Predstavitev števil in osnove dvojiške aritmetike

Digitalna vezja

Miha Moškon

miha.moskon@fri.uni-lj.si

https://fri.uni-lj.si/en/about-faculty/employees/miha-moskon

Dvojiški zapis števil

Nepredznačena števila: z *n* biti lahko predstavimo števila od 0 do 2^{n-1}.

Pozicijski zapis števil: vsak bit ima svojo težo glede na pozicijo (enako velja za desetiški, osmiški, šestnajstiški ali katerikoli drug sistem)

$$D = \sum_{i=0}^{n-1} d_i \cdot r^i$$

MSB: most significant bit (teža $i = n - 1$)

LSB: least significant bit (teža $i = 0$)

Šestnajskiški zapis števil

Bolj kompaktna predstavitev

Z eno števko (nibble) predstavimo 4 bite

od 0 do F

Predpona 0x (hex)

Primer: 255 = 0xFF = 11111111

Enostavna pretvorba med bin in hex (jemljemo po 4 bite naenkrat)

Seštevanje dvojiških števil

Postopek seštevanja

S = A + B

S … vsota

A,B: seštevanca (operanda)

Polovični seštevalnik (*half adder,* HA)

$$\begin{array}{cc} & x \\ + & y \\ \hline c & s \end{array}$$

Prenos (*Carry*)

Vsota (*Sum*)

$$\begin{array}{cc} & 0 \\ + & 0 \\ \hline 0 & 0 \end{array} \qquad \begin{array}{cc} & 0 \\ + & 1 \\ \hline 0 & 1 \end{array} \qquad \begin{array}{cc} & 1 \\ + & 1 \\ \hline 1 & 0 \end{array}$$

$$\begin{array}{cc} & 1 \\ + & 0 \\ \hline 0 & 1 \end{array}$$

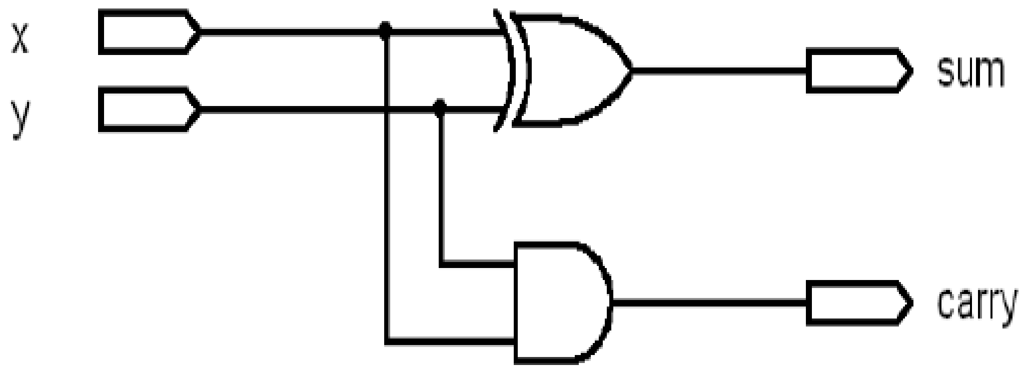
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Polovični seštevalnik

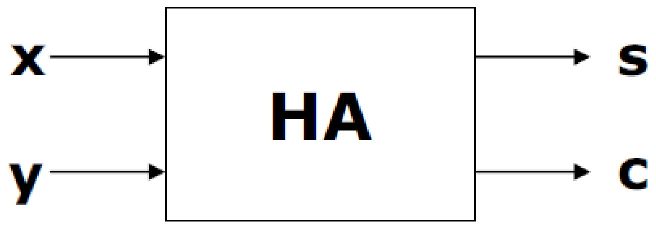
Osnovna izvedba:

$$s = x \nabla y$$

$$c = x \cdot y$$



x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Polni seštevalnik (*full adder*, FA)

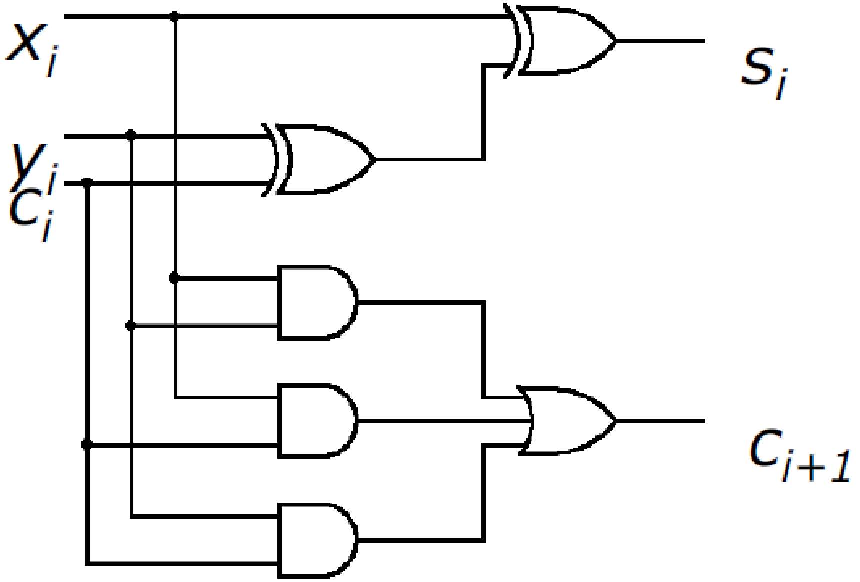
Kot vhod nastopa tudi carrry (kot rezultat seštevanja na manj pomembnem bitu)

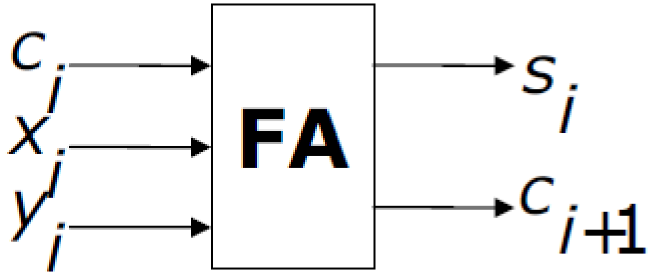
c_i	x_i	y_i	c_{i+1}	s_{i+1}
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$s_{i+1} = x_i \nabla y_i \nabla c_i$$

$$c_{i+1} = x_i y_i \vee y_i c_i \vee x_i c_i$$

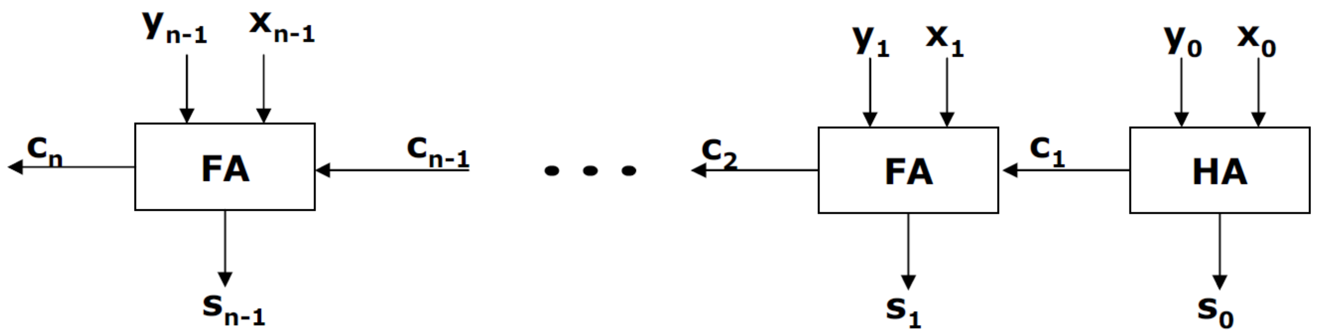
Polni seštevalnik





Ripple (Carry) Adder – plazoviti seštevalnik

Seštevanje n-bitnih števil z n-1 polnimi seštevalniki in enim polovičnim:



Odštevanje dvojiških števil

D = A – B

D … razlika

A … zmanjševanec

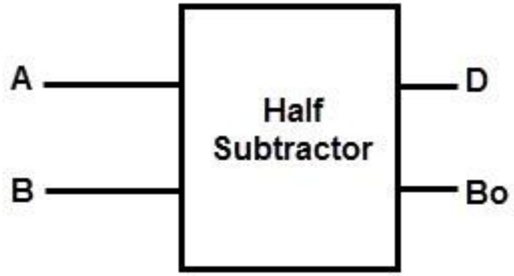
B … odštevanec

Postopek odštevanja

Polovični odštevalnik

D – difference

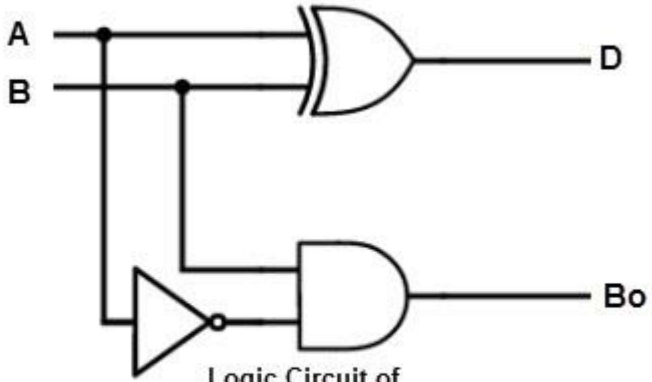
B – borrow



Block Diagram

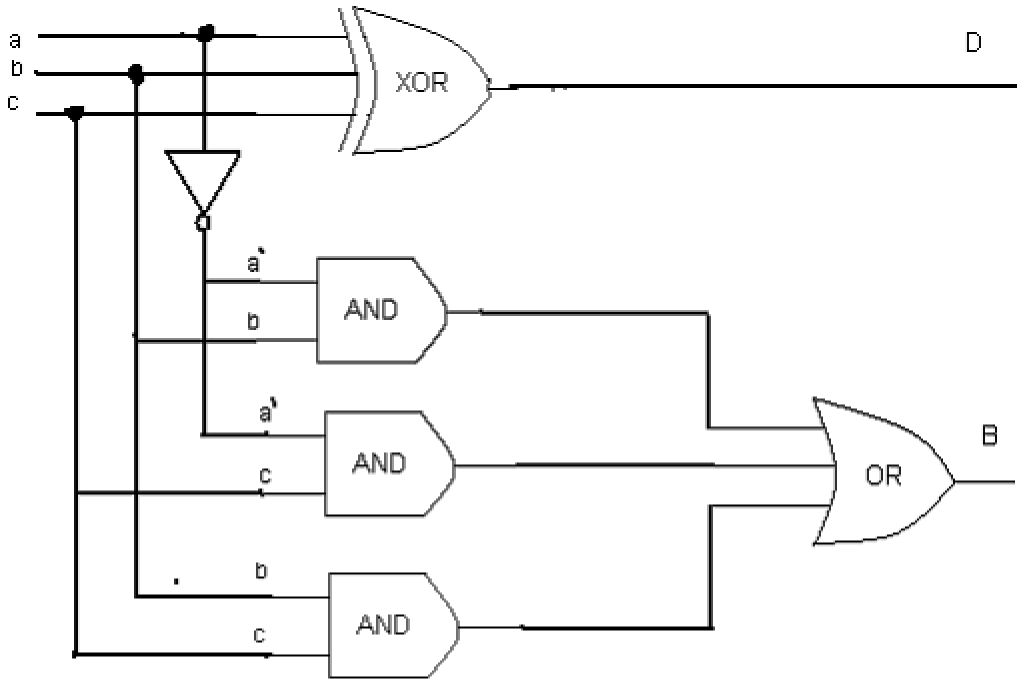
A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table



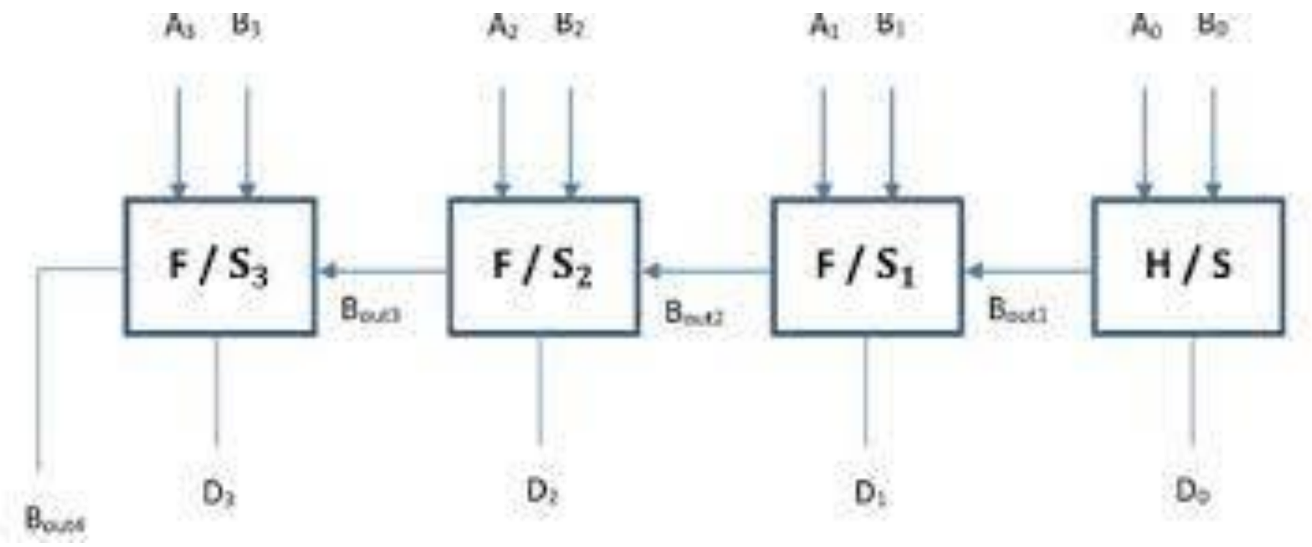
Logic Circuit of Half Subtractor

Polni odštevalnik



Ripple (Borrow) Subtractor – plazoviti odštevalnik

Odštevanje n-bitnih števil z n-1 polnimi odštevalniki in enim polovičnim:



Zapis predznačenih števil

- Zapis predznak in velikost
- Zapis z odmikom
- Dvojiški komplement

Zapis predznak in velikost

Dvodelna predstavitev: predznak (1 bit) + velikost (ostali biti)

MSB bit določa predznak
- 0: pozitivno število
- 1: negativno število

ostali biti določajo velikost števila

prednost
- lahko berljiv zapis (za človeka)

slabosti
- posebna obravnava MSB bita pri aritmetičnih operacijah
- seštevanje/odštevanje: primerjati moramo velikosti operandov, da določimo predznak rezultata
- dve ničli

Primer: 3-bitno število

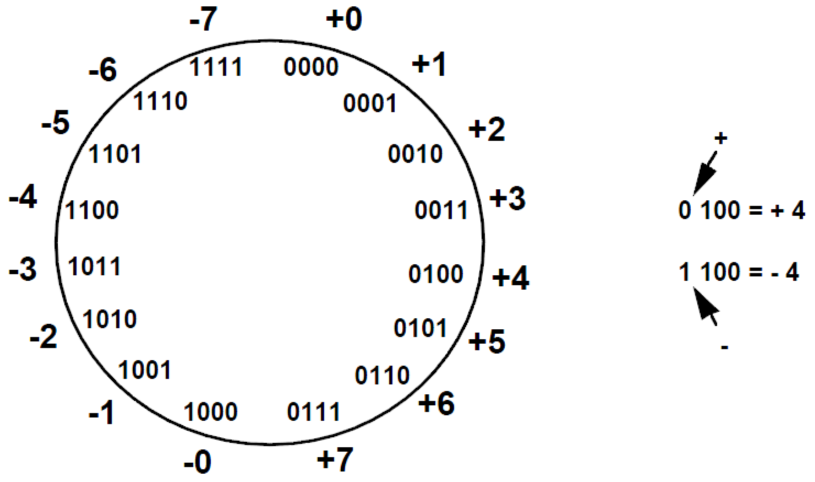
binarno	vrednost
000	+0
001	+1
010	+2
011	+3
100	-0
101	-1
110	-2
111	-3

Zapis predznak in velikost

Obseg števil za n bitov:

$$\pm 2^{n-1} - 1$$

Primer: 4-bitno število



Zapis z odmikom

množico vrednosti razpolovimo na negativne in pozitivne

kjer smo razpolovili, imamo 0 – **odmik**

Ponavadi $2^{n-1} - 1$

primer: 3-bitno število

- odmik = 3
- vrednost = zapis – 3

prednosti:

- samo ena ničla
- relacija < se ohranja

slabosti:

- pri aritmetičnih operacijah moramo odmik eksplicitno upoštevati

Uporaba: IEEE 754 – zapis eksponenta (bias) z odmikom – uporaba dvojiškega komplementa bi otežila primerjavo števil

Primer: 3-bitno število

binarno	vrednost
000	-3
001	-2
010	-1
011	**0**
100	+1
101	+2
110	+3
111	+4

Zapis z odmikom

primer seštevanja dveh števil:

$$-3_{10} + 1_{10} = 000_2 + 100_2 = 100_2 = 1_{10}$$

Rezultat je napačen! Zakaj?

x, y … števili, ki ju seštevamo

a, b … vrednosti števil zapisanih z odmikom

d … odmik

$a + b = x - d + y - d = x + y - 2d$

V primeru seštevanja, dobimo torej dva odmika.

Kaj pa odštevanje in množenje?

Primer: 3-bitno število

binarno	vrednost
000	-3
001	-2
010	-1
011	**0**
100	+1
101	+2
110	+3
111	+4

Dvojiški komplement

Dvojiški komplement (negativno vrednost) dobimo tako, da število odštejemo od 2^n: $Y = 2^n – X$

Na tak način lahko dobimo komplement števila v poljubnem številskem sistemu (odštevanje od r^n, r – radix oziroma baza).

Problem: odštevanje je potratna operacija.

Rešitev: število 2^n=100…0 zapišemo kot $(2^n-1)+1$ = 011…1 + 1

Odštevanje X od števila 011…1 je enako invertiranju vseh bitov številu X

Na koncu moramo prišteti še enico.

Dvojiški komplement Y torej dobimo tako, da invertiramo vse bite števila X in temu prištejemo 1.

Prednosti:

- samo ena ničla
- ni težav z aritmetičnimi operacijami

Slabost:

- negativna števila so večja od pozitivnih (MSB pove za kakšno število gre in ga moramo upoštevati pri primerjanju velikosti)

Primer: 3-bitno število

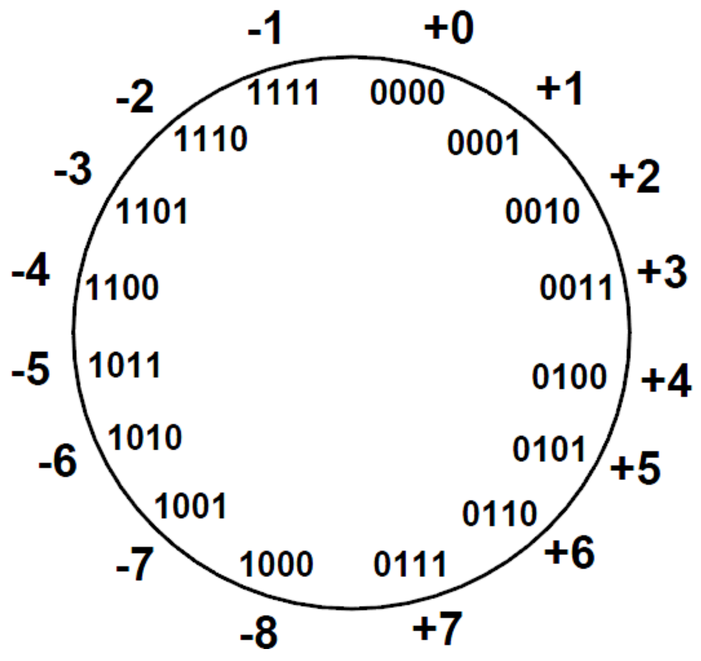
binarno	vrednost
000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

Dvojiški komplement

Samo 1 ničla

V zapisu je negativnih števil za 1 več kot pozitivnih števil (razen, če ničlo štejemo kot pozitivno število)

Primer: 4-bitna števila



Dvojiški komplement

Primer ročne pretvorbe:

$n = 11$

$p = 405 = (256+128+16+4+1) = 00110010101_2$

$-00110010101 \rightarrow 11001101010 + 1 \rightarrow 11001101011$

$k = 11001101011$

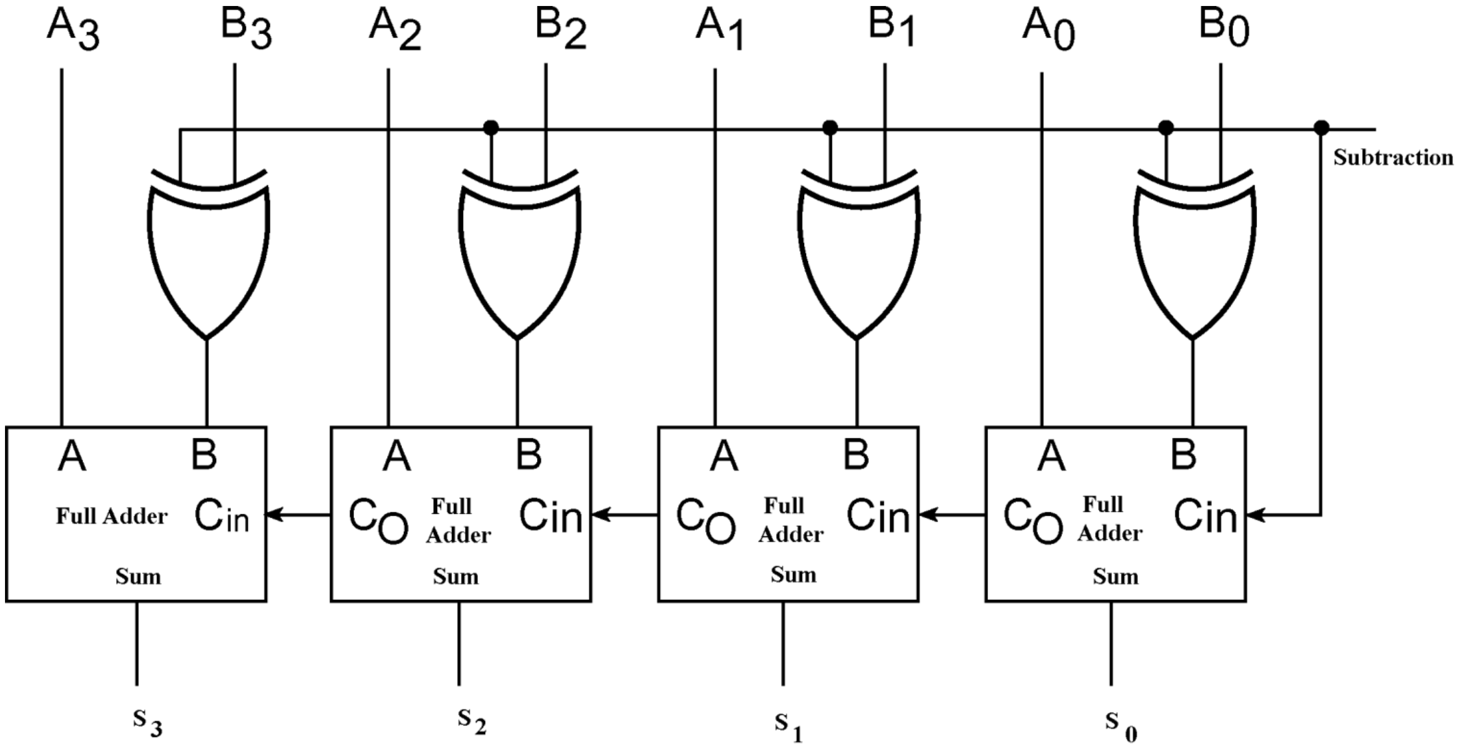
Odštevanje s seštevanjem

Izračunamo dvojiški komplement odštevanca in ga prištejemo zmanjševancu

$D = A - B = A + (-B)$

Seštevalnik/odštevalnik (2')

S=A-B



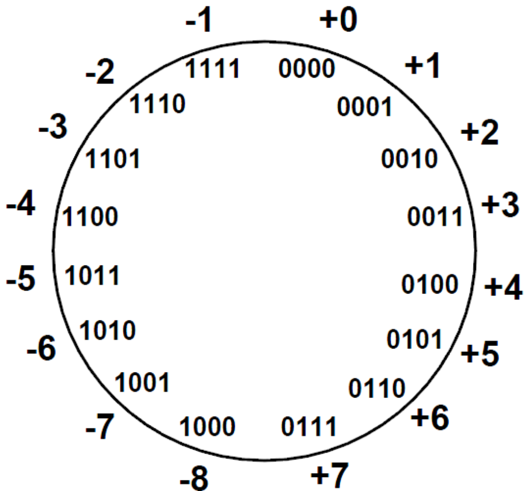
Seštevanje z dvojiškim komplementom

$$\begin{array}{r} (+5) \\ +(+2) \\ \hline (+7) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ +\,0\,0\,1\,0 \\ \hline 0\,1\,1\,1 \end{array} \qquad\qquad \begin{array}{r} (-5) \\ +(+2) \\ \hline (-3) \end{array} \qquad \begin{array}{r} 1\,0\,1\,1 \\ +\,0\,0\,1\,0 \\ \hline 1\,1\,0\,1 \end{array}$$

$$\begin{array}{r} (+5) \\ +\,(-2) \\ \hline (+3) \end{array} \qquad \begin{array}{r} 0\,1\,0\,1 \\ +\,1\,1\,1\,0 \\ \hline 1\,0\,0\,1\,1 \end{array} \qquad\qquad \begin{array}{r} (-5) \\ +\,(-2) \\ \hline (-7) \end{array} \qquad \begin{array}{r} 1\,0\,1\,1 \\ +\,1\,1\,1\,0 \\ \hline 1\,1\,0\,0\,1 \end{array}$$

ignoriramo

ignoriramo

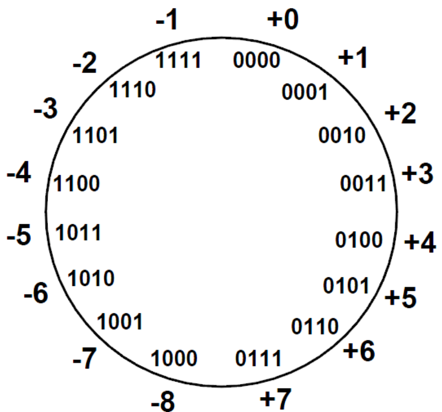


Odštevanje z dvojiškim komplementom

$$
\begin{array}{r}
(+5) \\
-(+2) \\
\hline
(+3)
\end{array}
\qquad
\begin{array}{r}
0\,1\,0\,1 \\
-\,0\,0\,1\,0 \\
\hline
\end{array}
\quad \Rightarrow \quad
\begin{array}{r}
0\,1\,0\,1 \\
+\,1\,1\,1\,0 \\
\hline
1\,0\,0\,1\,1
\end{array}
$$

ignoriramo

$$
\begin{array}{r}
(-5) \\
-(-2) \\
\hline
(-3)
\end{array}
\qquad
\begin{array}{r}
1\,0\,1\,1 \\
-\,1\,1\,1\,0 \\
\hline
\end{array}
\quad \Rightarrow \quad
\begin{array}{r}
1\,0\,1\,1 \\
+\,0\,0\,1\,0 \\
\hline
1\,1\,0\,1
\end{array}
$$



PreliV (*OVerflow*)

n-bitni zapis: števila v razponu

$$\left[\frac{-2^n}{2}, \frac{2^n}{2} - 1\right] = [-2^{n-1}, 2^{n-1} - 1]$$

primer:

- n =3: [-4,3]
- n = 16: [-32768,32767]: če 32767 prištejemo 1, dobimo -32768 ← narobe

Če rezultata ne moremo zapisati v območju predstavitve števila, govorimo o preliVu (**arithmetic oVerflow**)

Potrebujemo logiko za zaznavanje preliva!

Primeri preliva

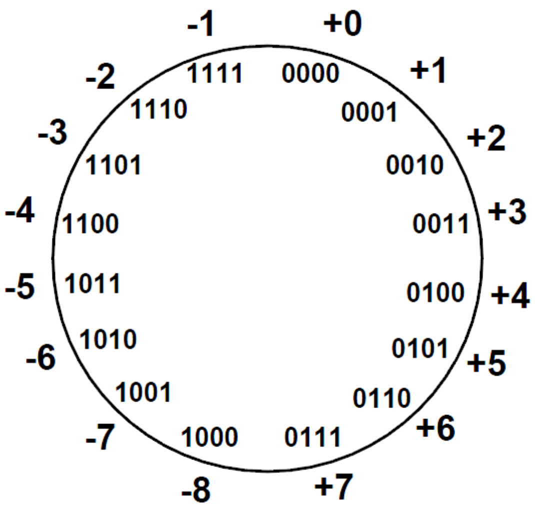
$$
\begin{array}{rcccc}
 & x_3 & x_2 & x_1 & x_0 \\
+ & y_3 & y_2 & y_1 & y_0 \\
\hline
c_4 & c_3 & c_2 & c_1 & \\
\hline
 & s_3 & s_2 & s_1 & s_0
\end{array}
$$

$$
\begin{array}{r}
(+7) \\
+(+2) \\
\hline
(+9)
\end{array}
\qquad
\begin{array}{rcccc}
 & 0 & 1 & 1 & 1 \\
+ & 0 & 0 & 1 & 0 \\
\hline
 & 1 & 0 & 0 & 1
\end{array}
\qquad
\begin{array}{r}
(-7) \\
+(+2) \\
\hline
(-5)
\end{array}
\qquad
\begin{array}{rcccc}
 & 1 & 0 & 0 & 1 \\
+ & 0 & 0 & 1 & 0 \\
\hline
 & 1 & 0 & 1 & 1
\end{array}
$$

$c_4 = 0$, $c_3 = 1$ $c_4 = 0$, $c_3 = 0$

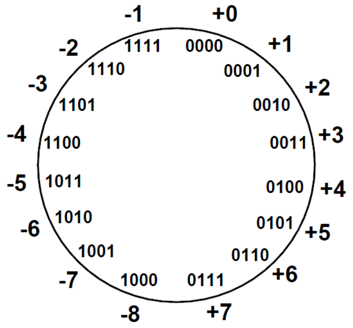
$$
\begin{array}{r}
(+7) \\
+(-2) \\
\hline
(+5)
\end{array}
\qquad
\begin{array}{rccccc}
 & & 0 & 1 & 1 & 1 \\
+ & & 1 & 1 & 1 & 0 \\
\hline
 & 1 & 0 & 1 & 0 & 1
\end{array}
\qquad
\begin{array}{r}
(-7) \\
+(-2) \\
\hline
(-9)
\end{array}
\qquad
\begin{array}{rccccc}
 & & 1 & 0 & 0 & 1 \\
+ & & 1 & 1 & 1 & 0 \\
\hline
 & 1 & 0 & 1 & 1 & 1
\end{array}
$$

$c_4 = 1$, $c_3 = 1$ $c_4 = 1$, $c_3 = 0$



Če imajo števila različne predznake, do preliva ne more priti!

Zaznavanje preliva



Seštevanje dveh pozitivnih števil nam vrne negativno število (MSB števil je 0, MSB rezultata je 1).

$$\overline{a}_{n-1}\overline{b}_{n-1}s_{n-1}$$

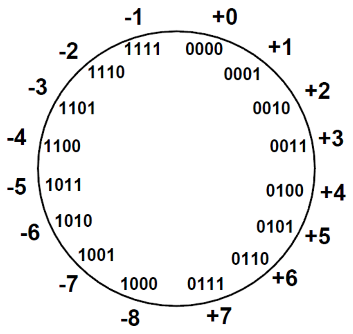
Seštevanje dveh negativnih števil na vrne pozitivno število (MSB števil je 1, MSB rezultata je 0).

$$a_{n-1}b_{n-1}\overline{s}_{n-1}$$

Zaznavanje preliva:

$$V = \overline{a}_{n-1}\overline{b}_{n-1}s_{n-1} \vee a_{n-1}b_{n-1}\overline{s}_{n-1}$$

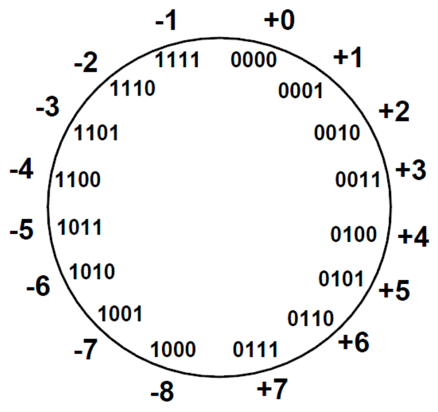
Zaznavanje preliva



Preliv lahko izrazimo s prenosoma pri seštevanju najbolj pomembnih bitov

a_{n-1}	b_{n-1}	c_{n-1}	c_n	s_{n-1}	V	$c_n \nabla c_{n-1}$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	1	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	1	1	0	0	0
1	1	0	1	0	1	1
1	1	1	1	1	0	0

$$V = \overline{a}_{n-1}\overline{b}_{n-1}s_{n-1} \vee a_{n-1}b_{n-1}\overline{s}_{n-1} = c_n \nabla c_{n-1}$$



Izogibanje prelivu

- Razteg predznaka (angl. *sign extension*).
- Podvojimo MSB, ki v dvojiškem komplementu določa predznak.
- Pri seštevanju dveh n-bitnih števil, tako dobimo (n+1)-bitni rezultat – do preliva ne more priti!
- Prenos na MSB lahko ignoriramo.
- Primer:
    - brez razširitve: – 4 – 6 = $1100_2 + 1010_2 = 0110_2 = 6$
    - z razširitvijo: – 4 – 6 = $(1)1100_2 + (1)1010_2 = \cancel{1}10110_2$

$10110_2 \rightarrow -(01001_2 + 1) = -01010_2 = -10$

$$k = (2^n - 1) - p + 1 = 2^n - p$$

Aritmetika s predznačenimi/nepredznačenimi števili

Pri zapisu v dvojiškem komplementu lahko uporabljamo enake gradnike

Različna interpretacija rezultata

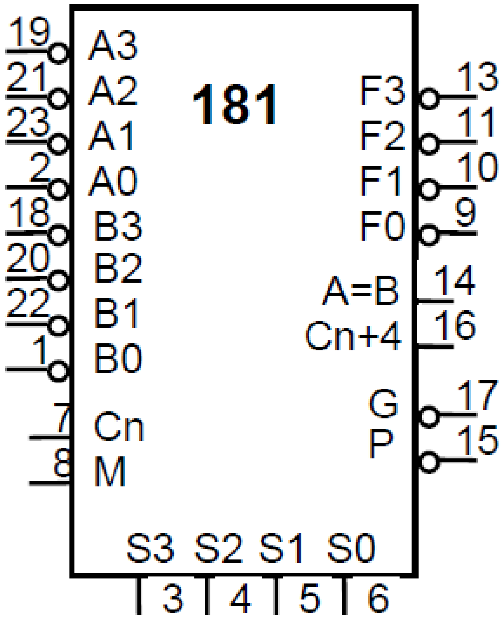
Carry/borrow iz MSB pri nepredznačenih številih – presegli smo obseg predstavitve

Carry/borrow iz MSB pri predznačenih številih – lahko ignoriramo; potrebujemo logiko za detekcijo preliVa.

Aritmetično logična enota (ALE)

- Arithmetic Logic Unit (ALU)
- Aritmetične operacije: npr. seštevanje in odštevanje
- Logične operacije: npr. logične funkcije in pomikanje podatkov

74181



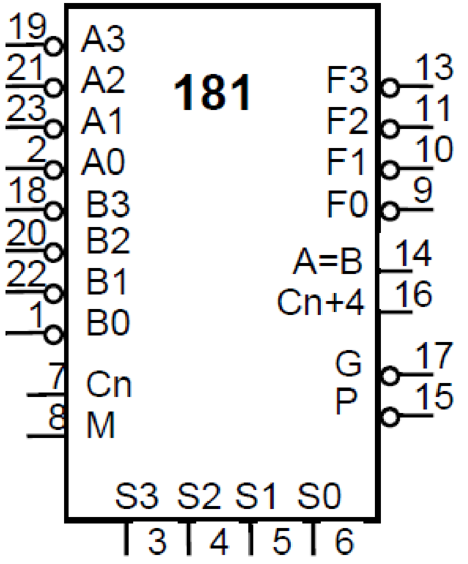
- Kombinatorno vezje za izvedbo aritmetično-logičnih operacij.

- Priključki:
- A,B: podatkovna vhoda
- F: podatkovni izhod
- C_{n+4}: prenos na najvišjem mestu
- A=B: enakost vhodov
- M: kontrolni vhod, ki določa tip operacije
  - M=0: aritmetične
  - M=1: logične
- S: kontrolni vhodi, ki določajo konkretno operacijo

- Izvedba 48 različnih operacij.

74181

$S_3S_2S_1S_0$	M=1	C=0	C=1
		M=0	
0 0 0 0	F = A′	F = A minus 1	F = A
0 0 0 1	F = (AB)′	F = AB minus 1	F = AB
0 0 1 0	F = A′∨B	F = AB′ minus 1	F = AB′
0 0 1 1	F = 1	F = minus 1 (2′K)	F = 0
0 1 0 0	F=(A∨B)′	F=A plus (A∨B′)	F = A plus (A ∨ B′) plus 1
0 1 0 1	F = B′	F=AB plus (A∨B′)	F = AB plus (A ∨ B′) plus 1
0 1 1 0	F=A≡B	F=A minus B minus 1	F = A minus B
0 1 1 1	F=A∨B′	F = A ∨ B′	F = (A ∨ B′) plus 1
1 0 0 0	F = A′B	F=A plus (A∨B)	F = A plus (A ∨ B) plus 1
1 0 0 1	F=A⊕B	F = A plus B	F = A plus B plus 1
1 0 1 0	F = B	F=AB′ plus (A∨B)	F = AB plus (A ∨ B) plus 1
1 0 1 1	F = A∨B	F = A ∨ B	F = (A ∨ B) plus 1
1 1 0 0	F = 0	F = A plus A	F = A plus A plus 1
1 1 0 1	F = AB′	F = AB plus A	F = AB plus A plus 1
1 1 1 0	F = AB	F = AB′ plus A	F = AB′ plus A plus
1 1 1 1	F = A	F = A	F = A plus 1



$F_i=f(A_i,B_i)$ i=0..3

Primer: M=1, S=1110

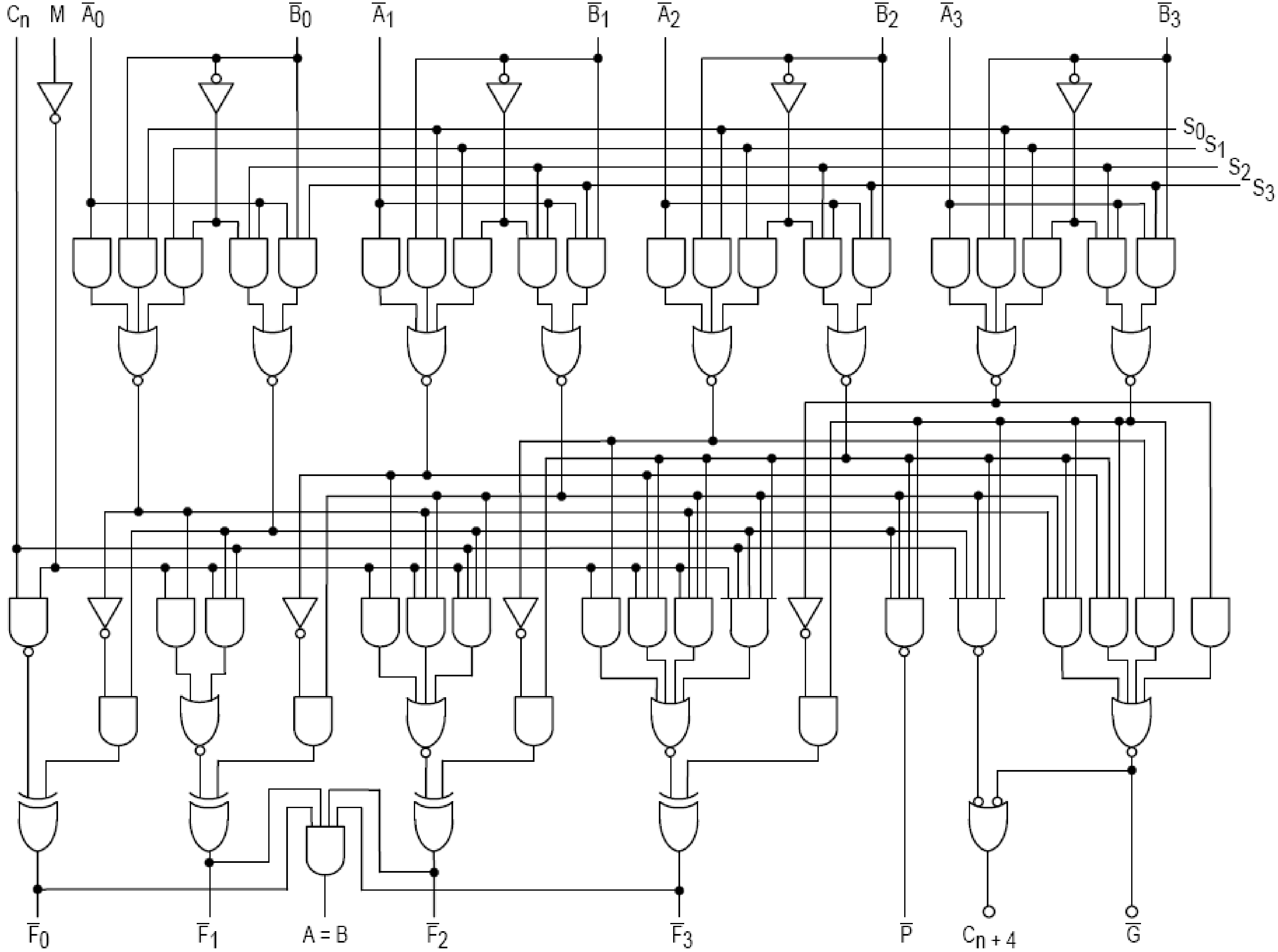
$F_0=A_0$ AND B_0

$F_1=A_1$ AND B_1

$F_2=A_2$ AND B_2

$F_3=A_3$ AND B_3

74181



C_n
M
$\overline{A}_0$
$\overline{B}_0$
$\overline{A}_1$
$\overline{B}_1$
$\overline{A}_2$
$\overline{B}_2$
$\overline{A}_3$
$\overline{B}_3$
S_0
S_1
S_2
S_3
$\overline{F}_0$
$\overline{F}_1$
A = B
$\overline{F}_2$
$\overline{F}_3$
$\overline{P}$
C_{n+4}
$\overline{G}$