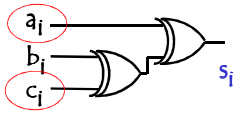
Adders

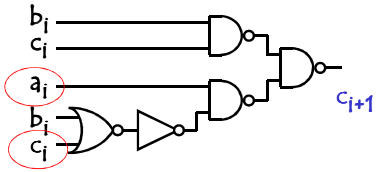
Adding two natural numbers

$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i.b_i + a_i.c_i + b_i.c_i$

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





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

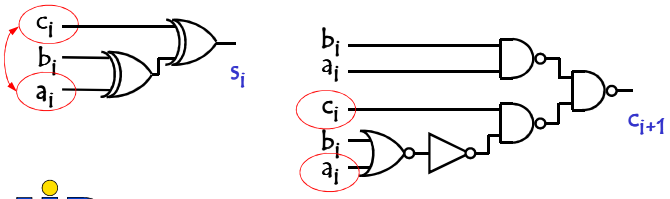
Adders

Adding two natural numbers

$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i.b_i + a_i.c_i + b_i.c_i$

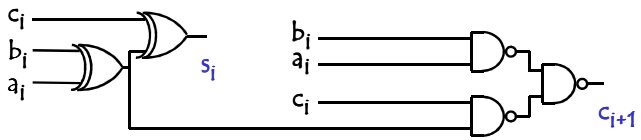
$c_{i+1} = a_i.b_i + (a_i+b_i) . c_i$



Adders

Adding two natural numbers

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



Adders

Adding two natural numbers

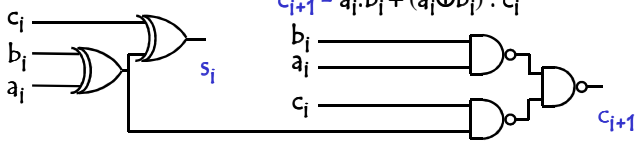
$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i.b_i + a_i.c_i + b_i.c_i$

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

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

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

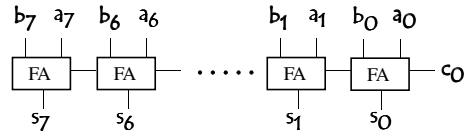


Adders

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



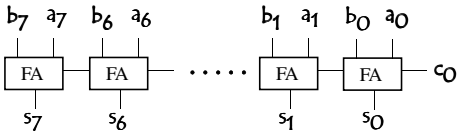
Ripple Carry Adder (RCA)

Adders

Adding two natural numbers

Ripple Carry Adder (RCA)

Area ∝ n Delay ∝ n



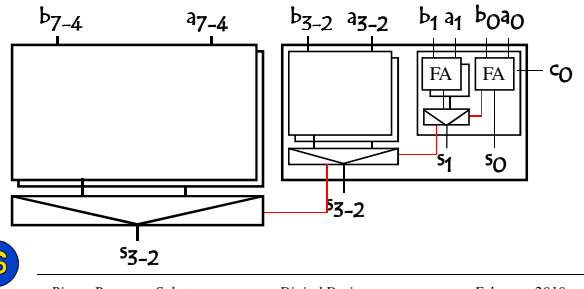
Timing should be improved

Adders

Adding two natural numbers

Carry Select Adder (CSLA)

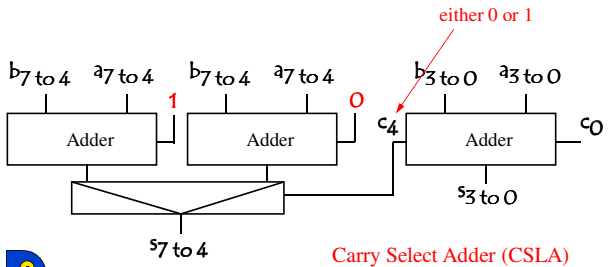
Area ∝ $3^{log(n)} = n^{log(3)} = n^{1.585}$ Delay ∝ *log* (n)



Adders

Adding two natural numbers

Acceleration technics

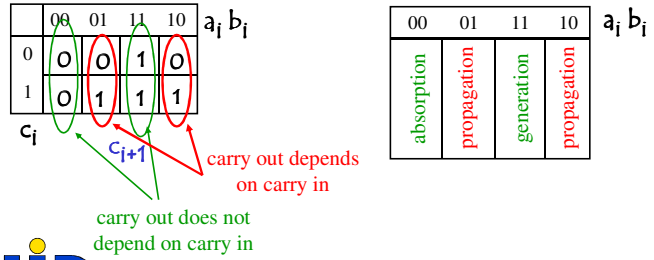


Carry Select Adder (CSLA)

Adders

Adding two natural numbers

Acceleration technics



Adders

Adding two natural numbers

Acceleration technics

	00	01	11	10	$a_i\ b_i$
0	0	0	1	0	
1	0	1	1	1	

c_i c_{i+1}

00	01	11	10	$a_i\ b_i$
absorption	propagation	generation	propagation	

$G_i = a_i\ b_i$

$P_i = a_i \oplus b_i$

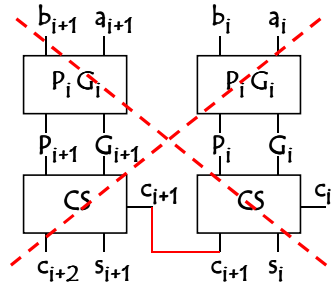
$c_{i+1} = G_i + P_i\ c_i$

$s_i = P_i \oplus c_i$

Adders

Adding two natural numbers

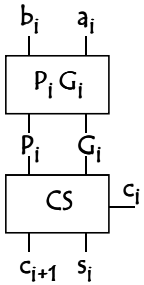
Acceleration technics



Adders

Adding two natural numbers

Acceleration technics



00	01	11	10	$a_i\ b_i$
absorption	propagation	generation	propagation	

$G_i = a_i\ b_i$

$P_i = a_i \oplus b_i$

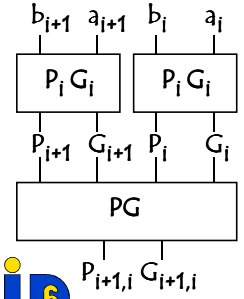
$c_{i+1} = G_i + P_i\ c_i$

$s_i = P_i \oplus c_i$

Adders

Adding two natural numbers

Acceleration technics



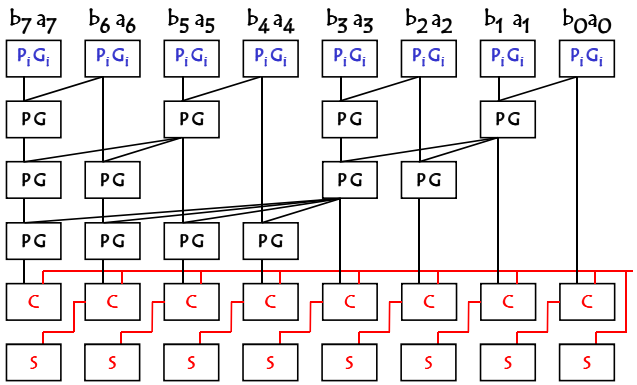
$G_{i+1} = a_{i+1}\ b_{i+1}$ $G_i = a_i\ b_i$

$P_{i+1} = a_{i+1} \oplus b_{i+1}$ $P_i = a_i \oplus b_i$

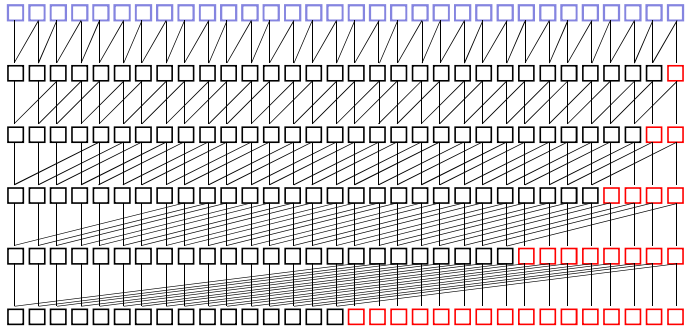
$G_{i+1,\,i} = G_{i+1} + G_i . P_{i+1}$

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

Adders

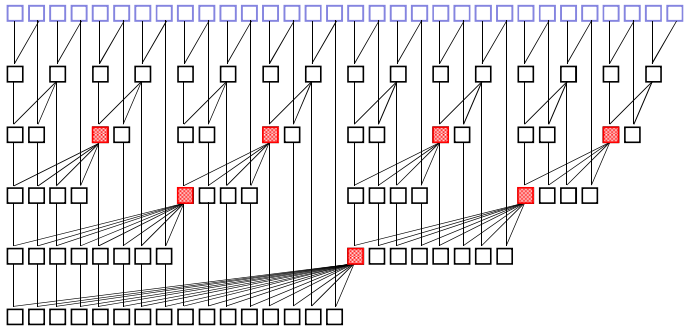


Adders



Kogge-Stone Adder

Adders



Slansky 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)