ФИКТ

Дигитална логика и системи

5

Аритметички операции и дигитални кола. Конвертори на код. Кодери, Декодери. Мултиплексери - Демултиплексери

Доц. д-р Никола Рендевски

nikola.rendevski@fikt.edu.mk

летен семестар 2017/2018

Аритметички операции

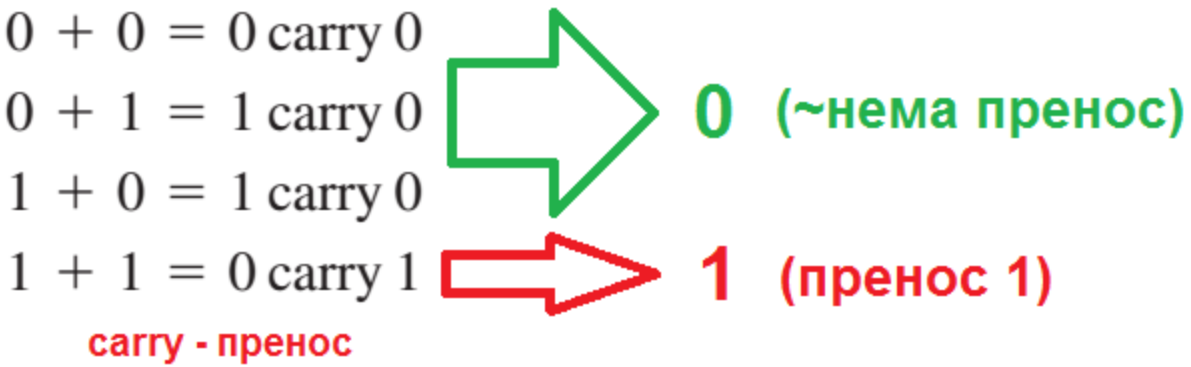
- Една од најважните функции на дигиталните системи и компјутери е извршувањето на аритметички операции
- Дигиталниот систем „дигиталната архитектурата“ за извршување (execution) на аритметички операции се базира на градбени блокови кои припаѓаат на основните логички порти
- Изведување на соодветна аритметичка операција се реализира со чекор-по-чекор процедура се додека не се добие посакуваниот резултат
- Секоја аритметичка операција може да биде реализирана во комбинаторна логика или со користење на MSI чипови (Medium Scale Integration) – Средно -интегрирани чипови од основни логички порти

Бинарна аритметика

- Процедури за четирите основни операции
  - собирање, одземање, множење и делење

Собирање

- Слична процедура како и кај декадниот систем, освен што бинарната сума е од 1 и 0.
- Кога бинарната сума надминува 1, се пренесува 1 (**carry**) во следната повеќе-сигнификантна колона (позиција)
- Четири комбинации за собирање

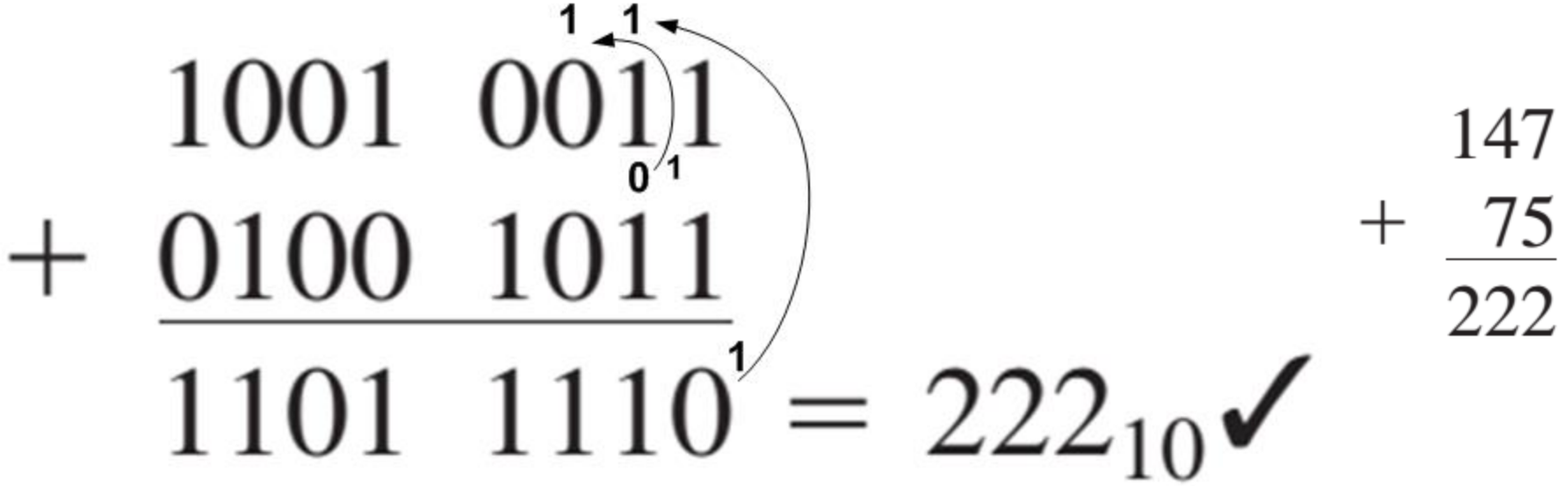


Собирање

Carry Out (пренос)

A_0	B_0	Σ_0	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$A_0 + B_0 = \Sigma_0 + C_{out}$$



Одземање

$0 - 0 = 0$ borrow 0

$0 - 1 = 1$ borrow 1

$1 - 0 = 1$ borrow 0

$1 - 1 = 0$ borrow 0

borrow - позајми

A_0	B_0	R_0	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

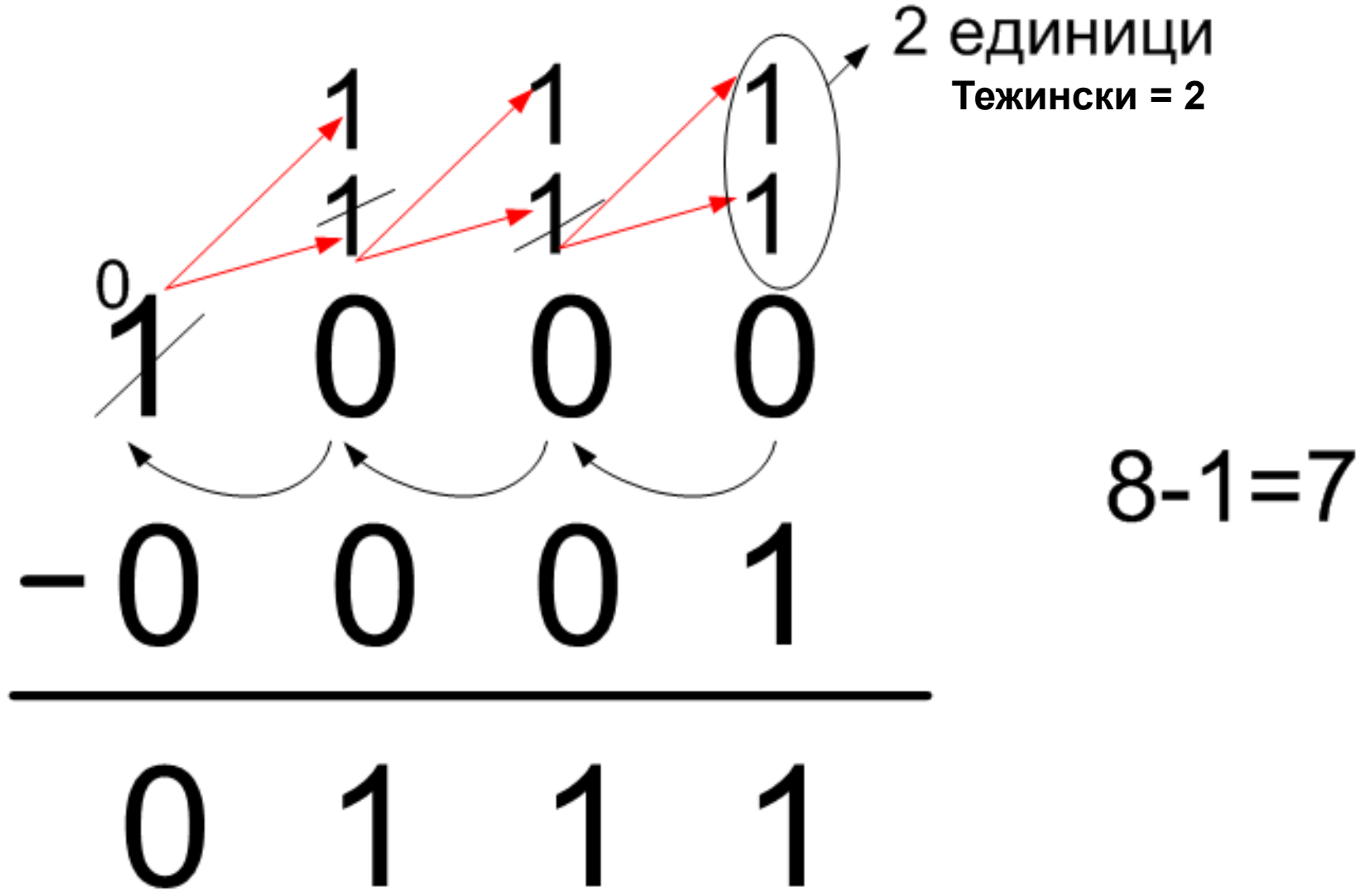
$$A_0 - B_0 = R_0 + B_{out}$$

Позајми 1 од A_1 → **A_1 станува 0, A_0 станува 10 или 2_{10}**

$$\begin{array}{cc} A_1 & A_0 \\ B_1 & B_0 \\ \hline R_1 & R_0 \end{array} \quad \begin{array}{c} {}^{0}\not{1} \; {}^{2}\not{0} \\ -0 \quad 1 \\ \hline 0 \quad 1 \end{array}$$

1
1

Одземање – Начин 2 (Пример - тежински)



Одземање – Примери

(a)	27	0001 1011
	− 10	− 0000 1010
	17	0001 0001 = 17_{10}✓
(b)	9	0000 1001
	− 4	− 0000 0100
	5	0000 0101 = 5_{10}✓
(c)	172	1010 1100
	− 42	− 0010 1010
	130	1000 0010 = 130_{10}✓
(d)	154	1001 1010
	− 54	− 0011 0110
	100	0110 0100 = 100_{10}✓
(e)	192	1100 0000
	− 3	− 0000 0011
	189	1011 1101 = 189_{10}✓

Множење

- Множењето на бинарни броеви можеме да го реализираме на ист начин како множењето на повеќецифрени декадни броеви
- Имајќи во предвид дека кај бинарниот систем имаме само единици и нули, тогаш меѓурезултатот секогаш ќе биде множеникот или пак нула (низа од нули)
- Секој меѓупроизвод која учествува во крајната сума е множеникот шифтириан во лево во насока од LSB кон MSB
- 1x0=0, 0x1=0, 0x0=0,1x1=1

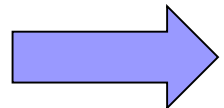
Множење на бинарни броеви

$$
\begin{array}{r}
23 \\
\times\ 9 \\
\hline
207
\end{array}
\qquad
\begin{array}{r}
0001\ 0111 \\
\times\ 0000\ 1001 \\
\hline
0001\ 0111 \\
00000\ 000 \\
000000\ 00 \\
0001011\ 1 \\
\hline
0001100\ 1111 = 1100\ 1111 = 207_{10}\checkmark
\end{array}
$$

$$
\begin{array}{r}
15 \\
\times\ 15 \\
\hline
75 \\
+\ 15 \\
\hline
225
\end{array}
\qquad
\begin{array}{r}
0000\ 1111 \\
\times\ 0000\ 1111 \\
\hline
0000\ 1111 \\
00001\ 111 \\
000011\ 11 \\
+\ 0000111\ 1 \\
\hline
0001110\ 0001 = 1110\ 0001 = 225_{10}\checkmark
\end{array}
$$

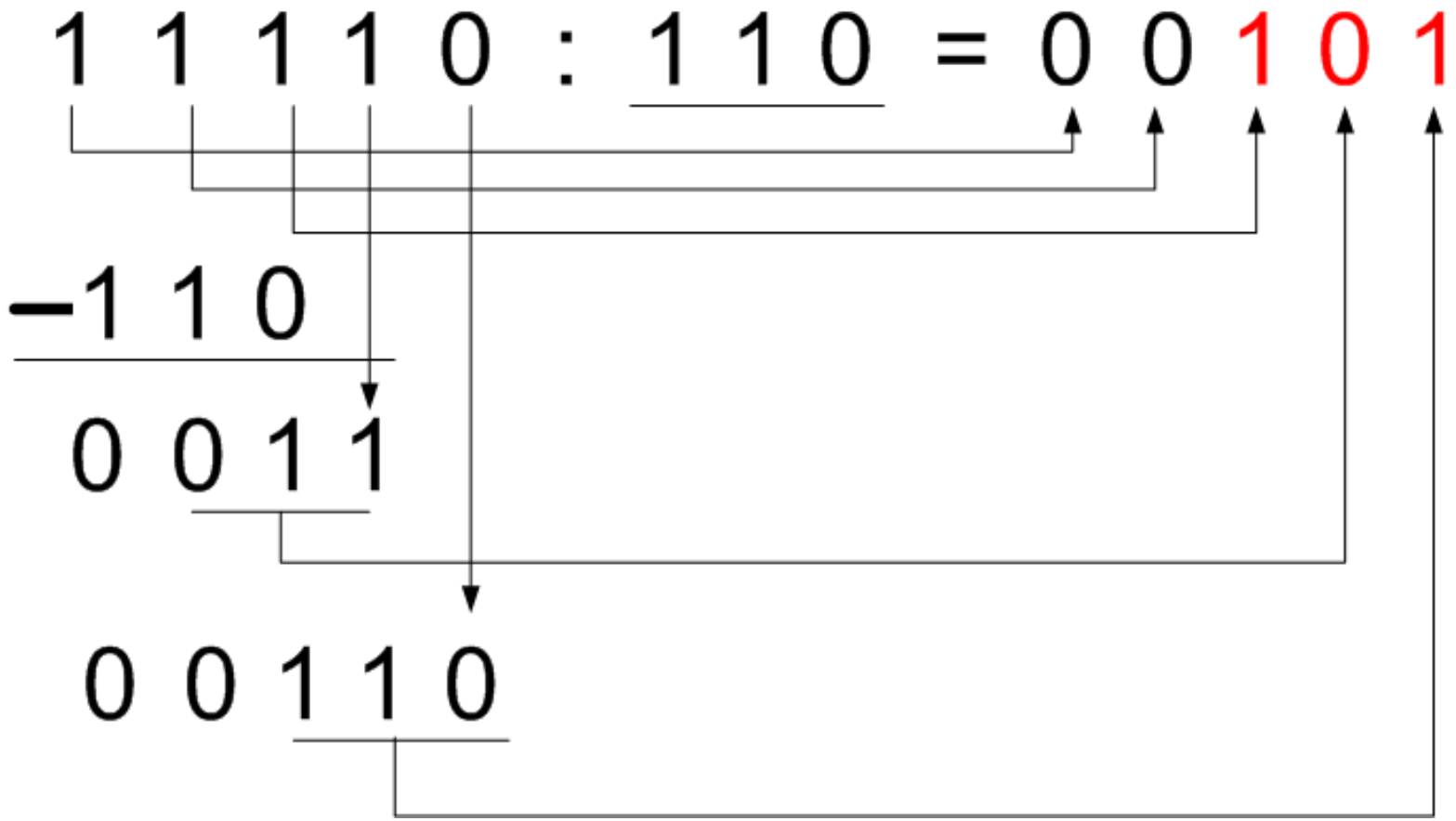
Делење на бинарни броеви

- Делењето на два бинарни броеви може да се реализира на сличен начин како кај декадниот
- Се тргнува од првата цифра на деленикот и се дели со делителот. Доколку бројот на до тековната цифра е помал од делителот се пишува 0, доколку може да се подели се пишува 1, а потоа се одзема од бројот до тековната цифра. Потоа се спушта друга цифра и постаптката се повторува до крај.
- Ќе го презентираме преку еден пример



Делење на бинарни броеви

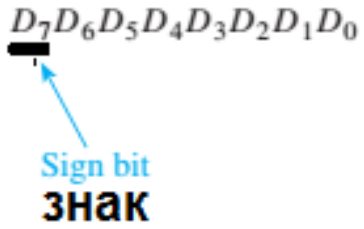
- Пример 32:6=5
- 32_{10}=11110, 6_{10}=110



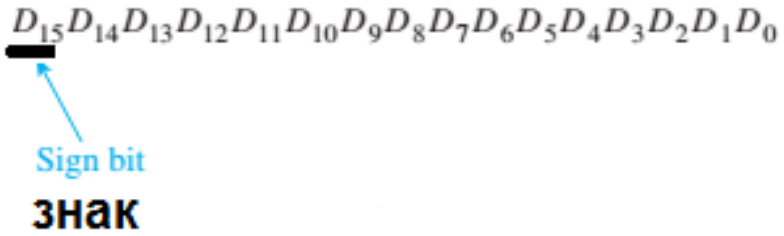
Двоен комплемент

- Најчеста форма на репрезентација на бинарните броеви при реализација на аритметичките операции е т.н двоен комплемент (two's complement method)
- На овој начин и позитивните и негативните броеви можат да се претстават во ист формат, а операцијата одземање е значително попроста
- Најчесто дигиталните/компјутерските системи се базирани на 8-, 16-, 32- или 64-битна презентација на бинарните секвенци (податоци, адреси, итн)
- Со цел да се претстацат позитивни и негативни вредности во формат на двоен комплемент, MSB битот се користи за знак (MSB – Sign Bit)
- 0 – позитивен број
- 1 – негативен број

Двоен комплемент



1 - НЕГАТИВЕН
0 - Позитивен



- При репрезентацијата со двоен комплемент, рангот на броевите (состојбите) кои можат да се репрезентираат со битова секвенца од N бита е 2^{N-1}
- На пример: со 8-бита можат да се претстават
  - Позитивните броеви од 00000000 – 01111111
  - Негативните броеви од 11111111 – 10000000
- Максималниот позитивен број е $2^{N-1}-1$
- Максималниот негативен број е $-(2^{N-1})$

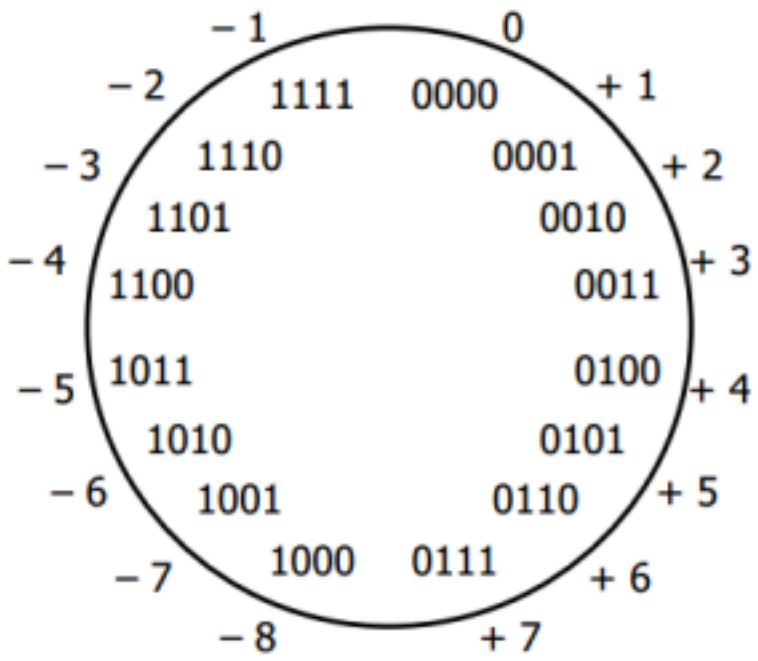
Двоен комплемент

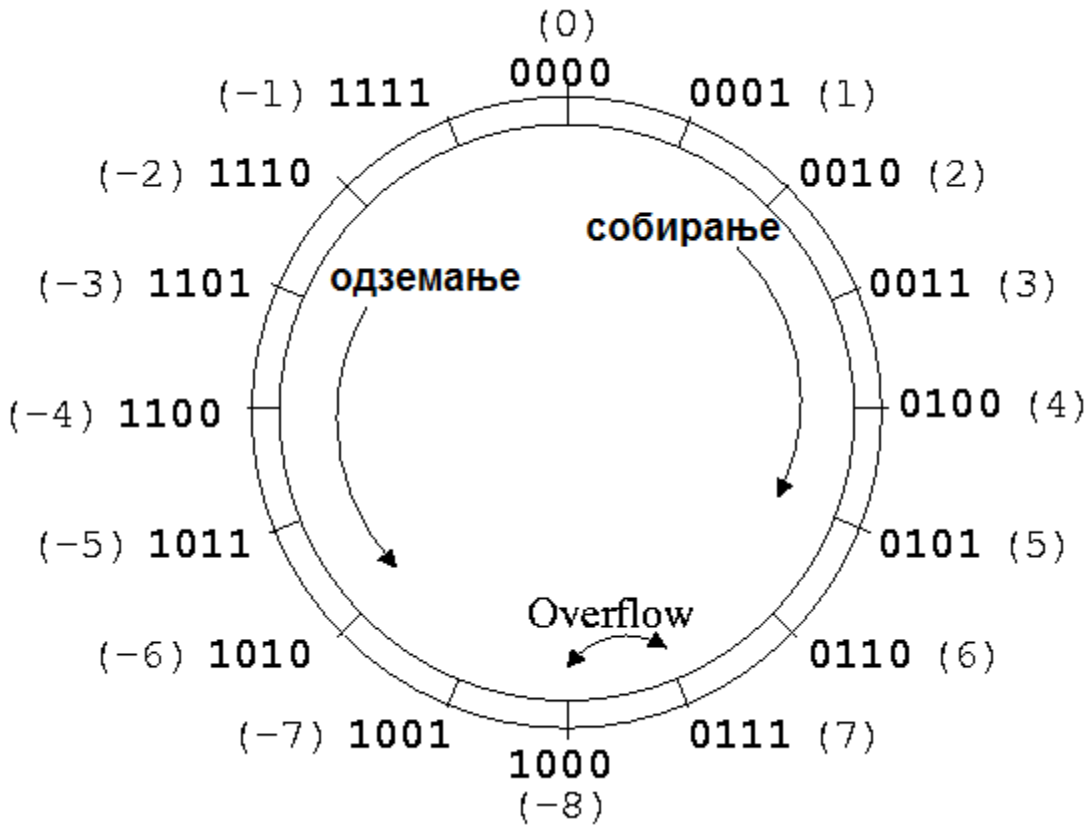
- Доколку бројот е позитивен тогаш неговата презентација во двоен комплемент е чисто неговиот бинарен еквивалент, Битот за знак е 0
- Доколку бројот е негативен тогаш
  1. Комплементирај го секој бит во бинарниот еквивалент, бит по бит) – прв комплемент (one's complement), потоа
  2. Додај 1 на бинарната секвенца од првиот комплемент
  - Битот за знак е 1 (MSB)
- Одземањето може да се реализира како собирање со двојниот комплемент

Двоен комплемент

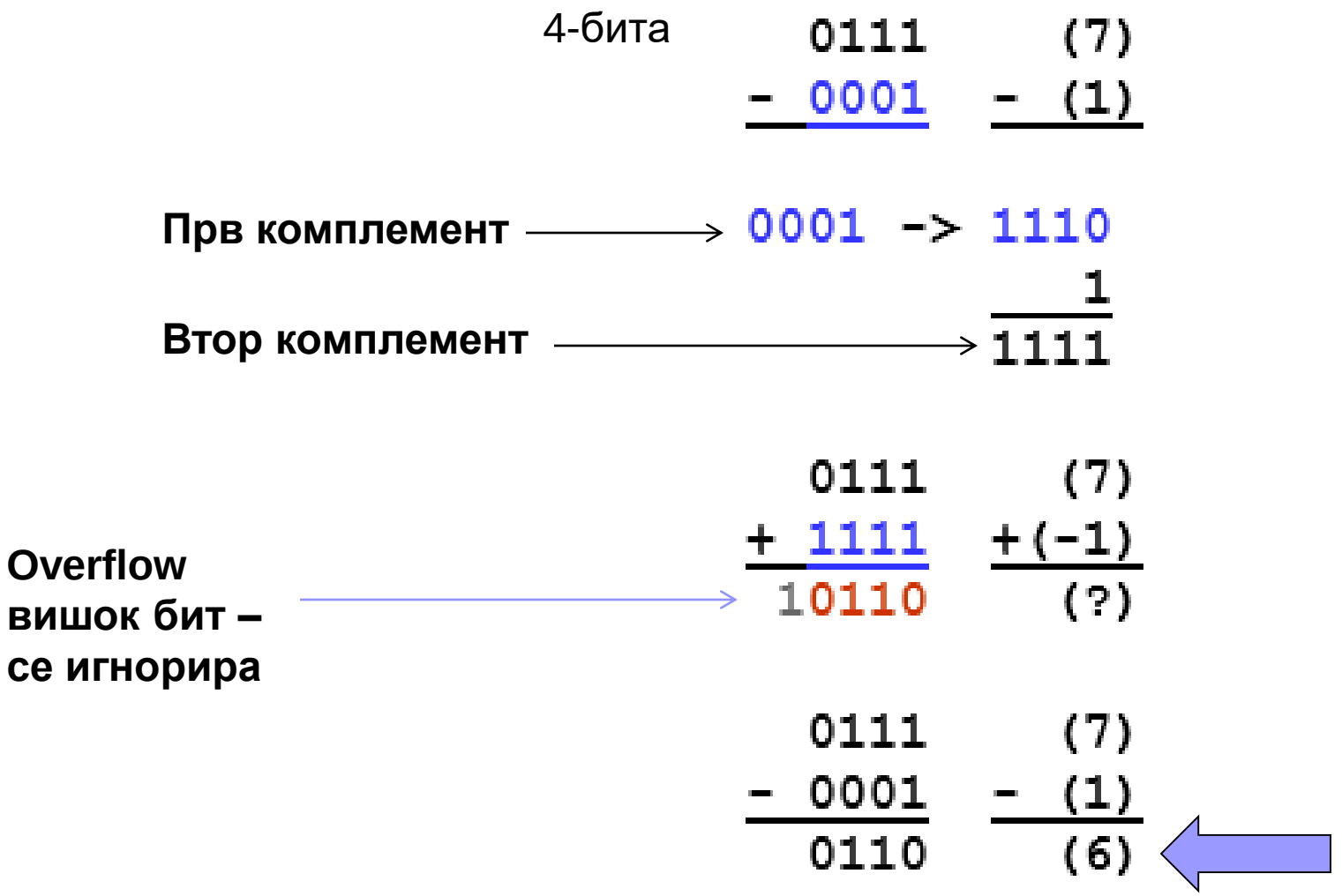
- Доколку бројот е позитивен тогаш неговата презентација во двоен комплемент е чисто неговиот бинарен еквивалент, Битот за знак е 0
- Доколку бројот е негативен тогаш
  1. Комплементирај го секој бит во бинарниот еквивалент, бит по бит) – прв комплемент (one's complement), потоа
  2. Додај 1 на бинарната секвенца од првиот комплемент
  - Битот за знак е 1 (MSB)
- Одземањето може да се реализира како собирање со двојниот комплемент

Двоен комплемент





Пример (overflow) – 7_{10}-1_{10}



Примери

$$
\begin{aligned}
+18 &= 0001\ 0010 \\
-7 &= \underline{1111\ 1001} \\
\text{Sum} &= 0000\ 1011 = 11_{10}
\end{aligned}
$$

$$
\begin{aligned}
+21 &= 0001\ 0101 \\
-13 &= \underline{1111\ 0011} \\
\text{Sum} &= 0000\ 1000 = 8_{10}
\end{aligned}
$$

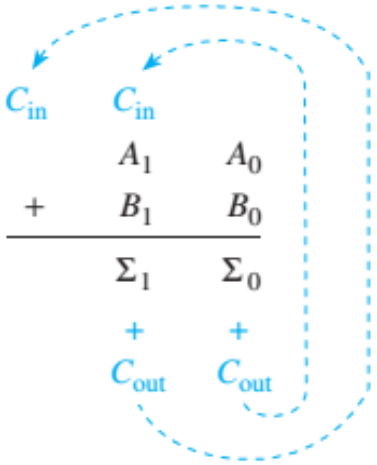
$$
\begin{aligned}
+59 &= 0011\ 1011 \\
-96 &= \underline{1010\ 0000} \\
\text{Sum} &= 1101\ 1011 = -37_{10}
\end{aligned}
$$

$$
\begin{aligned}
+118 &= 0111\ 0110 \\
-54 &= \underline{1100\ 1010} \\
\text{Sum} &= 0100\ 0000 = 64_{10}
\end{aligned}
$$

Дигитални кола за аритметички операции

- Полусобирач (Half Adder)
  - Два влеза и два излеза
    - Влезови (Вредностите на бинарните варијабли)
    - Излези (Збир и пренос)
- **Целосен собирач (Full Adder)**
  - **Три влезови и два излези**
    - Два влеза за влезни операнди (двата значајни битови)
    - Третиот влез преносот од помалку значајната позиција (Carry in)
    - Два излези се потребни имајќи во предвид дека сумата на три бинарни цифри може да се претстави со два бита (две линии)

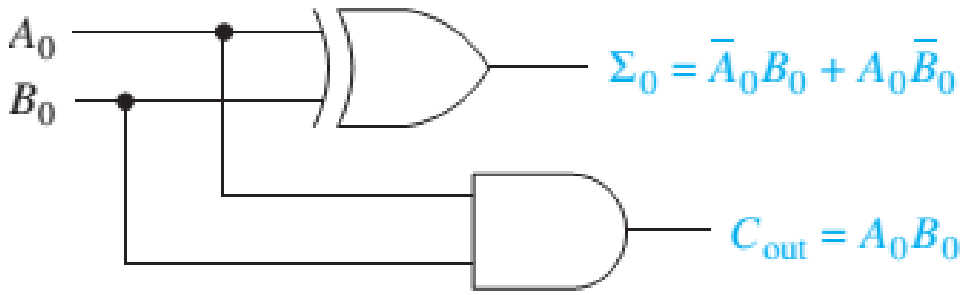
Полусобирач

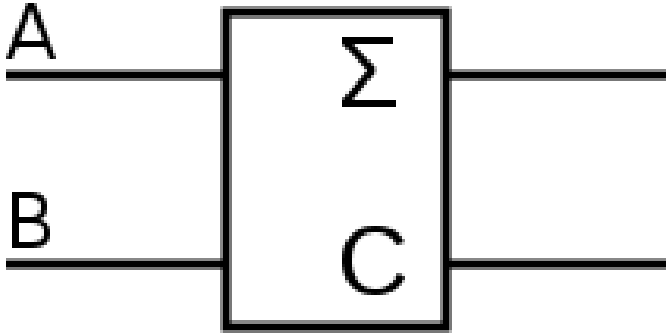


(a)

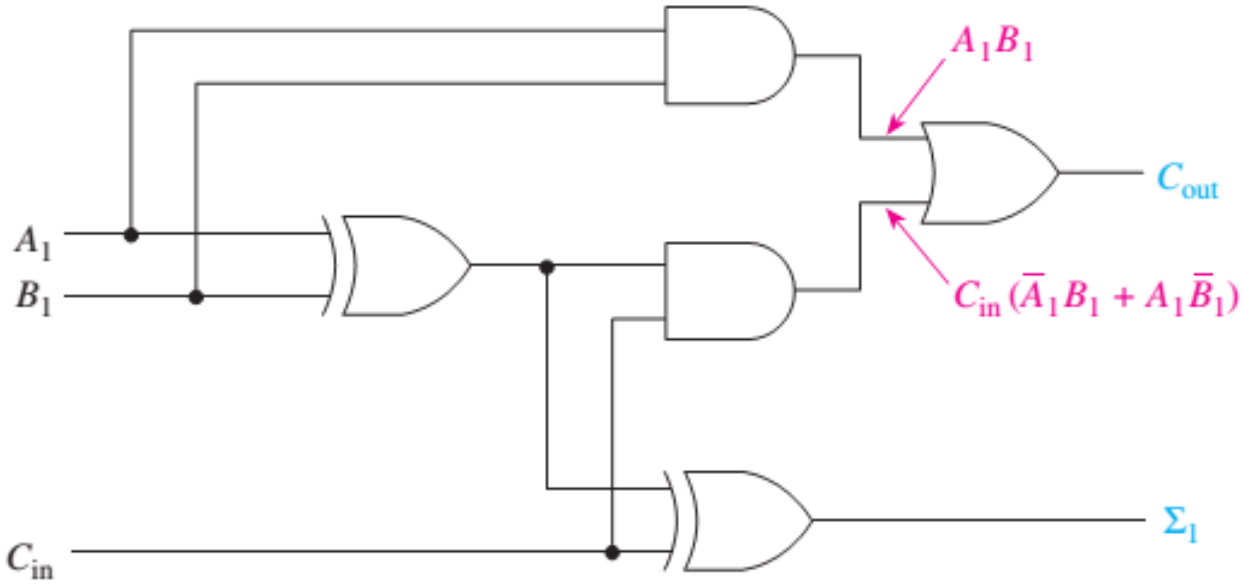
2 Влеза		2 Излези	
A_0	B_0	Σ_0	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

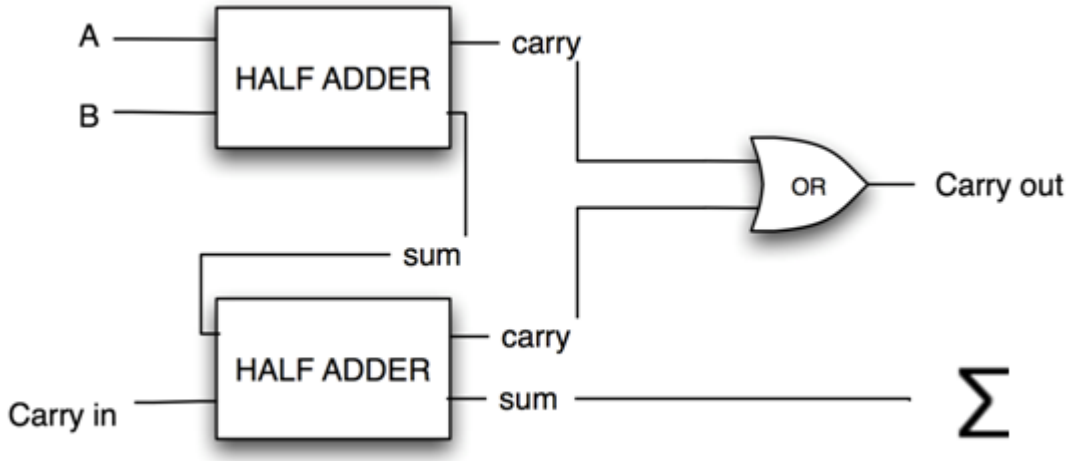
(b)





Целосен собирач

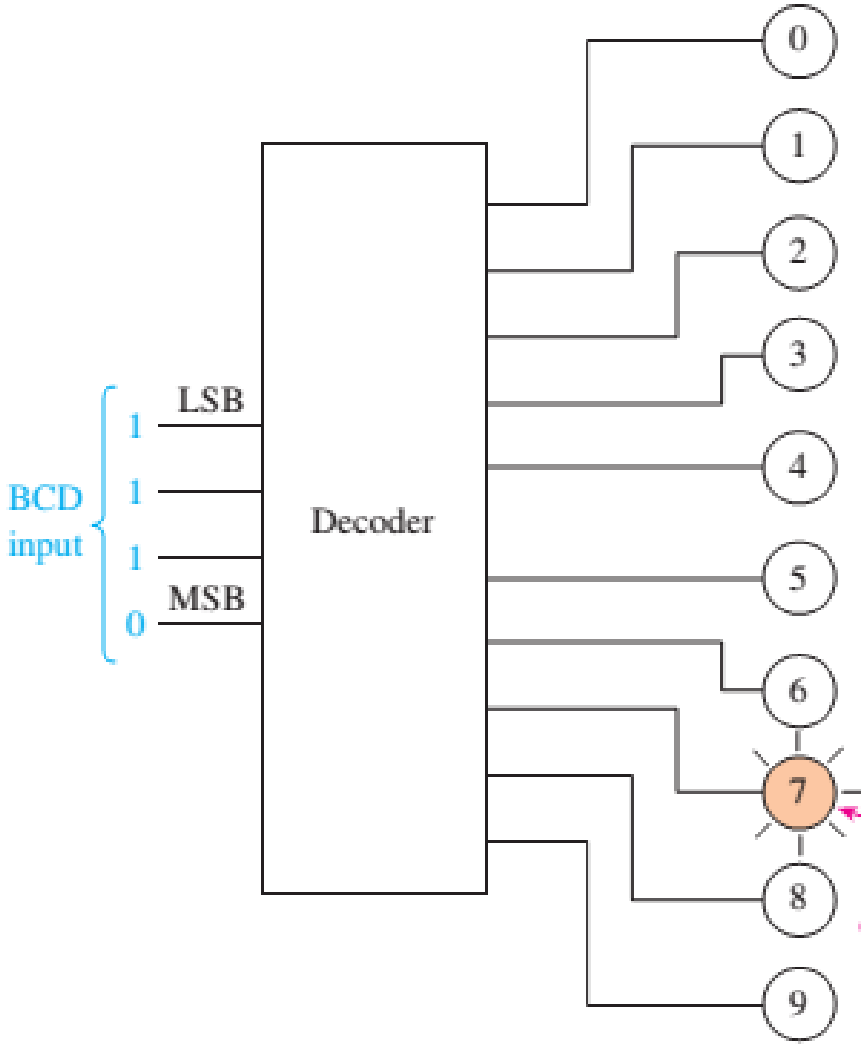




Декодер

- Во дигиталните системи, декодер е комбинаторно логичко коло (матрица) кој конвертира бинарна информација од n-кодирани влезови во максимум 2^n различни излези. Декодерот се користи за различни примени, како реализација на 7-сегментни дисплеи, декодирање на мемориски адреси итн.
- Постојат неколку типови на бинарни декодери, но во сите случаи се работи за дигитално коло кое ја конвертира секоја (единствена) влезна комбинација во единствена излезна комбинација
- Пример 1-of-n декодерот претставува логичка коло кое има множество на влезови (inputs) кои репрезентираат еден бинарен број, а го активира само излезот кој кореспондира со влезниот број
- Со други зборови, 1-of-n декодерот активира само еден излез кој одговара на вредноста на влезната бинарна секвенца, додека останатите излези се неактивни (0)
- Сингуларен активен излез! (за 1-of-n)
- Ќе видиме понатаму дека демултиплексерот е 1-of-n декодер

Декодер (1-од-n) – илустративен пример



BCD – во децимален декодер

Декодер (1-од-n)

- Постојат две можности за логичка изведба за 1-од-n декодер
  - Active-Low-Level (Активно ниско ниво)
  - Active-High-Level (Активно високо ниво)
- Избор на дизајнерот или во зависност од логичките порти/логиката на изведба на дигиталниот дизајн
- Вистинитосните табели на двете изведби се комплементирани една во однос на друга
- При дизајнот на декодерот во дигитална комбинаторна логика треба да се реализира таблица на вистинитост за секој излез во зависност од комбинацијата на влезот
- Најчесто (Често) достапните IC дигитални кола се Active-Low-Level

3-битен бинарен во октален

Active-HIGH Outputs активно-високо

Input			Output							
2^2	2^1	2^0	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

селектираниот излез - високо ниво

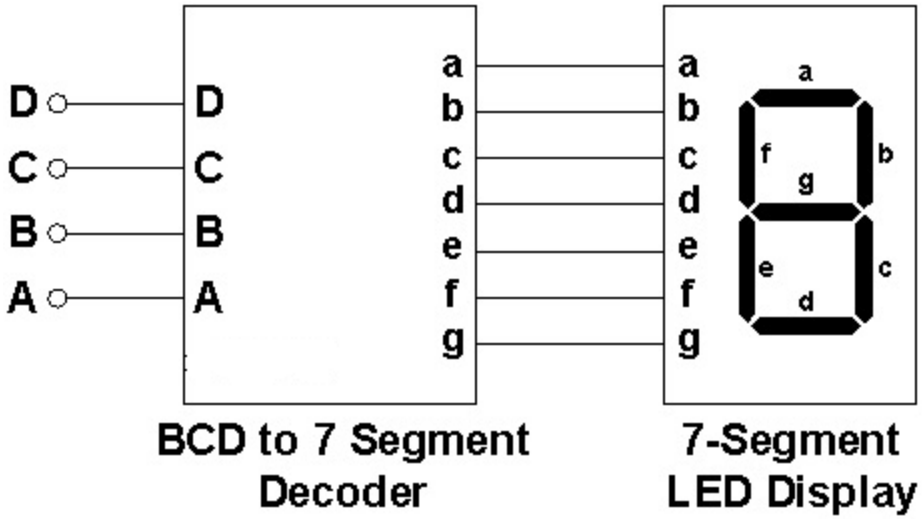
Active-LOW Outputs активно-ниско

Input			Output							
2^2	2^1	2^0	0	1	2	3	4	5	6	7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

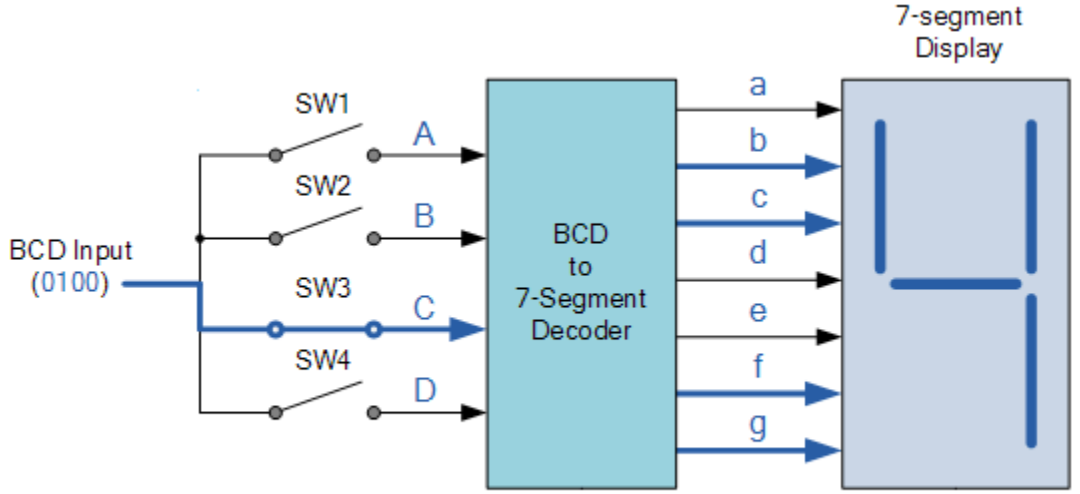
селектираниот излез - ниско ниво

BCD во 7-сегментен дисплеј декодер

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

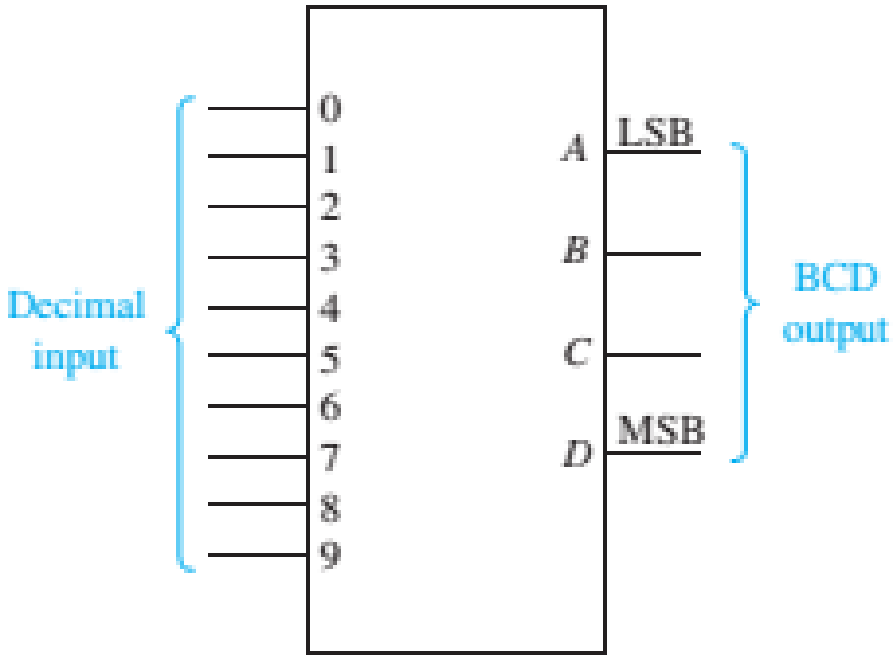


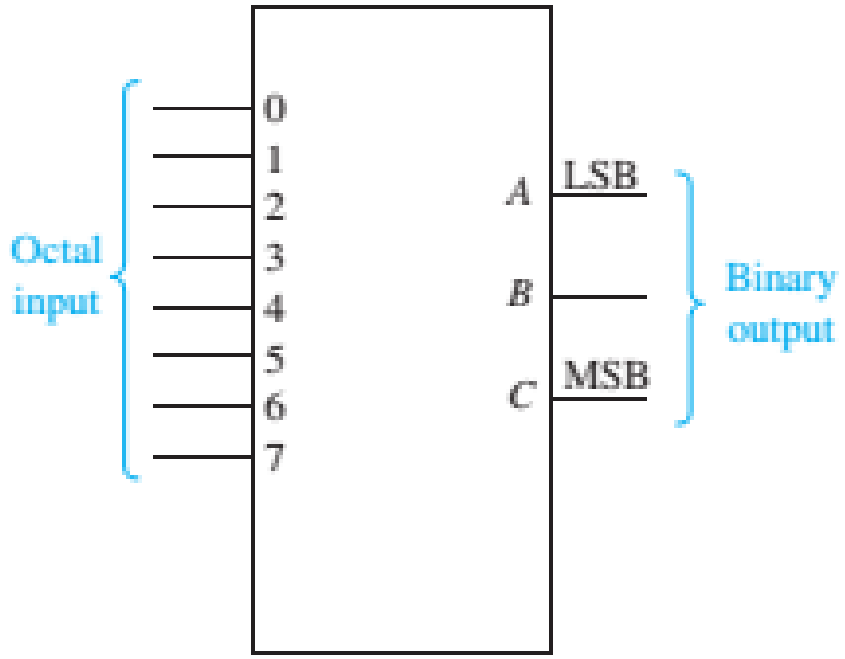
Пример 4_{BCD}



Кодер (Coder, Encoding)

- Кодирањето е обратен процес во однос на декодирањето
- Кодерот (англ. Encoder) е дигитално коло кое генерира кодиран излез од влезот.
- Пример: Декаден во бинарен, Декаден во BCD итн

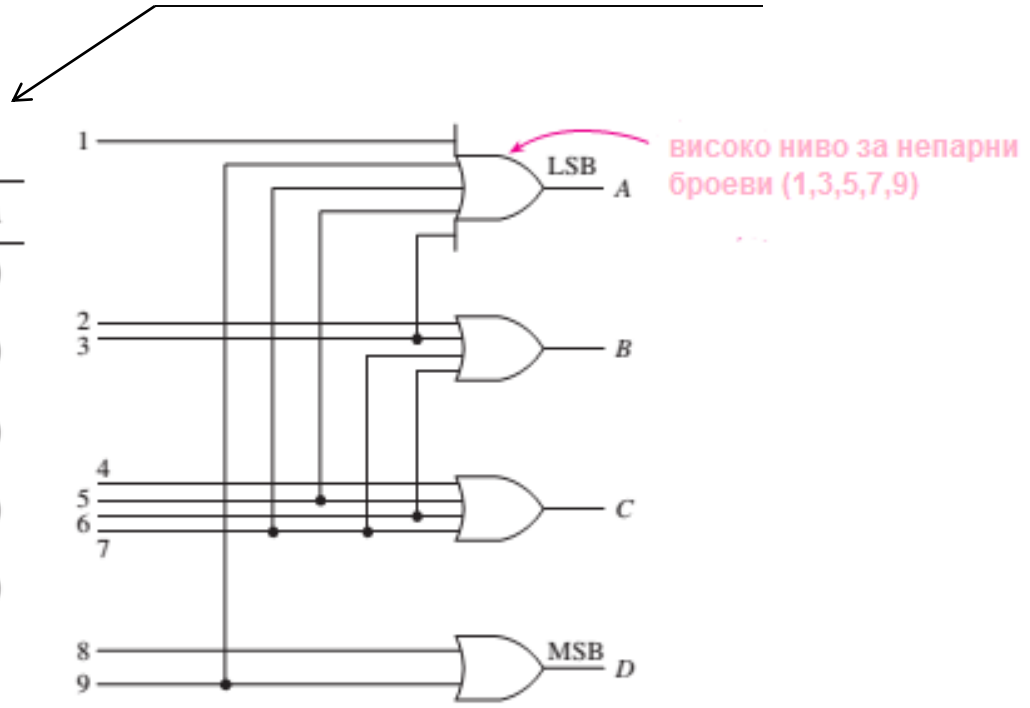




Кодер (Coder, Encoding)

- Кодирањето е обратен процес во однос на декодирањето
- Кодерот е дигитално коло кое генерира кодиран излез од влезот
- Пример: Декаден во BCD итн.

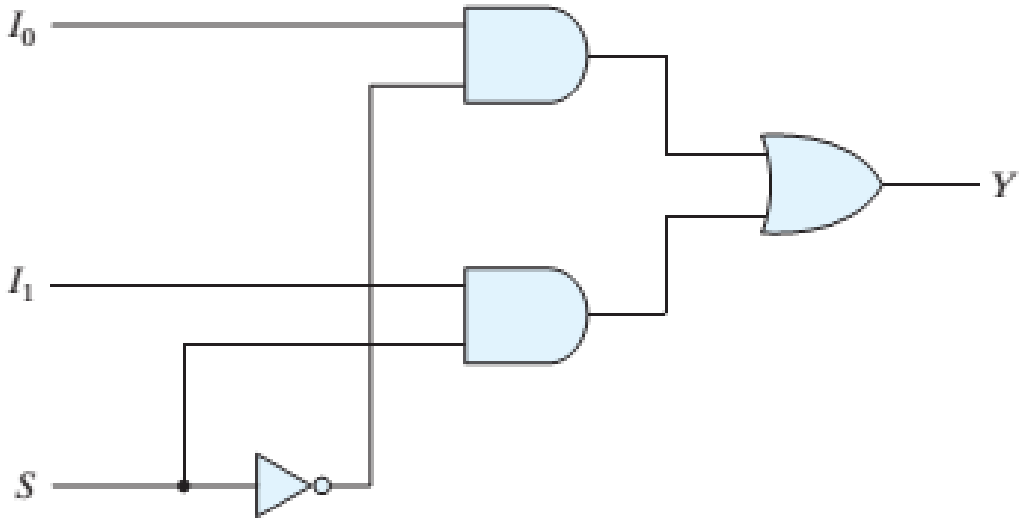
Decimal Input	BCD Output			
	D	*C*	*B*	*A*
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1



Мултиплексер/Демултиплексер

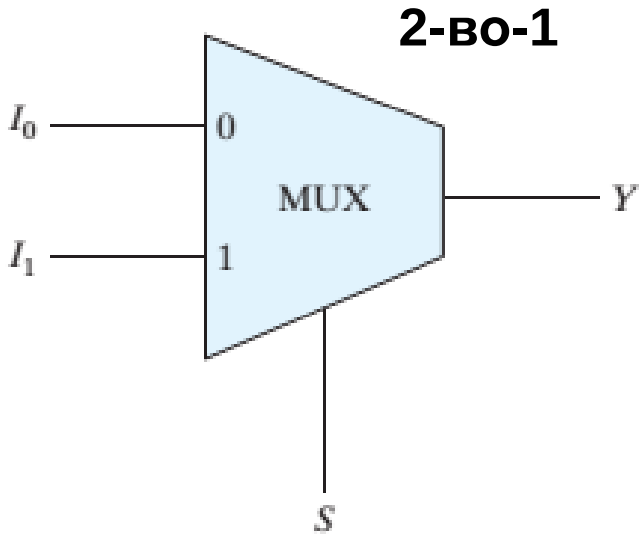
- Мултиплексерот претставува комбинаторно дигитално коло кое одбира една од повеќето влезни линии и ја насочува кон една излезна линија.
  - Логичкото ниво на избраната влезна линија, станува логичко ниво на излезот.
- Демултиплексерот – единствената влезна линија (влез) ја носи (поврзува) со една од повеќето излезни линии
  - Логичкото ниво на единствената влезна линија, станува логичко ниво на една од повеќето излезни линии
- MUX/DeMUX
- Прекинувачка/насочувачка логика
- Контролни линии (линии за одбирање)
- Бројот на контролни линии е $\log_2$ од бројот на влезни линии кај MUX или излезни линии кај DeMUX
- Пример: за 2-во-1 MUX потребна е 1 контролна линија, за 4-во-1 потребни се 2, за 8-во-1 потребни се 3 итн...

MUX

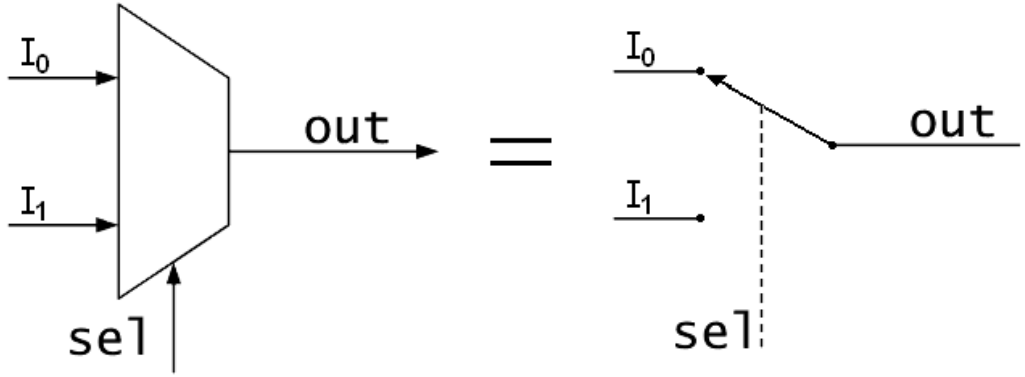


Логички дијаграм

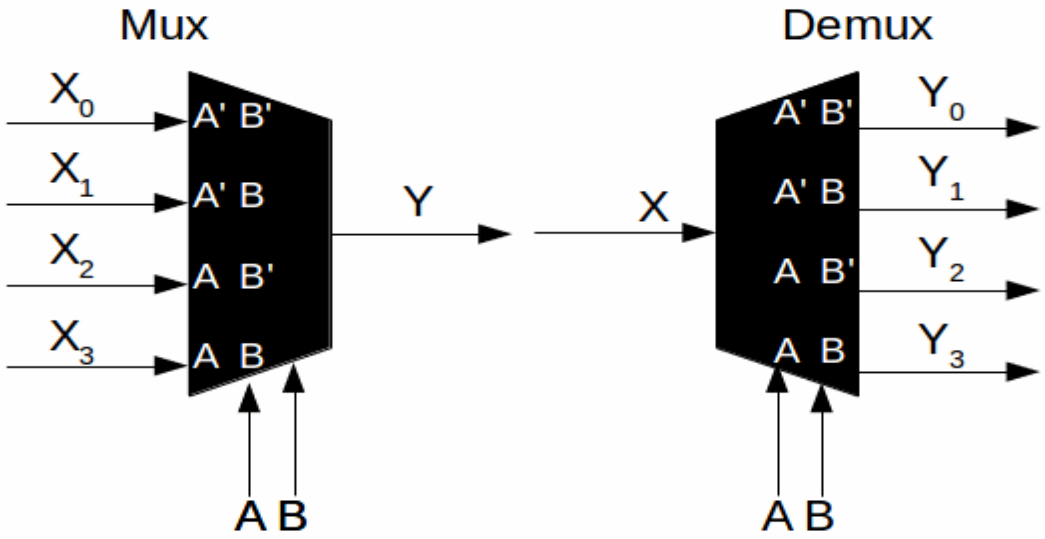
2-во-1



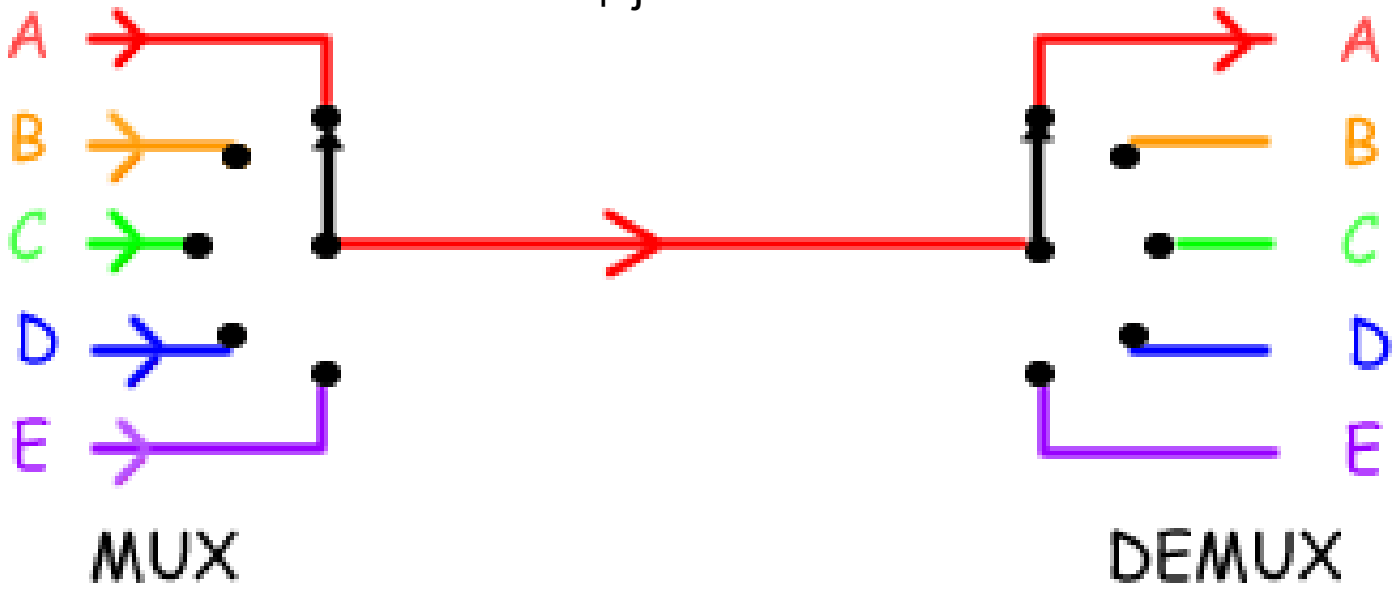
Блок дијаграм (симбол)



MUX/DeMUX

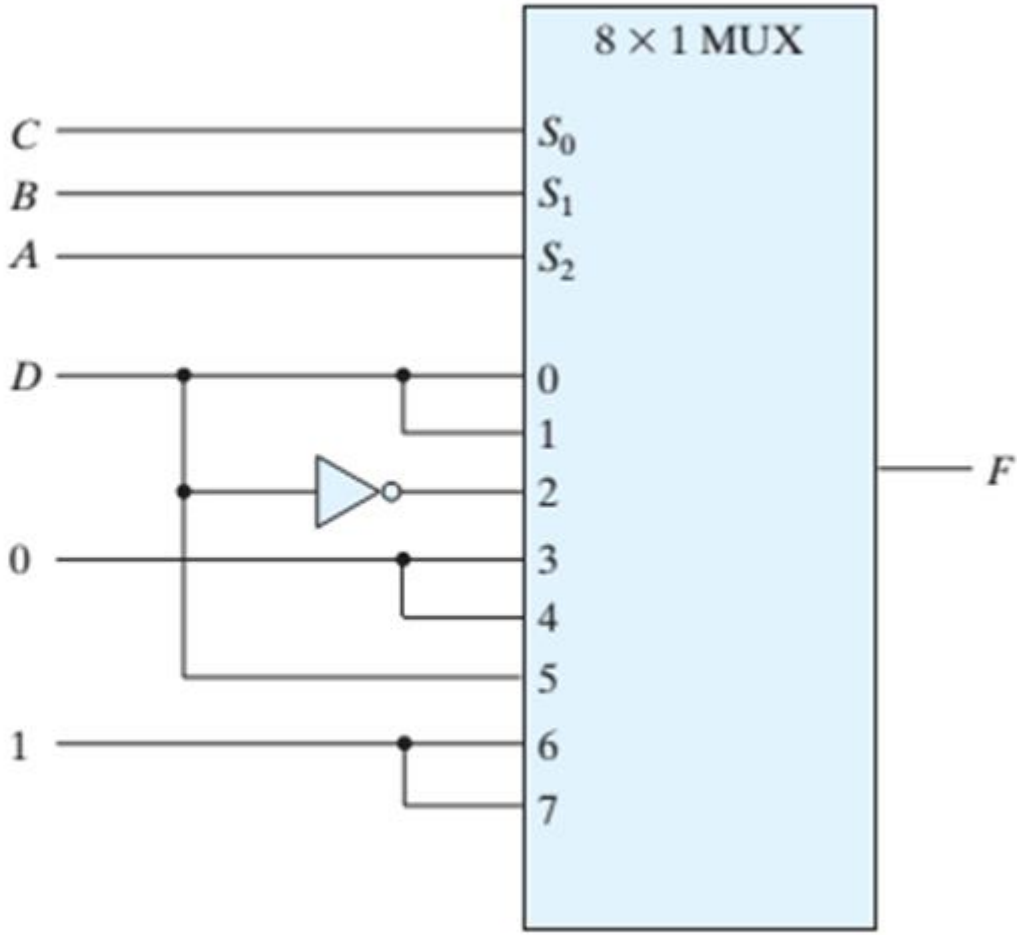


Анимација*



MUX (реализација на логичка функција со 4 влеза со MUX) - Пример

	A	B	C	D	F	
0	0	0	0	0	0	$F = D$
	0	0	0	1	1	
1	0	0	1	0	0	$F = D$
	0	0	1	1	1	
2	0	1	0	0	1	$F = D'$
	0	1	0	1	0	
3	0	1	1	0	0	$F = 0$
	0	1	1	1	0	
4	1	0	0	0	0	$F = 0$
	1	0	0	1	0	
5	1	0	1	0	0	$F = D$
	1	0	1	1	1	
6	1	1	0	0	1	$F = 1$
	1	1	0	1	1	
7	1	1	1	0	1	$F = 1$
	1	1	1	1	1	



Понатаму – Примена на теорема за експанзија (Шенонова теорема за експанзија)

Примери за примена

- Булова теорема за експанзија позната како Шенонова теорема за експанзија или декомпозиција

$$F = x \cdot F_x + x' \cdot F_{x'}$$

- Каде F е било која логичка функција од променливата *x*, *x*’ е комплемент на променливата *x*, а F_x и $F_{x'}$ се вредностите на функцијата F кога x=1 (F_x), и кога x=0 ($F_{x'}$)
- F_x и $F_{x'}$ се нарекуваат шенонови кофактори
- Ова е фундаментална теорема на буловата алгебра

- Да се имплементира f=A’C’+AB+AC со 2-во-1 MUX
- Пример ако f=A’C’+AB+AC, тогаш f може да биде запишана како f=A’*f(0,B,C)+A*f(1,B,C) →

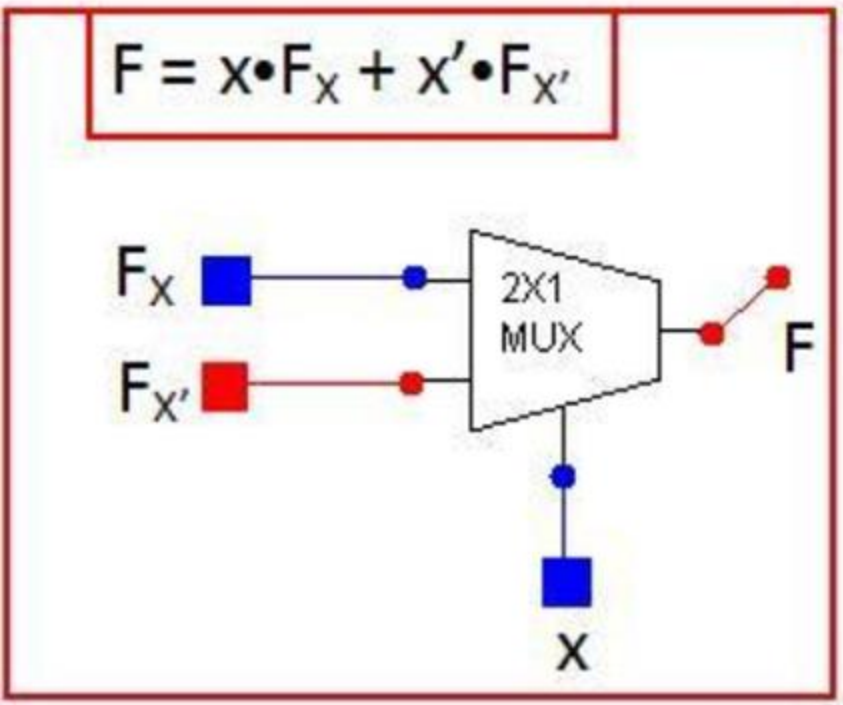
$$f = \bar{A}\bar{C} + AB + AC =$$

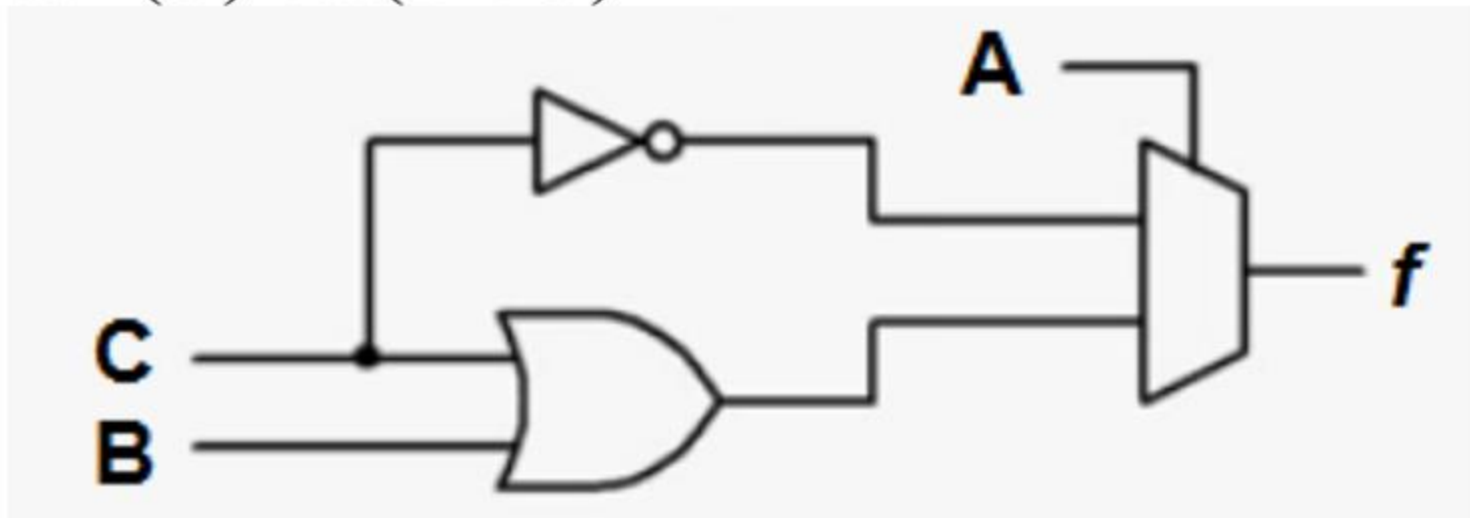
Вадиме 1 променлива

$$\bar{A} * f(0,B,C) + A * f(1,B,C) =$$

$$\bar{A} * (1 * \bar{C} + 0 * B + 0 * C) + A * (0 * \bar{C} + 1 * B + 1 * C) =$$

$$\bar{A} * (\bar{C}) + A(B + C)$$



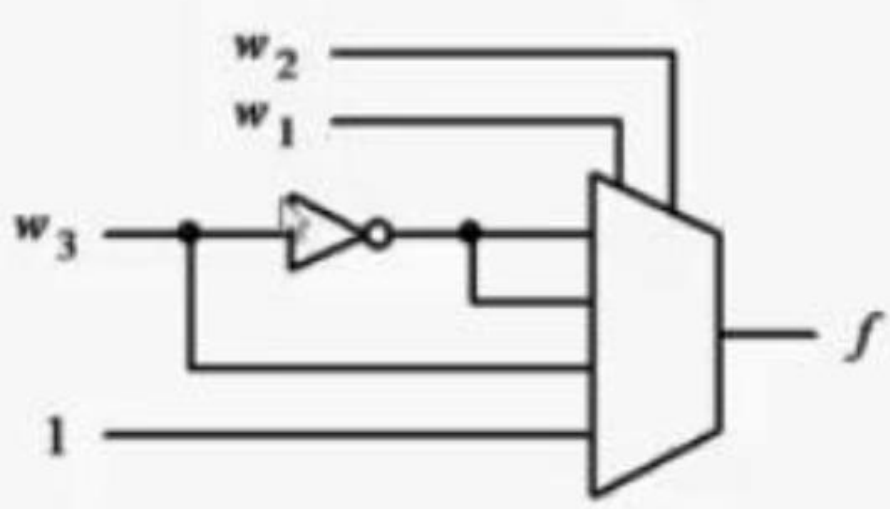


- Со примена на Теоремата за експанзија да се реализира функцијата $f = \overline{w}_1\overline{w}_3 + w_1w_2 + w_1w_3$ со примена на 4-1 MUX
- Овде реализираме со 2 променливи и 4 кофактори (2 контролни влеза)

$$f = \overline{w}_1\overline{w}_2 f(0,0,w_3) + \overline{w}_1 w_2 f(0,1,w_3) + w_1\overline{w}_2 f(1,0,w_3) + w_1 w_2 f(1,1,w_3)$$

$$= \overline{w}_1\overline{w}_2(1\cdot\overline{w}_3 + 0\cdot 0 + 0\cdot w_3) + \overline{w}_1 w_2(1\cdot\overline{w}_3 + 0\cdot 1 + 0\cdot w_3) + w_1\overline{w}_2(0\cdot\overline{w}_3 + 1\cdot 0 + 1\cdot w_3) + w_1 w_2(0\cdot\overline{w}_3 + 1\cdot 1 + 1\cdot w_3)$$

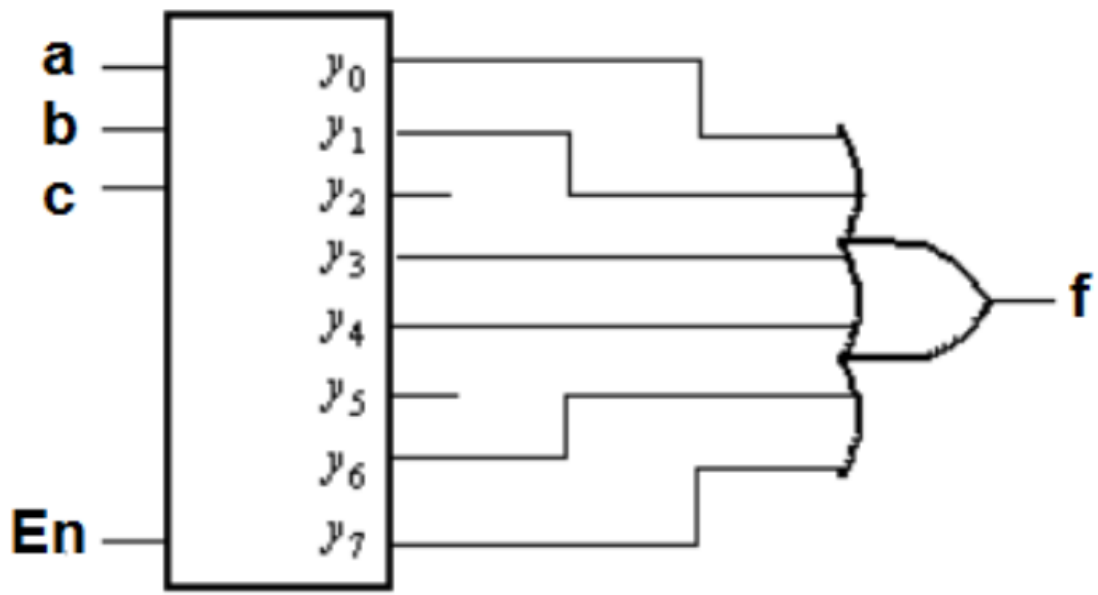
$$= \overline{w}_1\overline{w}_2(\overline{w}_3) + \overline{w}_1 w_2(\overline{w}_3) + w_1\overline{w}_2(w_3) + w_1 w_2(1)$$



- Да се имплементира функцијата со 3-8 бинарен декодер:

$$f(a,b,c) = \sum m(0,1,3,4,6,7)$$

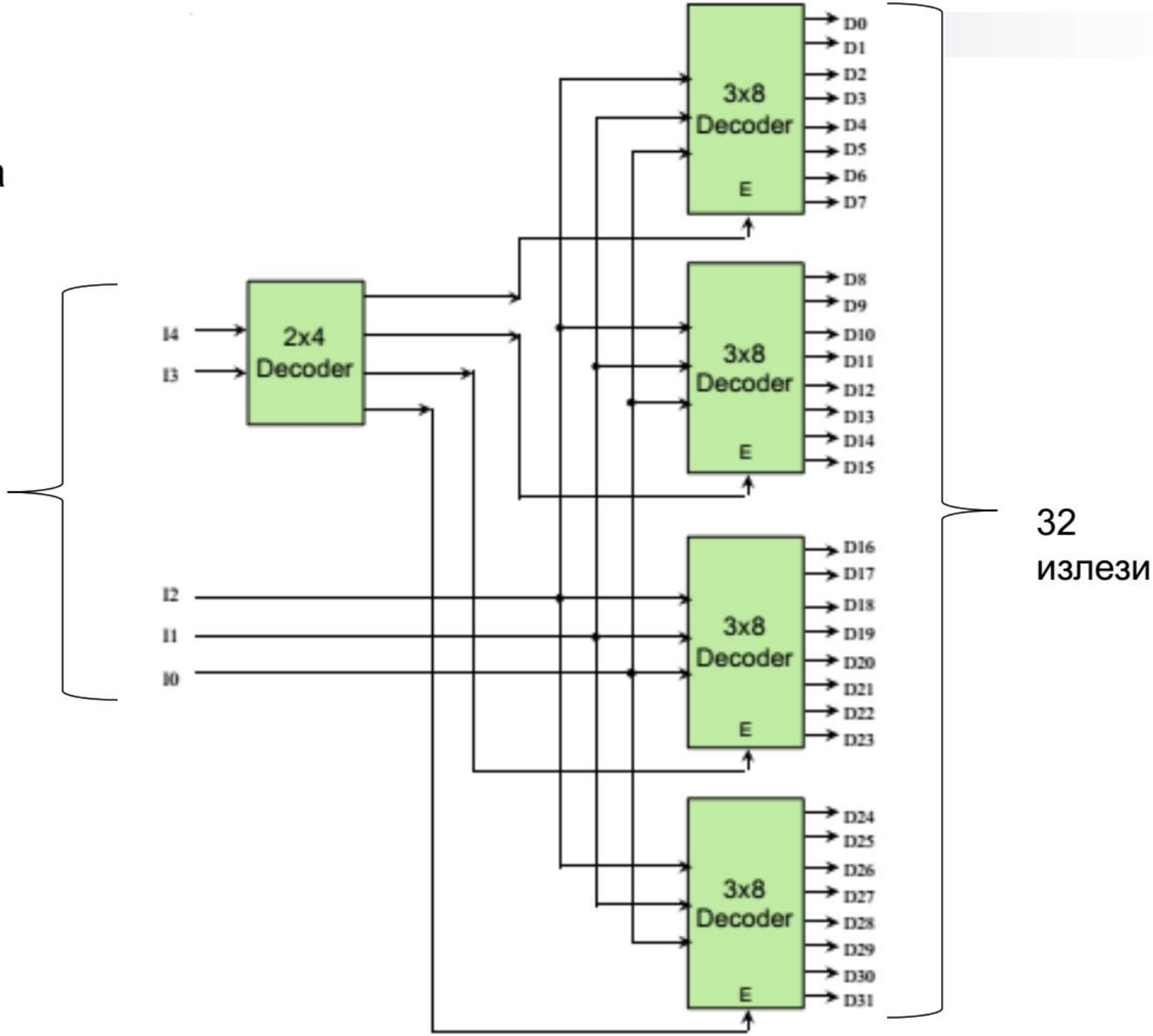
- Решение: декодерот ќе генерира посебен излез за секој минтерм (1 – високо ниво) од функцијата:



- Реализација (експанзија) на декодер со повеќе влезови и излези од декодери со помалку влезови и излези
- Да се реализира 5-во-32 со 3-во-8. Што уште е потребно?
- Потребен е уште еден декодер:
  - 2-во-4
  - Ги активира 4-те 3-во-8
  - Излезите на 2-во-4 се поврзани на линијата за овозможување на 4-те 3-во-8 декодери
  - 3 од влезовите одат директно на влез на 3-8 декодерите
  - 2 од влезовите реализираат мултиплексирачка операција за овозможување/активирање на 3-8 декодерите

- Изведба

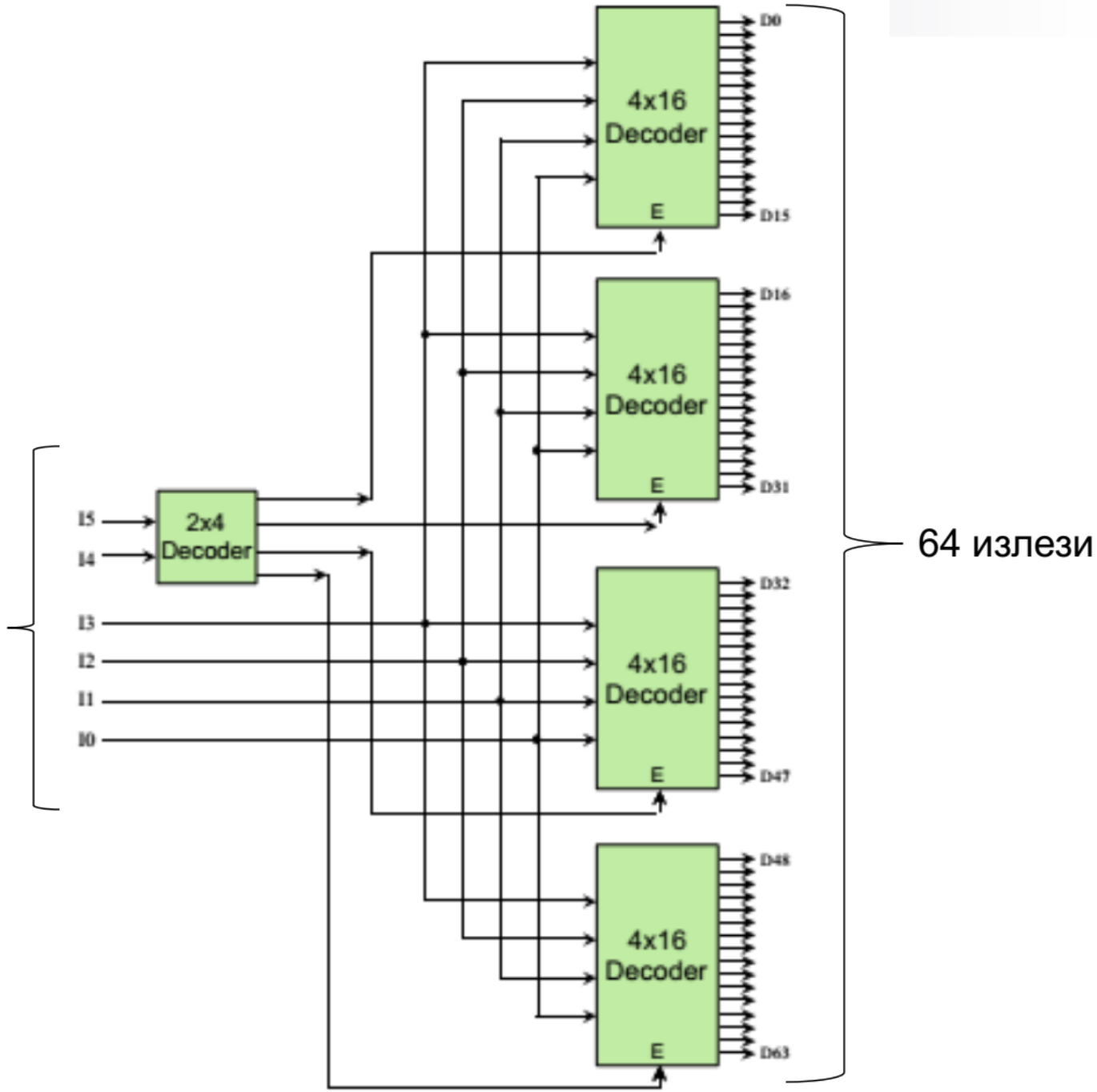
5 влеза



32
излези

- Да се реализира 6-во-64 со 4-во-16. Што уште е потребно?
- Потребен е уште еден декодер:
  - 2-во-4
  - Ги активира 4-те 4-16
  - Излезите на 2-во-4 се поврзани на линијата за овозможување на 4-те 4-во-16 декодери
  - 4 од влезовите одат директно на влез на 4-16 декодерите
  - 2 од влезовите реализираат мултиплексирачка операција за овозможување/активирање на 4-16 декодерите

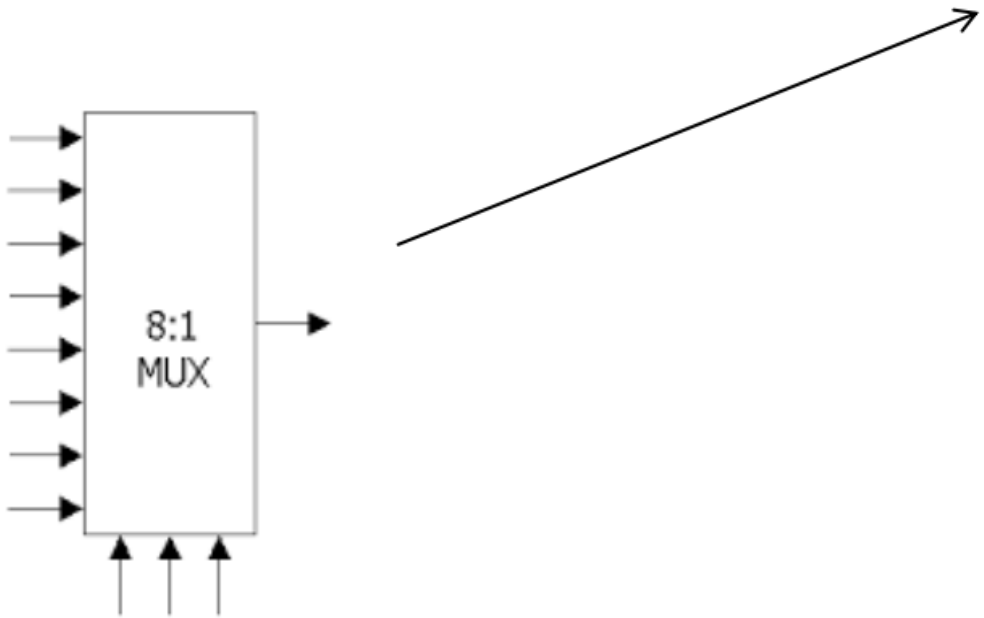
Изведба/Решение

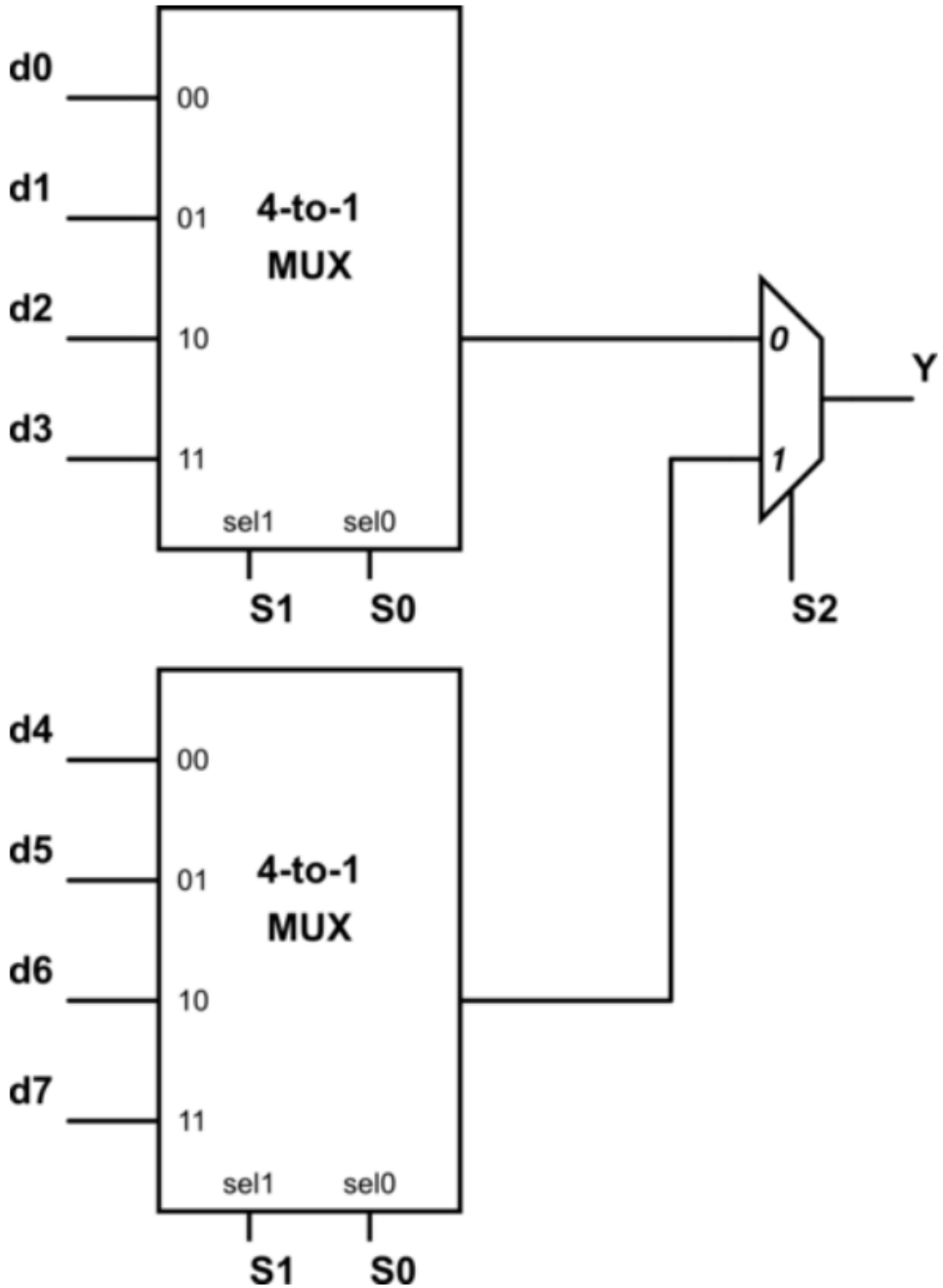


6 влеза

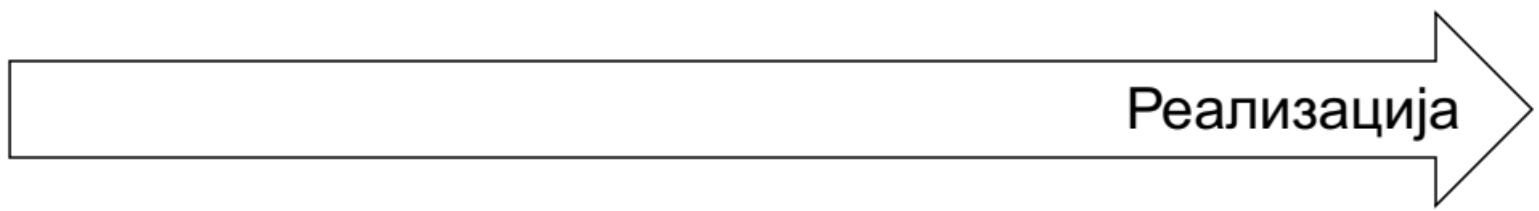
64 излези

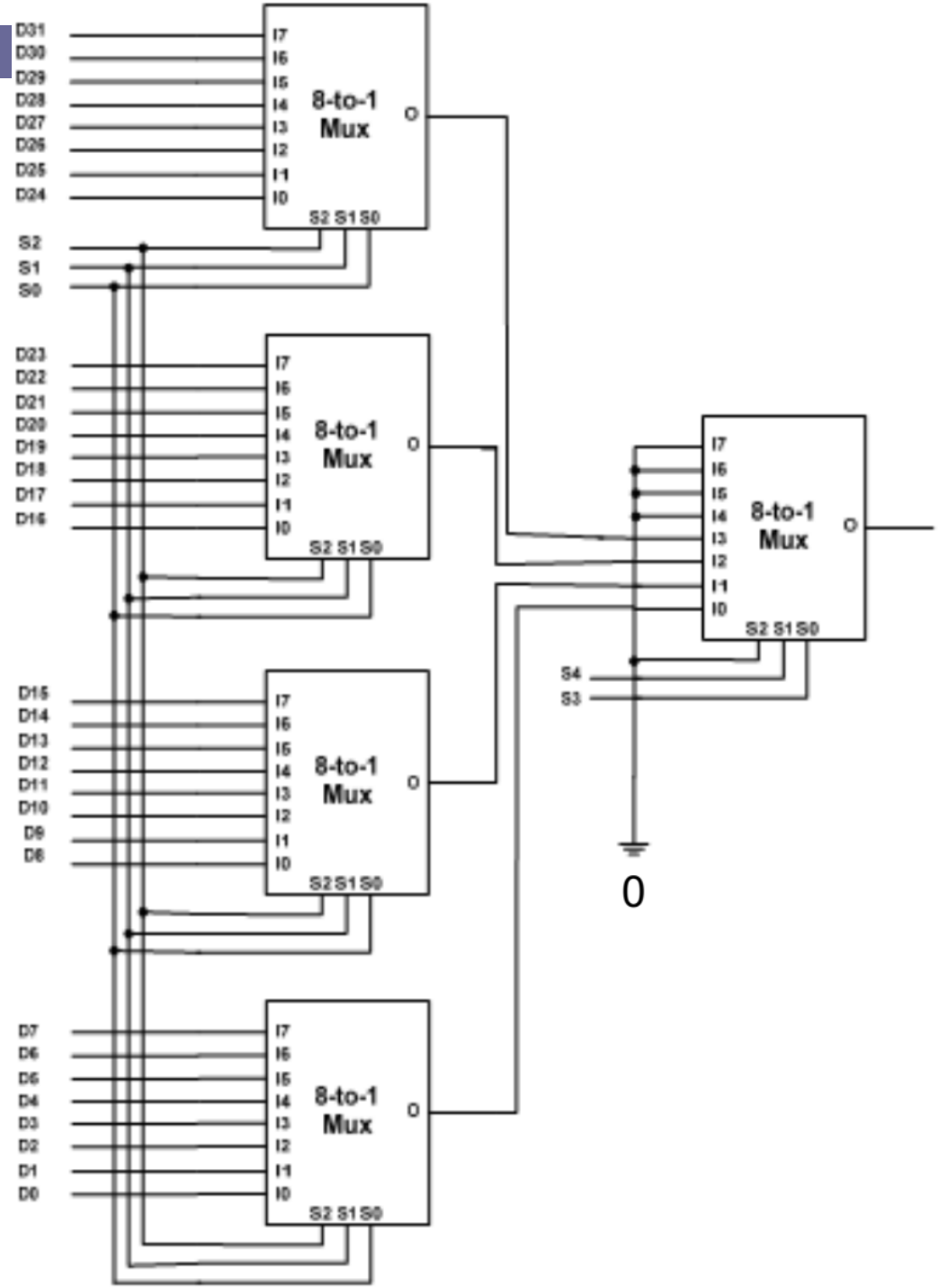
- Да се реализира 8-во-1 MUX со два 4-1 и еден 2-1
- 2-1 MUX ги прима излезите од првиот и вториот 4-1 MUX и ги мултиплексира, а за тоа му е потребна една линија за селекција која одговара на најсигнификантниот бит за селекција на целиот 8-1 MUX





- Да се реализира 32-во-1 MUX само со примена на 8-во-1
- Бидејќи ни се потребн 4 8-во-1 MUX да обезбедиме 32 влеза, ќе имаме четири излези кои понатаму треба да ги мултиплексираме во еден
- Бидејќи на располагање имаме само 8-во-1 MUX тогаш само 4 влезови треба да се активни, останатите ги поврзуваме со 0
- За 4 влезови потребни се 2 селекциони линии, а 8-1 MUX има 3. Двте селекциони линии ги користиме, а третата ја врзуваме на нула бидејќи немаме потреба да ги адресираме сите 8 влезови туку само 4




D31
D30
D29
D28
D27
D26
D25
D24
I7
I6
I5
I4
I3
I2
I1
I0
8-to-1
Mux
O
S2 S1 S0
S2
S1
S0
D23
D22
D21
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
S4
S3
0