

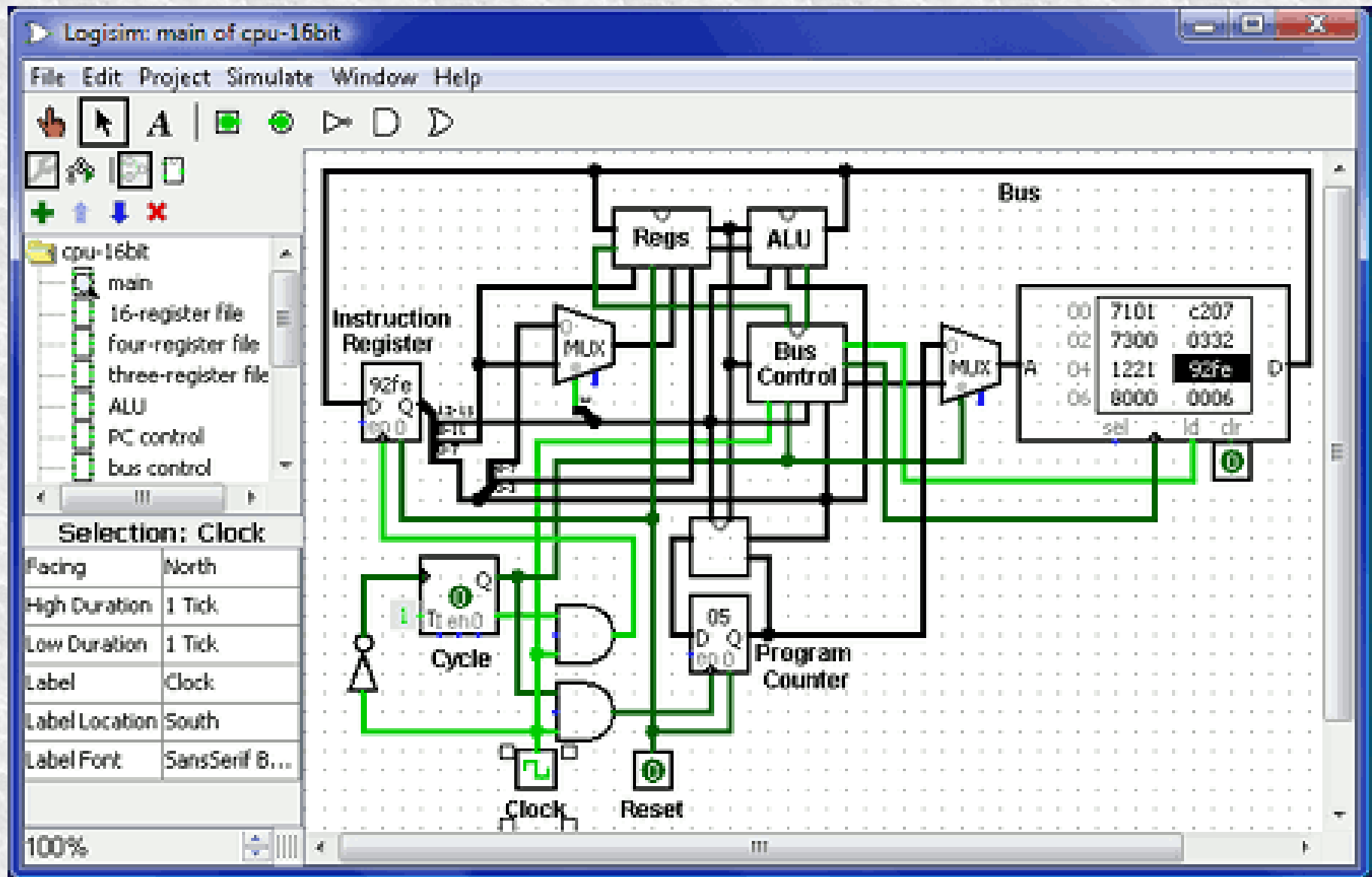
Компютърни архитектури CSCB008

Приложения на комбинационните логически схеми в
компютърните архитектури

доц. д-р Ясен Горбунов
2021

Софтуер за логически синтез и симулация

LogiSim - <http://www.cburch.com/logisim/>



Logic Friday - <https://sontrak.com/>

File Operation TruthTable Equation Gates View Help

Funct...	Inputs	Outputs	True	False	DC	PI	Gates
F0	3	1	5	3	0	2	3

X	M	C	F0
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Entered by truthtable:

$$F_0 = A' M C' + A' M C + A M' C + A M C' + A M C;$$

Minimized:

$$F_0 = M + A C;$$

Ready

Елементи на микроархитектурата

Полусуматор

Пълен суматор

Суматор със знак

Умножител

Мултиплексор

Демултиплексор

Декодер

часть 1

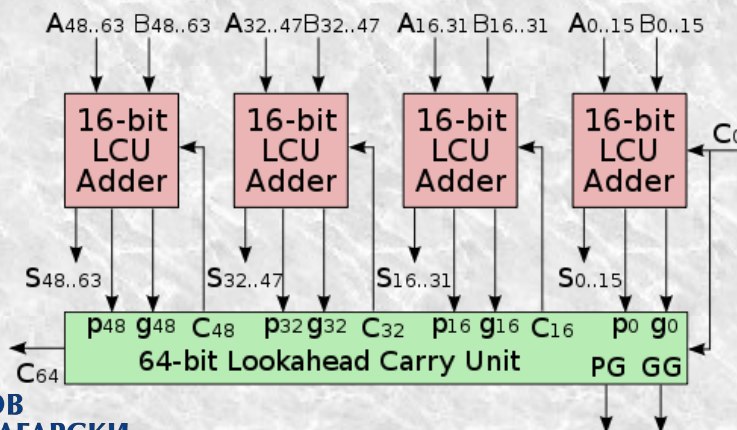
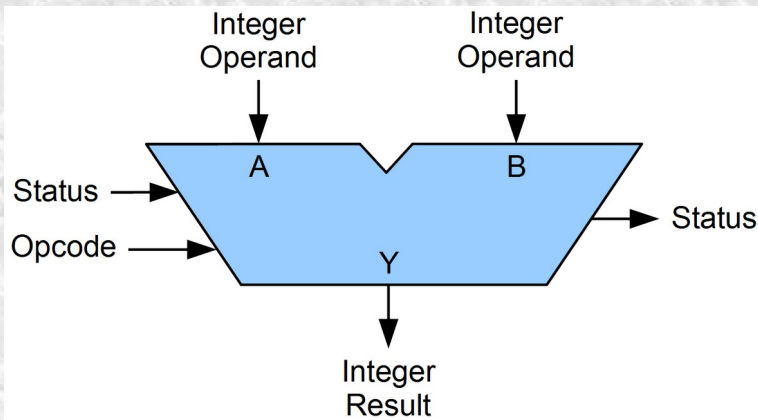
часть 2

Суматор

В основата на цифровите компютри
стои централния процесор
CPU (Central Processing Unit)

В основата на CPU стои
аритметично-логическото устройство
ALU (Arithmetic-Logic Unit)

В основата на ALU стоят
сумиращите устройства и
мултиплексорите



Проектиране на полусуматор

Half Adder

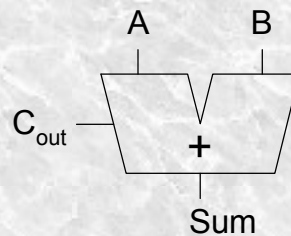


таблица на истинност

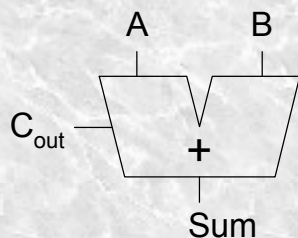
A	B	C _{out}	Sum
0	0		
0	1		
1	0		
1	1		

Sum =

C_{out} =

Проектиране на полусуматор

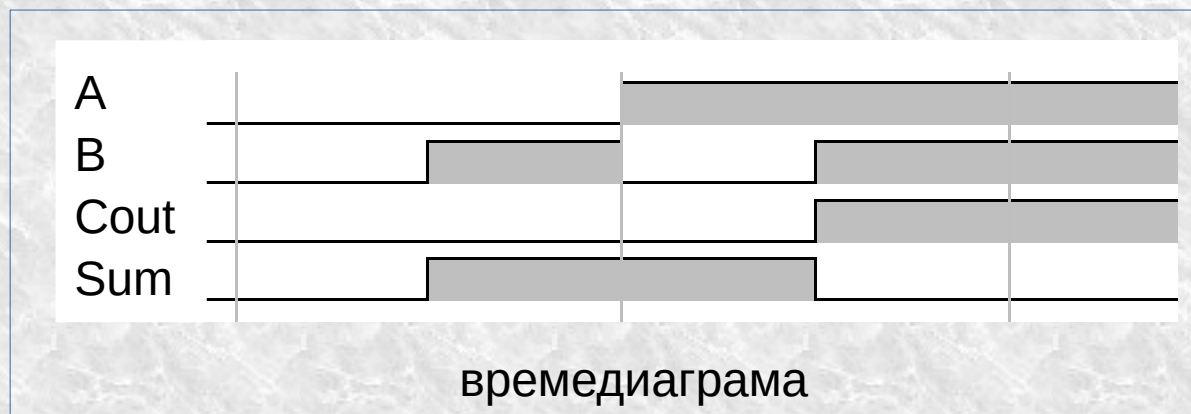
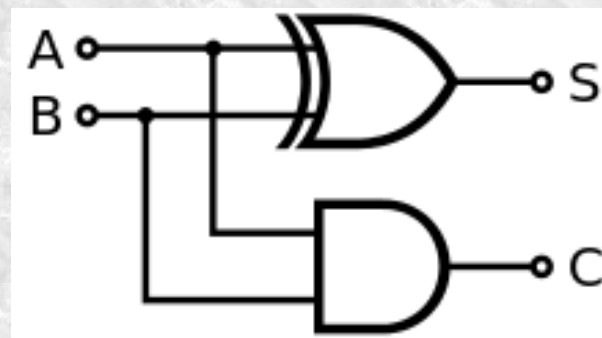
Half Adder



A	B	C _{out}	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

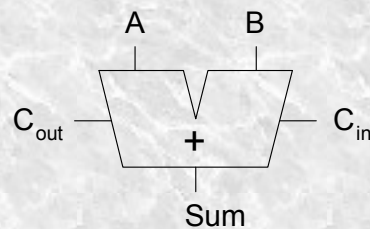
$$\text{Sum} = A \oplus B$$

$$C_{\text{out}} = AB$$



Проектиране на пълен суматор

Full Adder



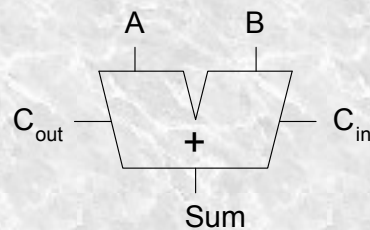
C _{in}	A	B	C _{out}	Sum
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Sum =

C_{out} =

Проектиране на пълен суматор

Full Adder

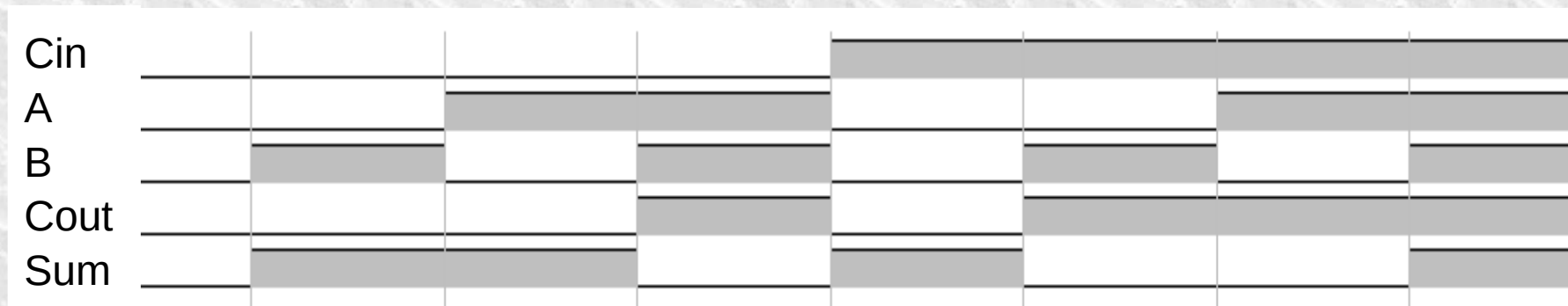
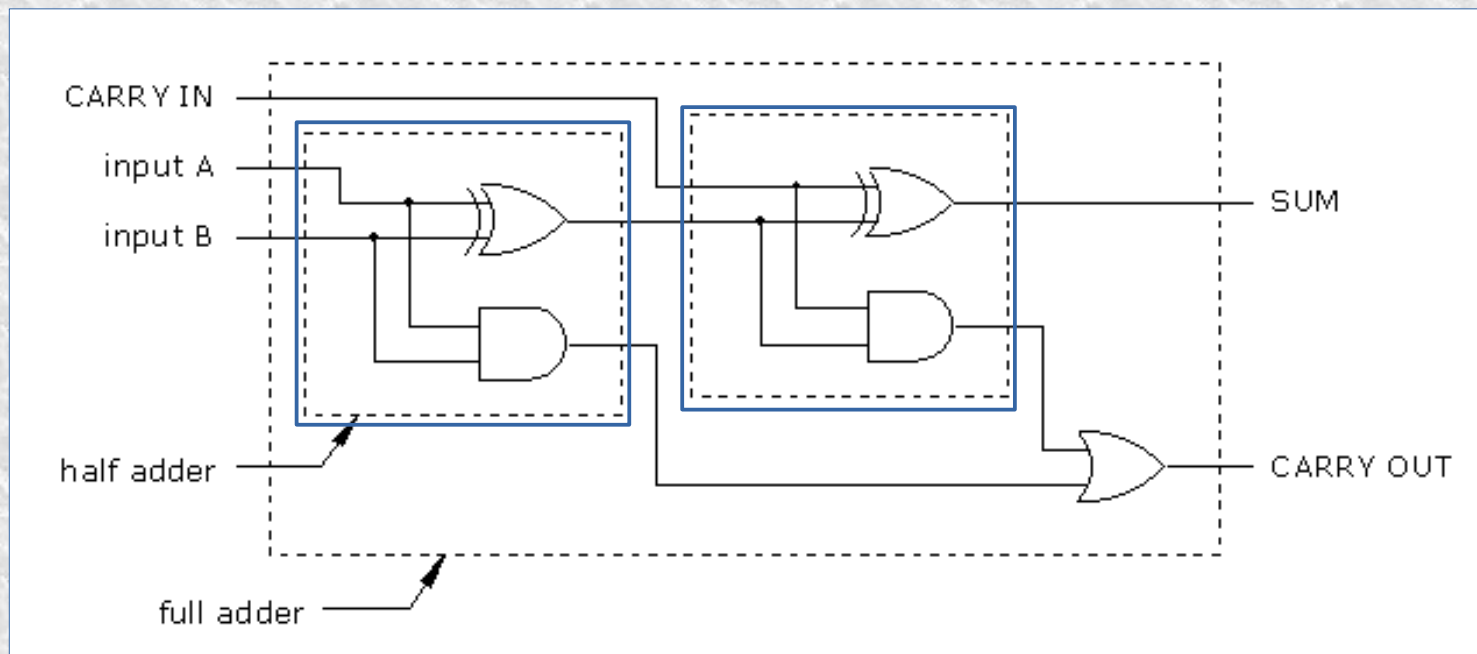


C_{in}	A	B	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

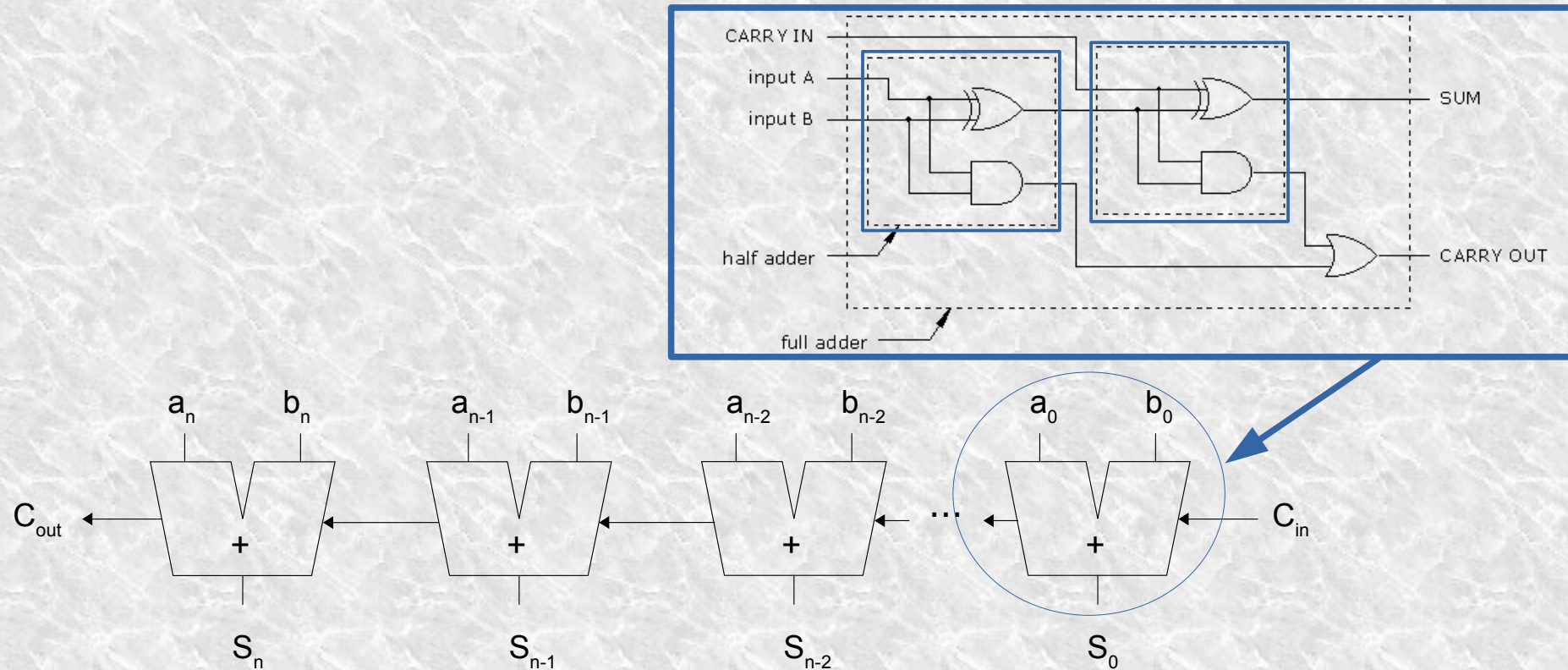
$$\text{Sum} = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

Проектиране на пълен суматор



Пълен суматор с последователен пренос – **Ripple-Carry Adder**



- сумирането зависи от последователното преминаване на флага за пренос през всички стъпала

- закъснение при този вид суматор

$$t_{\text{RIPPLE}} = N \cdot t_{\text{FA}}$$

Пълен суматор с ускорен (предвиден) пренос Carry-Lookahead Adder – CLA

- разделяне на суматора на *блокове* (обикновено 4-битови)
- генериране на два сигнала, описващи как *колона* или *блок* определят преноса C_{out}

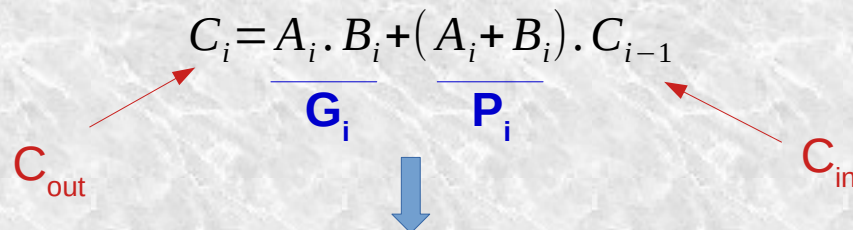
$$G_i = A_i \cdot B_i$$

generate (G) - ако събирането води до пренос независимо от C_{in}

$$P_i = A_i + B_i$$

propagate (P) – ако има пренос C_{in} И $A_i=1$ или $B_i=1$

за всеки следващ пренос C_i е валидно

$$C_i = \underbrace{A_i \cdot B_i}_{G_i} + \underbrace{(A_i + B_i)}_{P_i} \cdot C_{i-1}$$


$$C_i = G_i + P_i \cdot C_{i-1}$$

Пълен суматор с ускорен (предвиден) пренос Carry-Lookahead Adder – CLA

пример за колони (3:0)

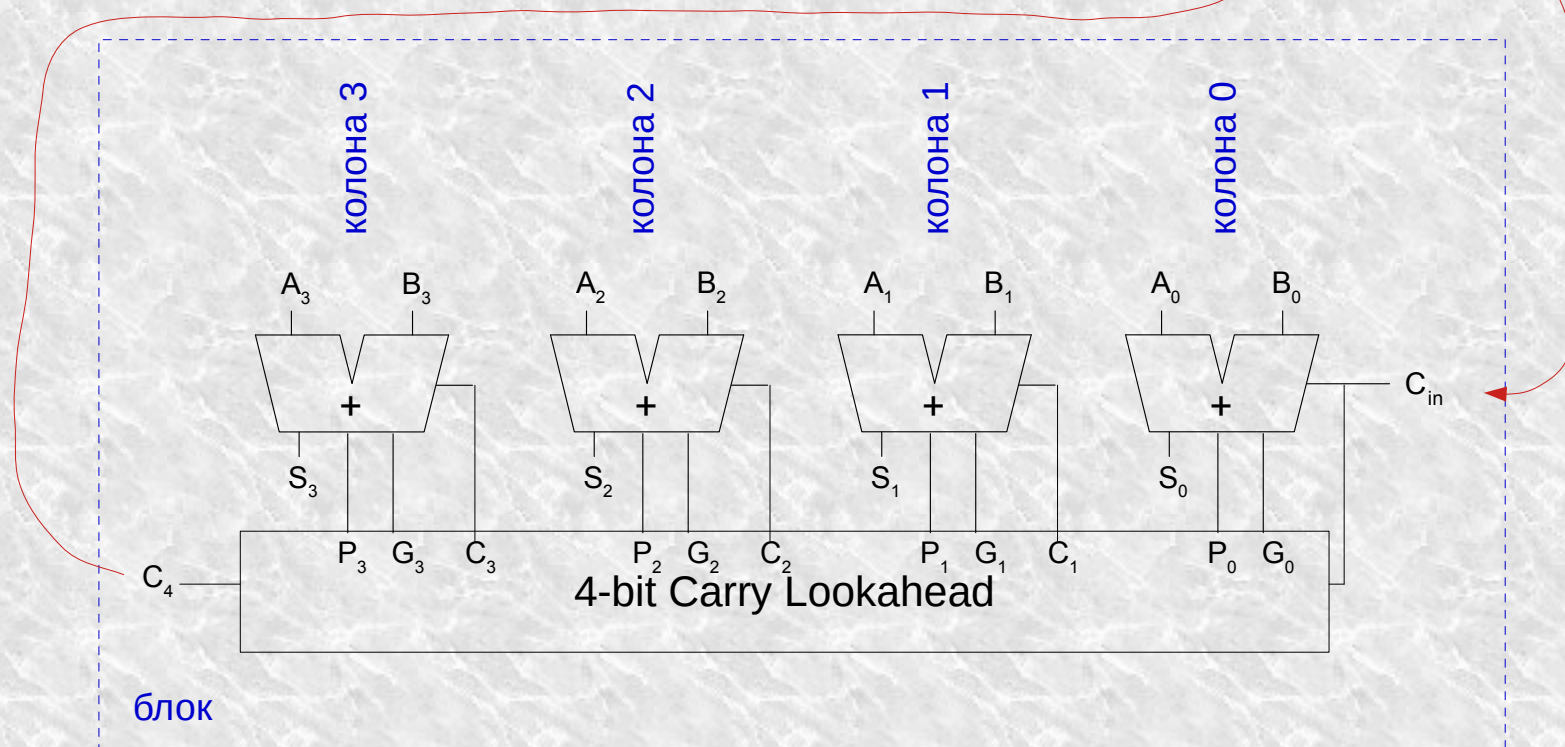
$$G_{3:0} = G_3 + P_3 \cdot (G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0))$$

$$P_{3:0} = P_3 \cdot P_2 \cdot P_1 \cdot P_0$$

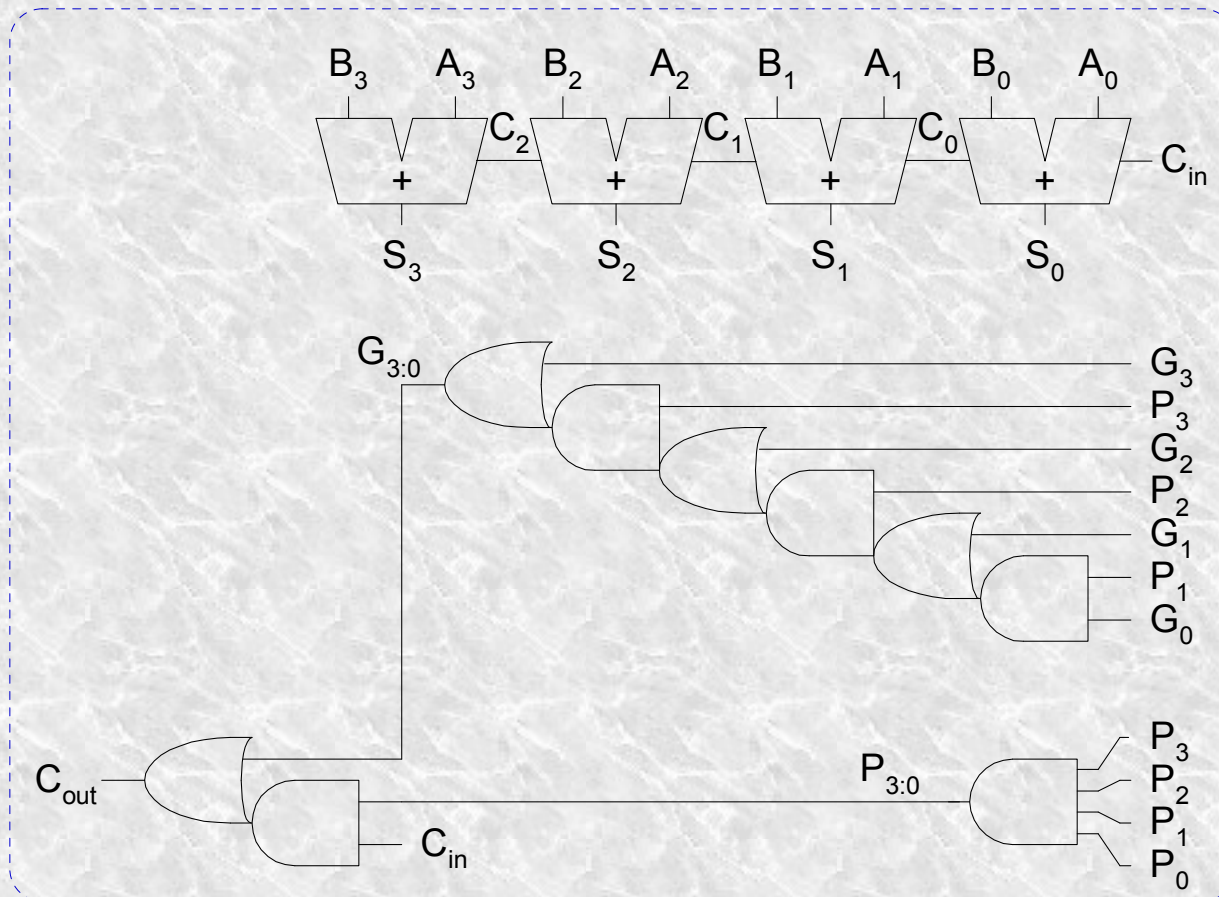
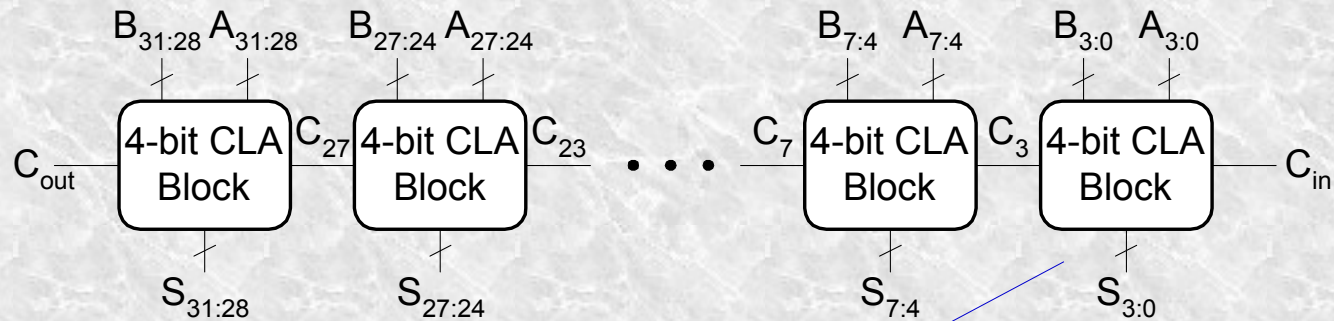


пренос за блока

$$C_i = G_{i:j} + P_{i:j} \cdot C_j$$



Суматор



$$G_{3:0} = G_3 + P_3 \cdot (G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0))$$

$$P_{3:0} = P_3 \cdot P_2 \cdot P_1 \cdot P_0$$

Суматор

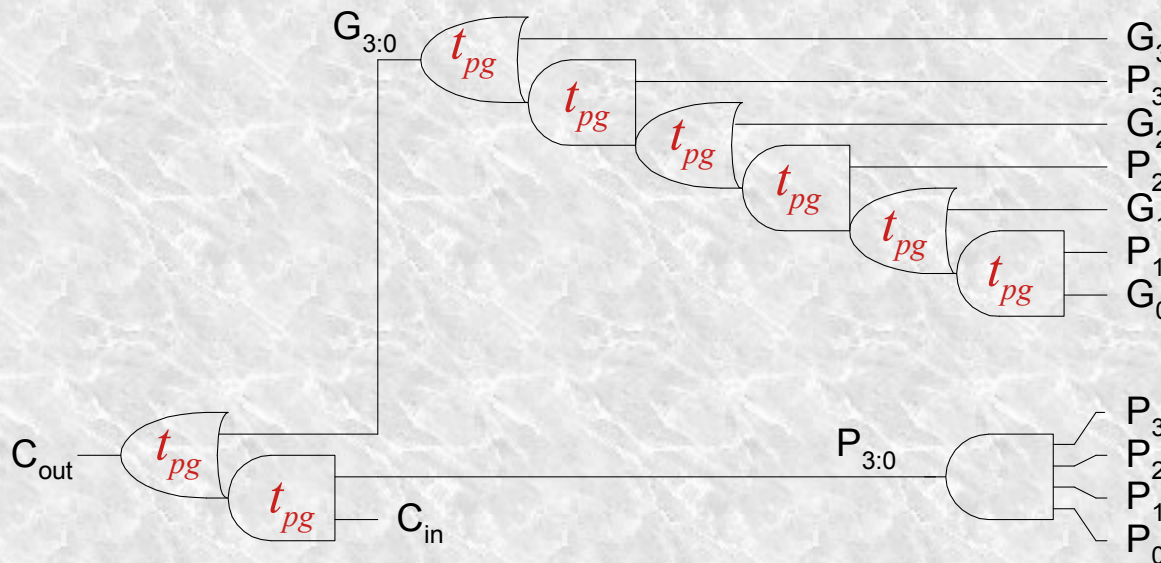
За N -bit CLA с k -bit блокове закъснението е:

$$t_{CLA} = t_{pg} + t_{pg_block} + \left(\frac{N}{k} - 1\right) \cdot t_{AND_OR} + k \cdot t_{FA}$$

t_{pg} : закъснение на отделните лог. елементи (gates) при генериране на P_i и G_i

t_{pg_block} : закъснение при генериране на P_{ij} и G_{ij} за k -bit блок

t_{AND_OR} : закъснение от C_{in} към C_{out} през последните AND/OR лог. елементи на k -bit CLA блок



$$t_{pg_block} = 6 \cdot t_{pg}$$
$$t_{AND_OR} = 2 \cdot t_{pg}$$

За N -bit CLA с k -bit блокове закъснението е:

$$t_{CLA} = t_{pg} + t_{pg_block} + \left(\frac{N}{k} - 1\right) \cdot t_{AND_OR} + k \cdot t_{FA}$$

t_{pg} : закъснение на отделните лог. елементи (gates) при генериране на P_i и G_i

t_{pg_block} : закъснение при генериране на P_{ij} и G_{ij} за k -bit блок

t_{AND_OR} : закъснение от C_{in} към C_{out} през последните AND/OR лог. елементи на k -bit CLA блок

N -битов суматор с ускорен (предвиден) пренос е много по-бърз от суматор с последователен пренос при $N > 16$.

Пример – сравнение на 32-bit RCA и 32-bit CLA (CLA group 4-bit) суматори.

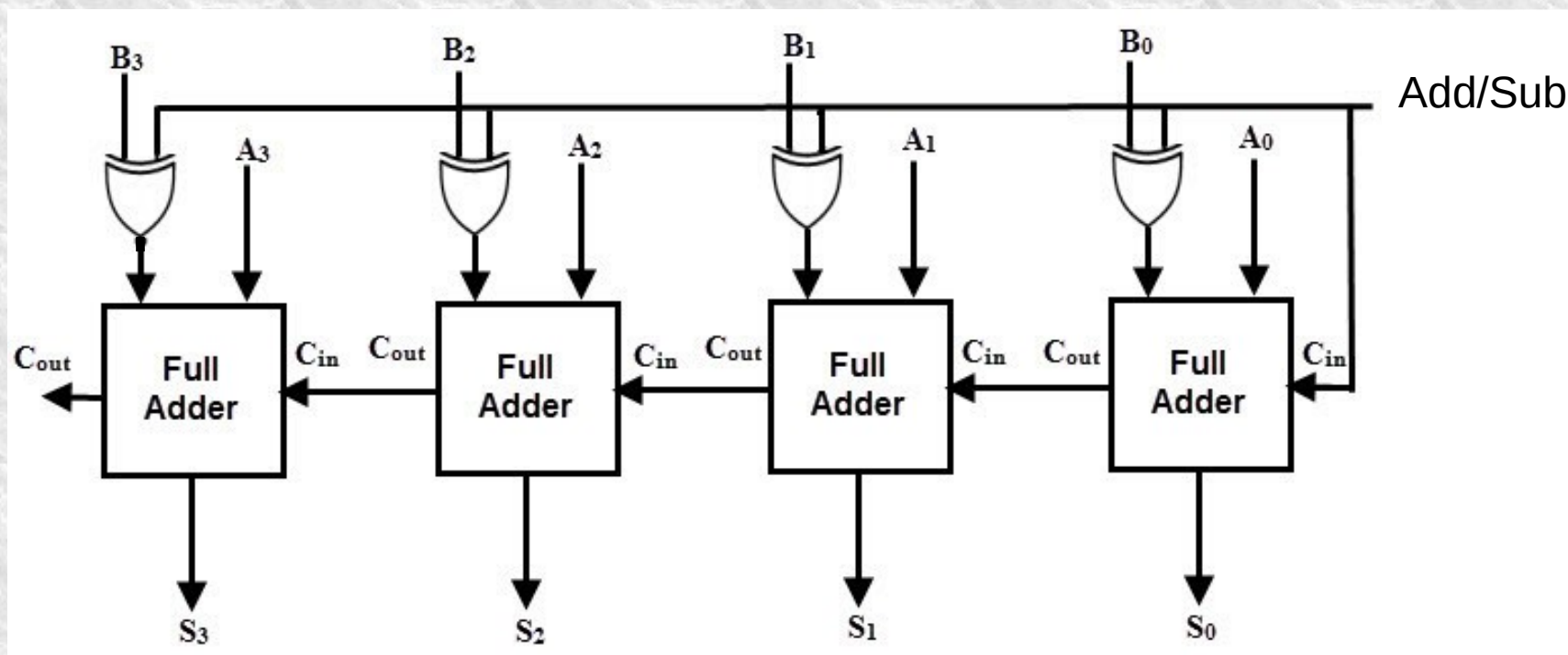
$$t_{pg} = 100\text{ps}; t_{FA} = 300\text{ps}$$

$$t_{RCA} = 9.6 \text{ ns}$$

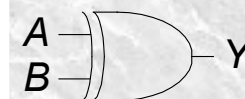
$$t_{CLA} = 3.3 \text{ ns}$$

Суматор

Проектиране на пълен суматор / субтрактор



XOR



$$Y = A \oplus B$$

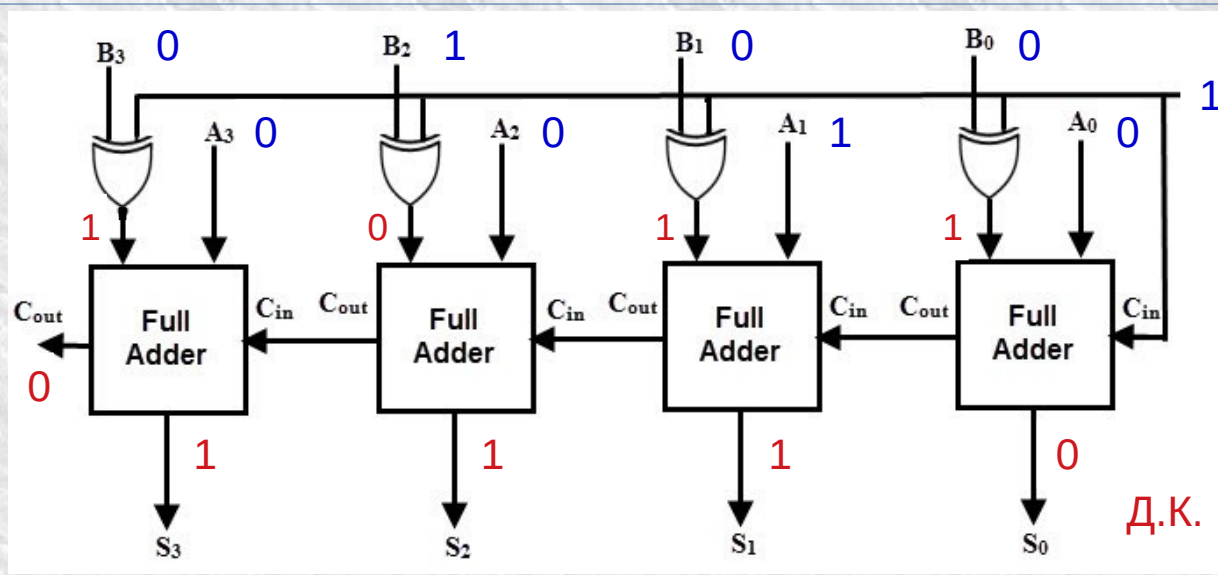
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

XOR – инвертиране на битове от B_i
при Add/Sub=0 изходът на XOR повтаря B

C_{in} – 1) определя типа на операцията
– 2) добавяне на 0 или 1

допълнителен код

Проектиране на пълен суматор / субтрактор - пример



$$2 - 4 = -2$$

$$A = 2_{\text{DEC}} \rightarrow 0010$$

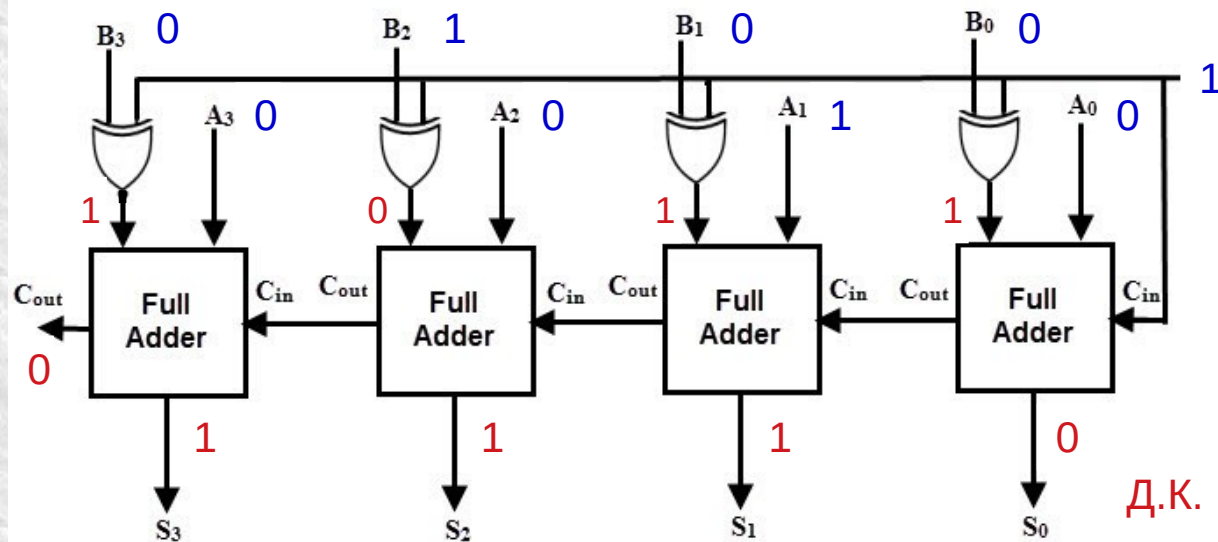
$$B = 4_{\text{DEC}} \rightarrow 0100$$

$$-4 \rightarrow (1011 + 1 = 1100 \text{ Д.К.})$$

$$\begin{array}{r}
 0010 \\
 + \quad A \\
 + \quad \sim B \\
 + \quad \text{Cin} \\
 \hline
 1110 \text{ Д.К.}
 \end{array}$$

Суматор

Проектиране на пълен суматор / субтрактор - пример



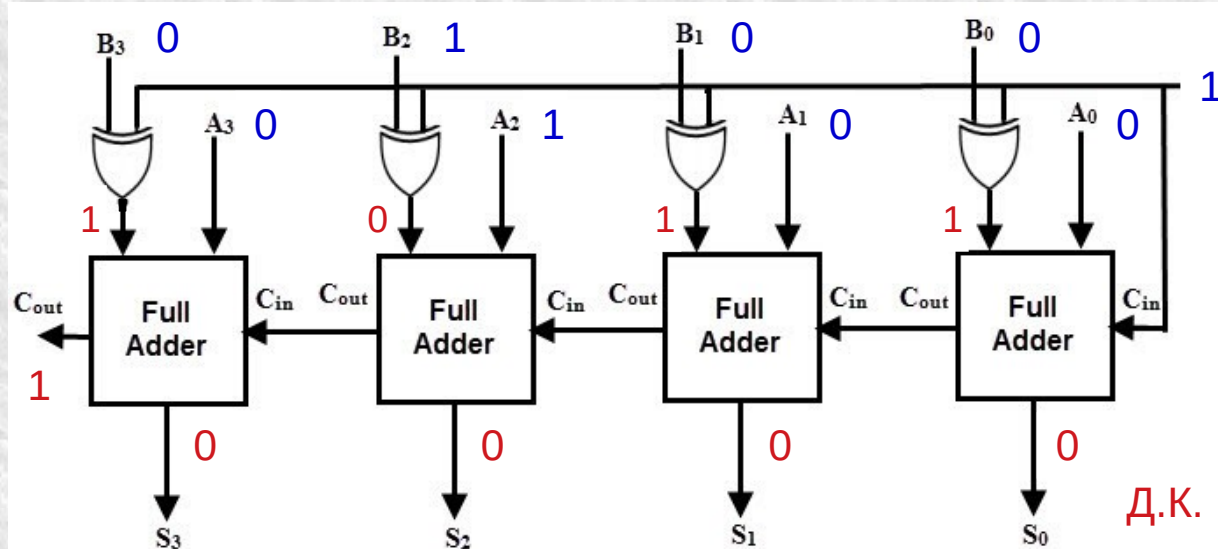
$$2 - 4 = -2$$

$$A = 2_{DEC} \rightarrow 0010$$

$$B = 4_{DEC} \rightarrow 0100$$

$$-4 \rightarrow (1011 + 1 = 1100 \text{ Д.К.})$$

$$\begin{array}{r} 0010 \\ + \quad \sim B \\ + \quad C_{in} \\ \hline 1110 \text{ Д.К.} \end{array}$$



$$4 - 4 = 0$$

$$A = 4_{DEC} \rightarrow 0100$$

$$B = 4_{DEC} \rightarrow 0100$$

$$-4 \rightarrow (1011 + 1 = 1100 \text{ Д.К.})$$

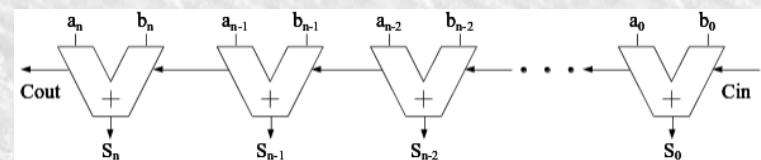
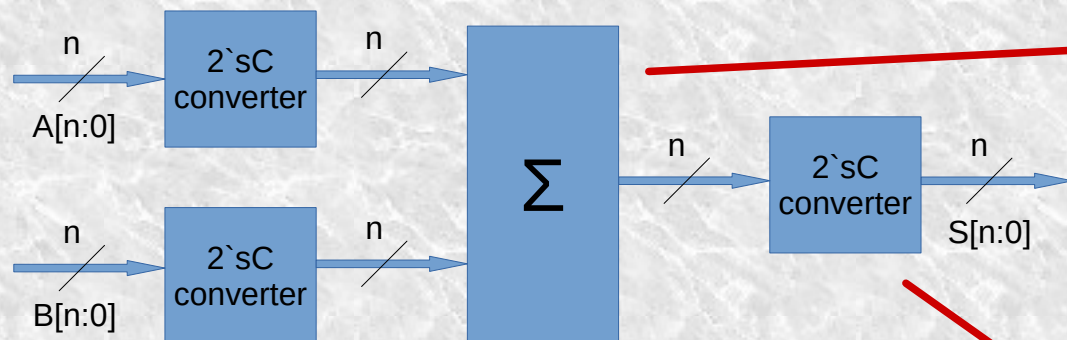
$$\begin{array}{r} 0100 \\ + \quad \sim B \\ + \quad C_{in} \\ \hline 10000 \text{ Д.К.} \end{array}$$

$$C_{out} \rightarrow 10000 \text{ Д.К.}$$

Суматор

Проектиране на пълен суматор със знак

ако и двата операнда притежават знак → **допълнителен код**



Допълнителния код на допълнителния код дава правия код!

Кодов преобразувател

подаване на 0 към единия вход на пълен суматор
→ опростяване на уравненията

C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$


C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	0	0	0	0
0	0	1	0	1
1	0	0	0	1
1	0	1	1	0
1	0	0	0	1
1	0	1	1	0

$$S = B \oplus C_{in}$$

$$C_{out} = C_{in} \cdot B$$
