Návrh číslicových systémů (INC)

Otto Fučík

Vysoké učení technické v Brně
Fakulta informačních technologií
Božetěchova 2, 612 66 Brno

Použitá literatura

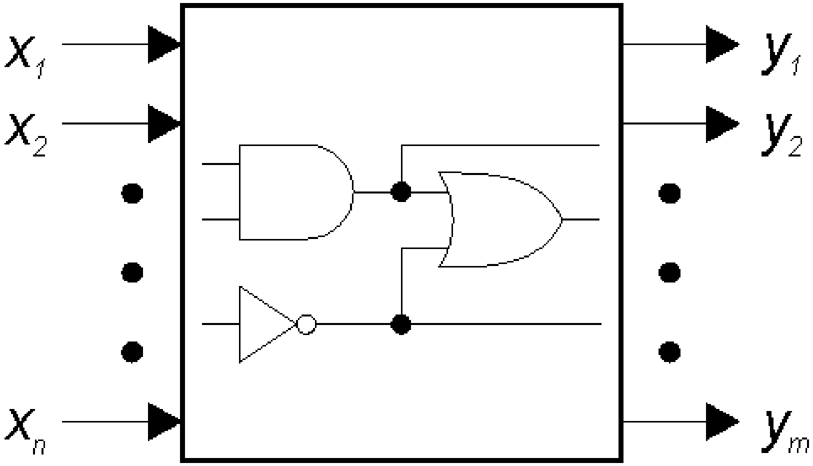
N. Frištacký, M. Kolesár, J. Kolenička a J. Hlavatý: „Logické systémy", SNTL Praha, 1986
M. Eysselt: „Logické systémy", SNTL Praha, skriptum VUT v Brně, 1985
J. F. Wakerly: „Digital Design. Principles and Practices", Prentice Hall, ISBN 0-13-769191-2, 2000
V. P. Nelson, H.T.Nagle, B.D.Carroll, J.D.Irwin: „Digital Logic Circuit Analysis & Design", ISBN 0-13-463894-8, 1995
T.L.Floyd: „Digital Fundamentals", Prentice Hall, ISBN 0-13-080850-4, 2000
D.J Smith: „HDL Chip Design", ISBN 0-9651934-3-8, 1996

Kombinační logické obvody

Kombinační logické obvody

- Hierarchicky uspořádaný obvod, ve kterém jednotlivé komponenty zpracovávají a mezi sebou komunikují informaci reprezentovanou v binární podobě (log. úrovně)

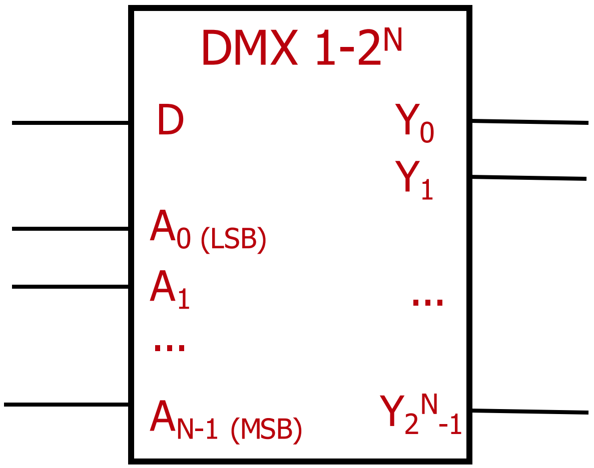
  - Každá komponenta má kombinační chování
  - Vstup každé komponenty je připojen pouze k jednomu výstupu předchozí komponenty nebo zdroji log. „0" či „1"
  - Výstupy nelze spojovat
  - Pozn.: Pouze v případě tzv. montážní logiky se spojením výstupů hradel (např. s tzv. otevřeným kolektorem) realizují log. funkce - viz dále
  - Struktura neobsahuje cykly (zpětné vazby)
  - Funkční a časové chování lze odvodit z funkčního a časového chování jednotlivých komponent

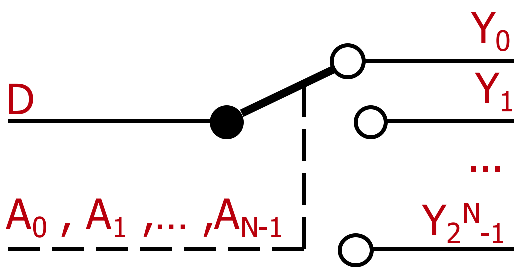


Kombinační obvody: Moduly

- Z praktického hlediska je účelné vytvářet funkční moduly, které
  - Jsou sestaveny z jednodušších komponent (log. členů, jednodušších modulů)
  - Vykonávají specifickou (často používanou) funkci
  - Slouží jako stavební bloky složitějších log. obvodů
  - Jsou ekonomické - vyrábí se ve velkých sériích
- Příklad
  - Demultiplexor
  - Dekodér
  - Multiplexor
  - Kodér
  - Sčítačka
  - Posouvač
  - Násobička atd.

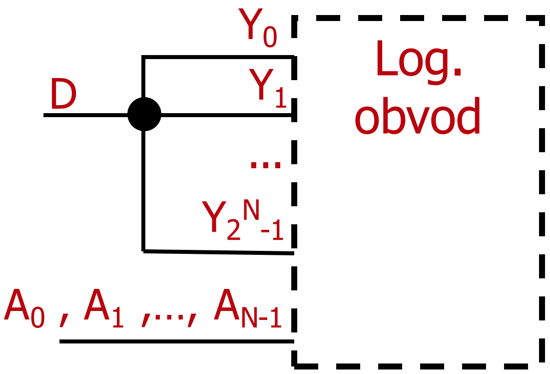
- Charakteristika
  - Demultiplexor je kombinační log. síť s jedním datovým vstupem D, N adresovými vstupy A0..A_{N-1} a 2^N výstupy $Y_0..Y_{2^N-1}$
  - Přenáší logickou hodnotu z datového vstupu D na jeden z 2^N výstupů, přičemž ostatní výstupy mají neaktivní log. úrovně
  - Výstup je určen binární hodnotou adresovacích vstupů
  - Funguje jako přepínač jednoho vstupu na 2^N výstupů
  - Značí se DMX 1-2^N
  - Platí $Y_i=D\cdot m_i$, kde
    - Y_i…výstup i
    - D…datový vstup
    - m_i…minterm i určený adresou A





Demultiplexor (anglicky Demultiplexer)

- Pokud se na datový vstup D převede konstanta 0 nebo 1, dostáváme tzv. dekodér s výstupy aktivními v log. 0, resp. v log. 1, viz dále
- Funkci DMX lze v některých případech nahradit propojením datového vstupu D se všemi výstupy
  - Logické obvody, které jsou připojeny na výstup demultiplexoru, nejsou „aktivovány" signály $Y_0,\ldots,Y_{2^N-1}$, ale čtou si příslušnou log. hodnotu signálu $D=Y_0,\ldots,Y_{2^N-1}$ na základě adresy $A_0,\ldots,A_{N-1}$
  - Demultiplexor tedy nepřenáší hodnotu datového vstupu na výstup, ale tato hodnota je přivedena na vstupy všech následujících obvodů, které si ji čtou na základě adresy samy



Demultiplexor 1-4

- Příklad implementace
  - Pravdivostní tabulka
  - Výraz
  - Logické značky
  - Logické schéma

$$Yi = D \cdot m_i$$

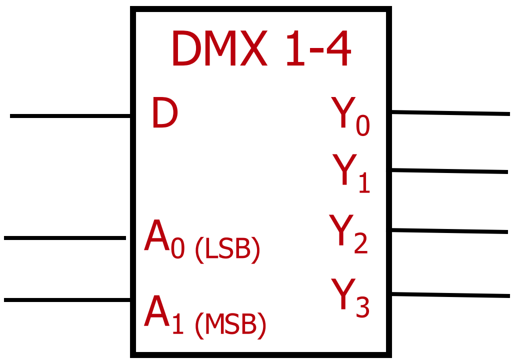
$$Y_0 = D \cdot \overline{A_1} \cdot \overline{A_0} = D \cdot m_0$$

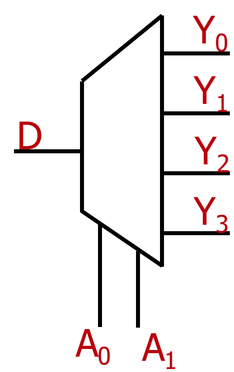
$$Y_1 = D \cdot \overline{A_1} \cdot A_0 = D \cdot m_1$$

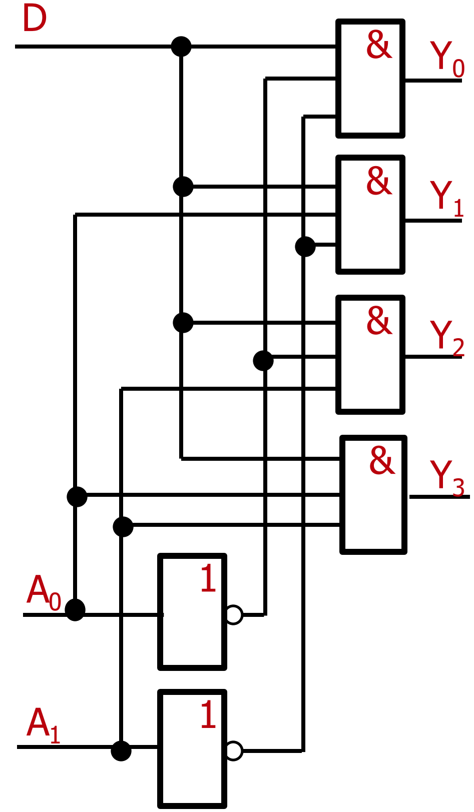
$$Y_2 = D \cdot A_1 \cdot \overline{A_0} = D \cdot m_2$$

$$Y_3 = D \cdot A_1 \cdot A_0 = D \cdot m_3$$

Vstupy			Výstupy			
D	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	**0**
0	0	1	0	0	**0**	0
0	1	0	0	**0**	0	0
0	1	1	**0**	0	0	0
1	0	0	0	0	0	**1**
1	0	1	0	0	**1**	0
1	1	0	0	**1**	0	0
1	1	1	**1**	0	0	0



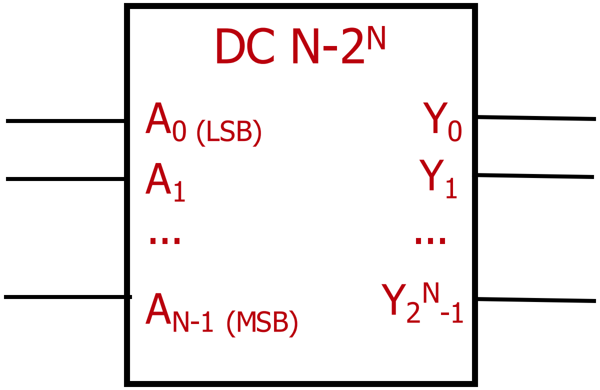




Dekodér (anglicky Decoder)

- Charakteristika
  - Pokud na datový vstup D demultiplexoru přivedeme hodnotu log. 1, dostáváme dekodér s výstupy aktivními v log. 1
  - Dekodér je kombinační log. síť s N adresovými vstupy a 2^N výstupy
  - Převádí binární kód (N bitů) na kód 1 z $M=2^N$
  - Nazýváme též N-bitový dekodér
  - Binární kód na vstupu určuje, který výstup bude platný (vždy jen jeden)
  - Některé výstupy mohou být nevyužity – např. BCD dekodér, viz dále
  - Značí se DC N-2^N (též DC 1-2^N)
- Použití
  - Adresový dekodér (paměti)
  - Generování log. funkcí
  - Dekodér pro displeje atd.

$$Y_i = m_i$$



- Příklad implementace pomocí demultiplexoru 1-4
  - Pozn.: m_i = minterm i

$$Y_i = m_i$$

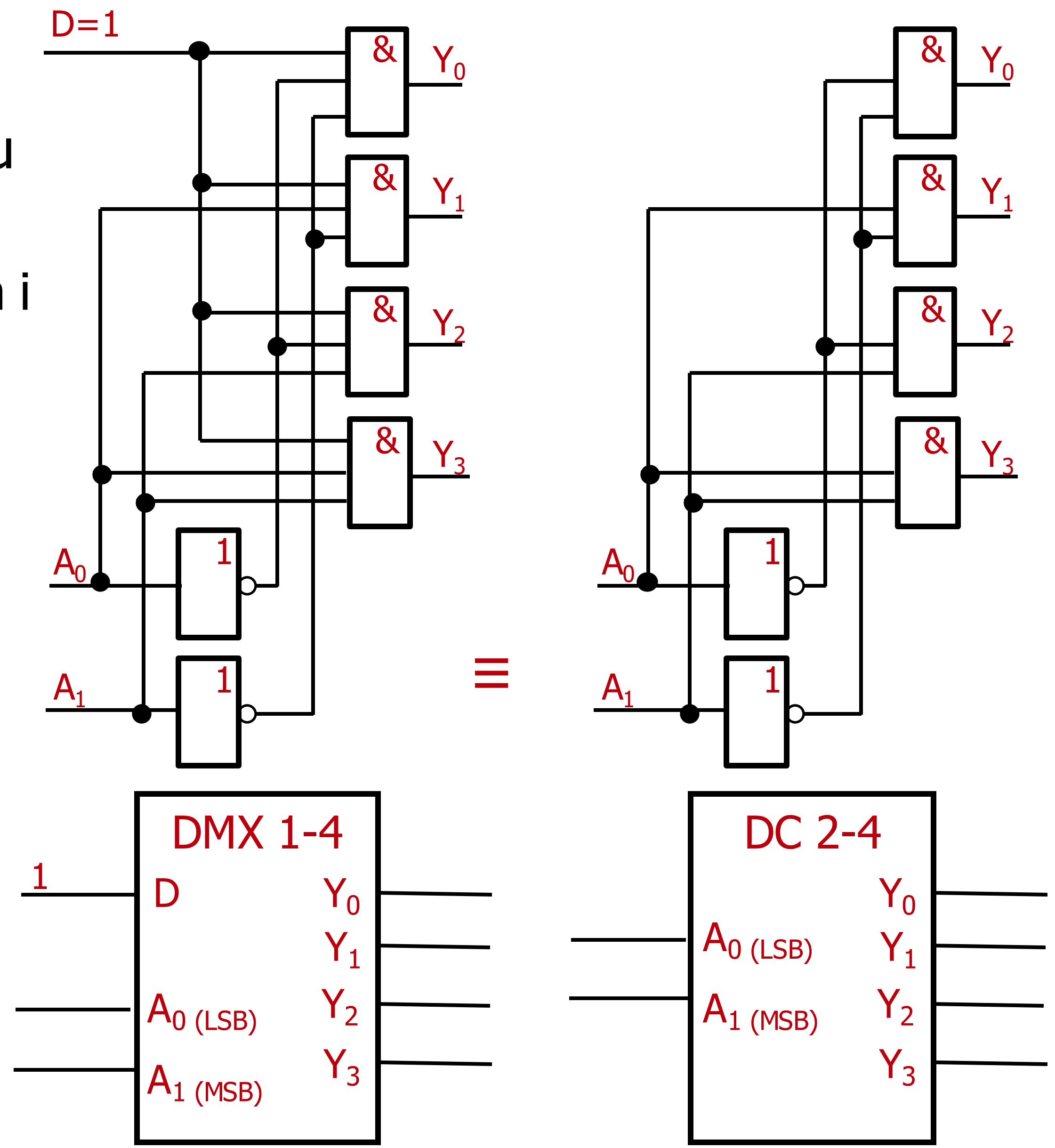
$$Y_0 = \overline{A_1} \cdot \overline{A_0} = m_0$$

$$Y_1 = \overline{A_1} \cdot A_0 = m_1$$

$$Y_2 = A_1 \cdot \overline{A_0} = m_2$$

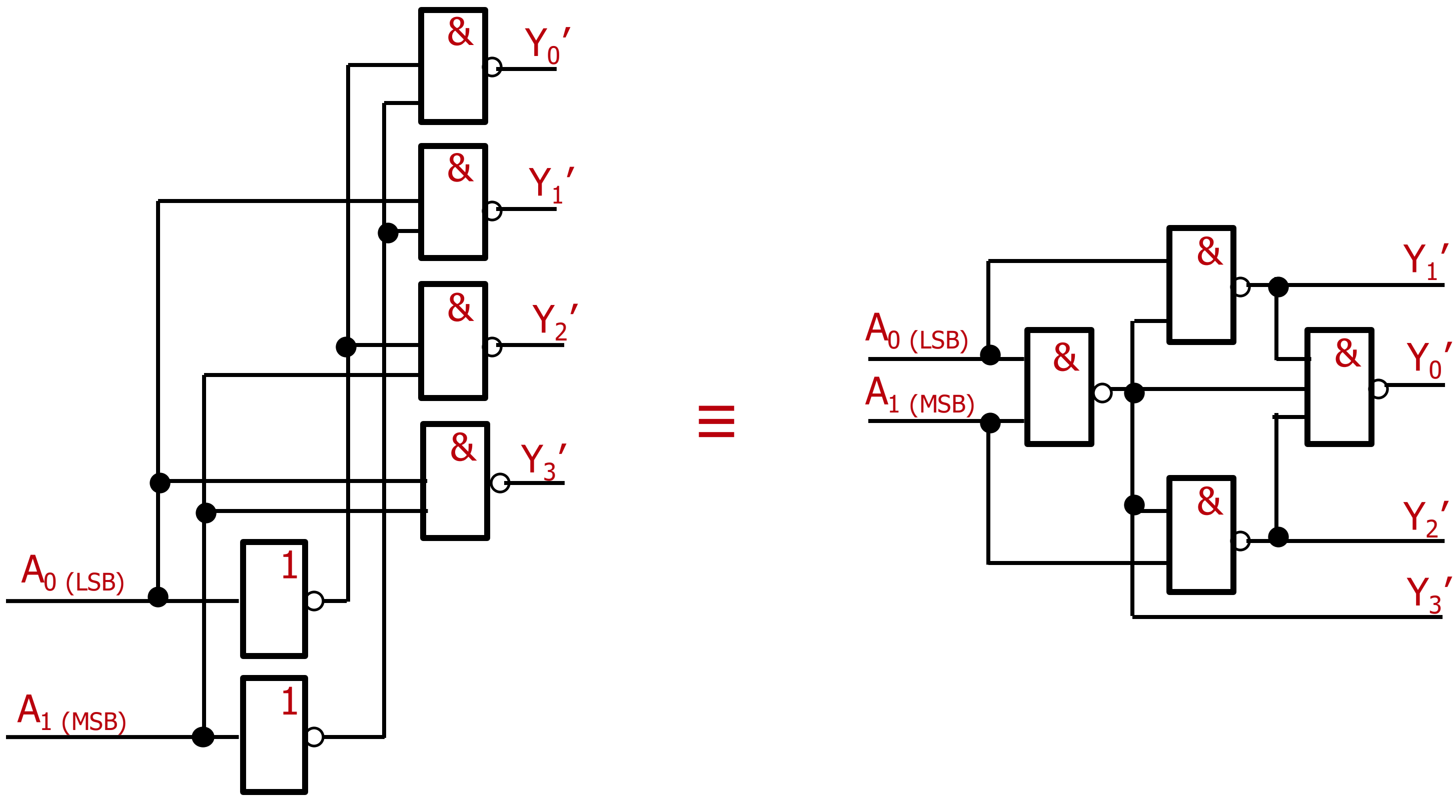
$$Y_3 = A_1 \cdot A_0 = m_3$$

Vstupy		Výstupy			
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	**1**
0	1	0	0	**1**	0
1	0	0	**1**	0	0
1	1	**1**	0	0	0



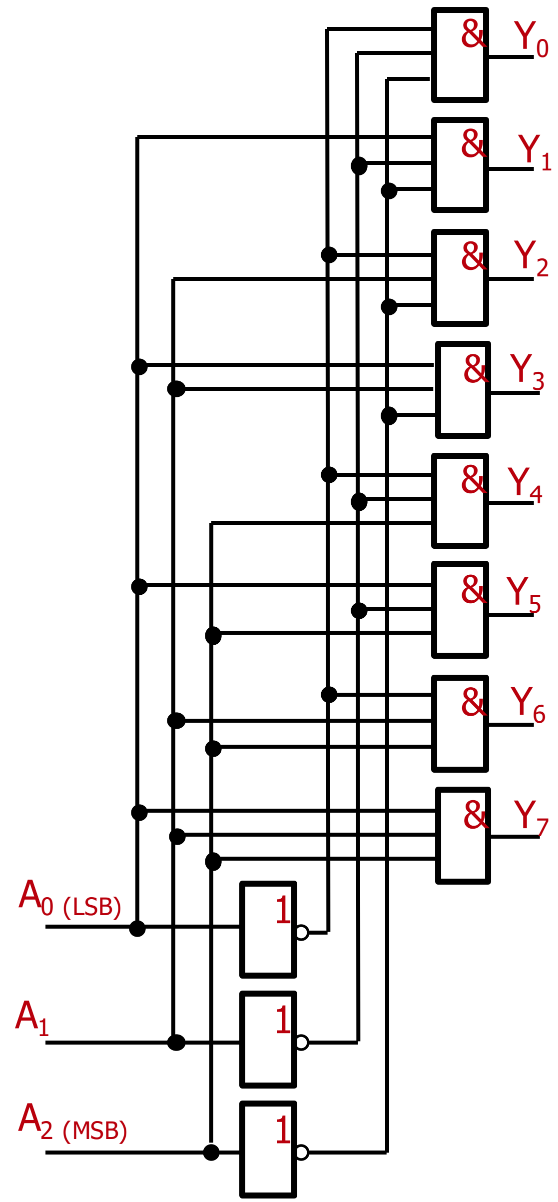
Dekodér 2-4 (1 ze 4)

- Příklady realizace
  - S výstupy aktivními v nule (použité členy INV, NAND)
  - Alternativní řešení s výstupy aktivními v nule (pouze členy NAND)



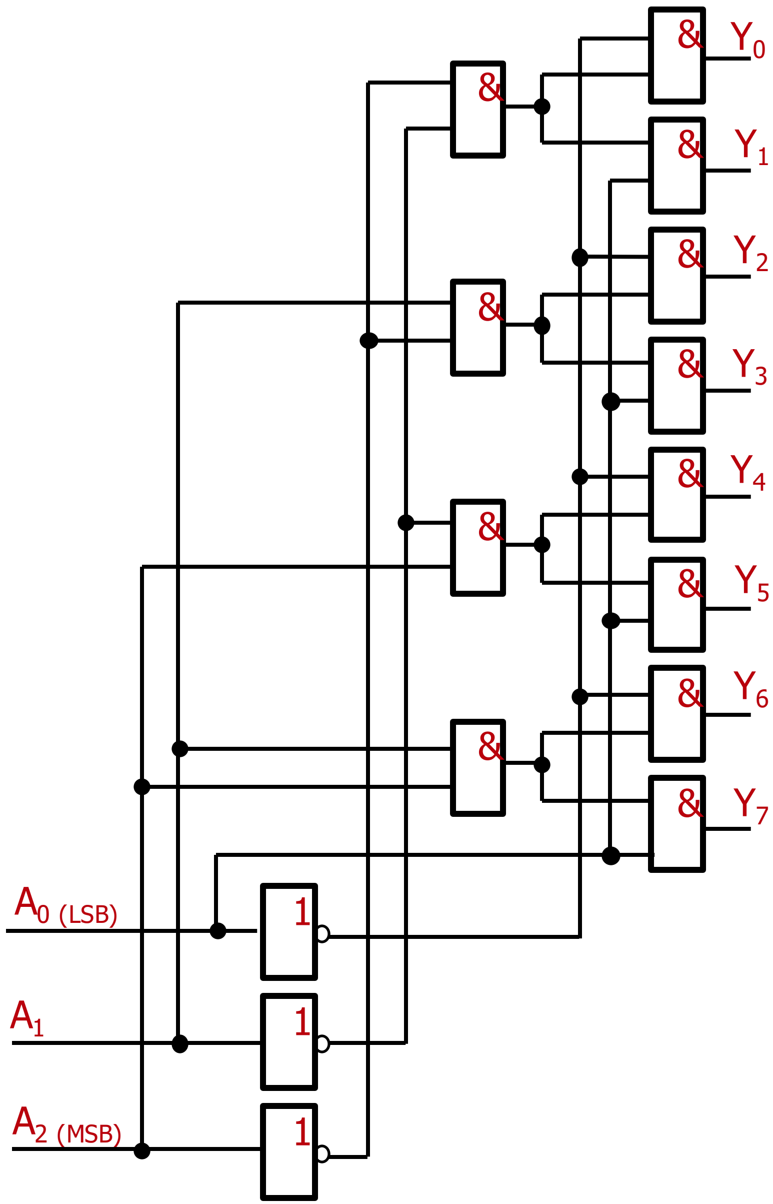
Dekodér 3-8 (1 z 8)

- Paralelní struktura
  - Pro každý výstup je třeba realizovat jeden minterm - log. člen AND s N vstupy, kde N je rovno počtu vstupů
  - V praxi jsou však počty vstupů log. členů AND limitovány technologickými možnostmi - nutno použít stromovou či maticovou strukturu



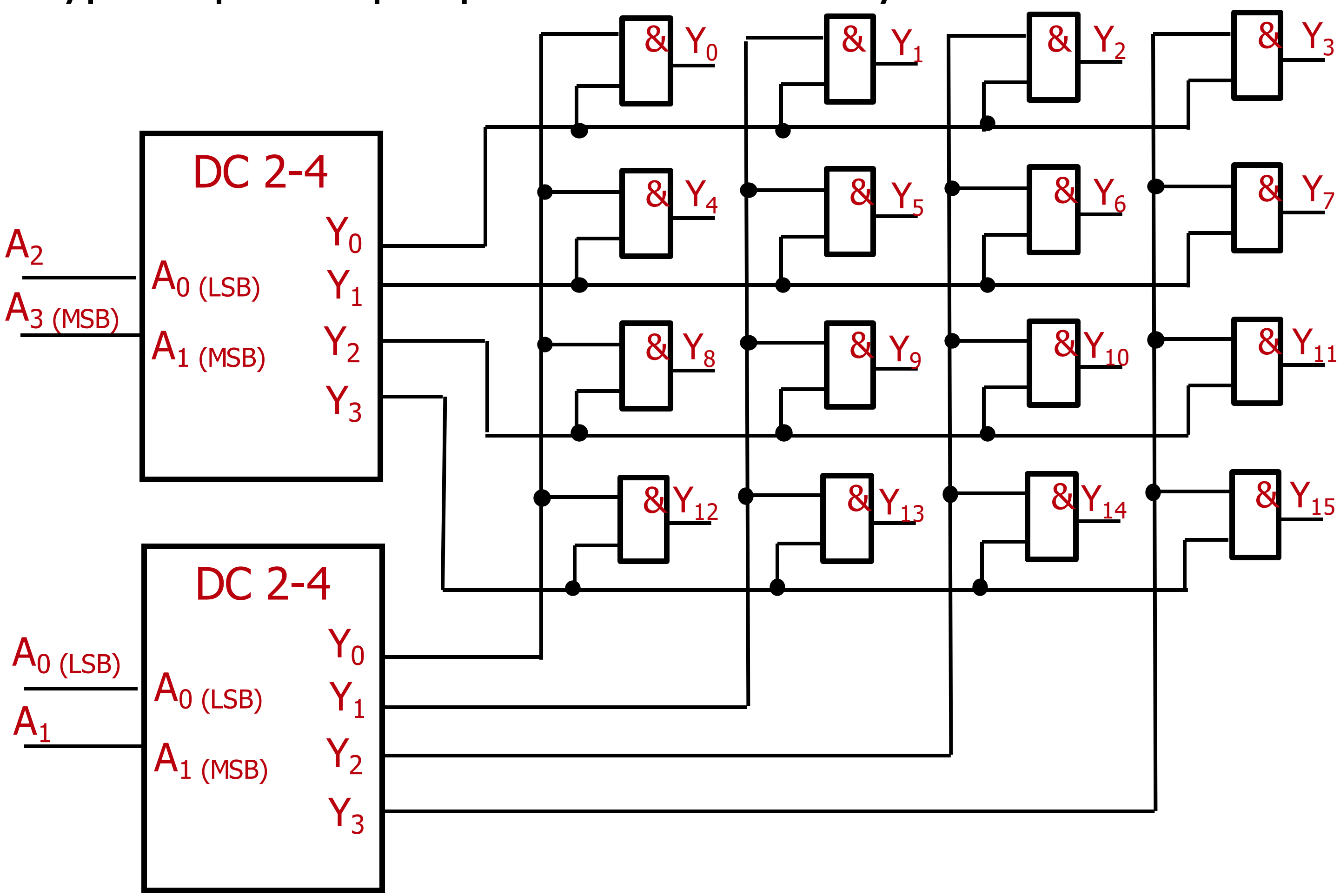
Dekodér 3-8 (1 z 8)

- Stromová struktura
  - Lze použít hradla AND s méně vstupy, než je počet vstupních proměnných
  - Větší zpoždění než paralelní struktura



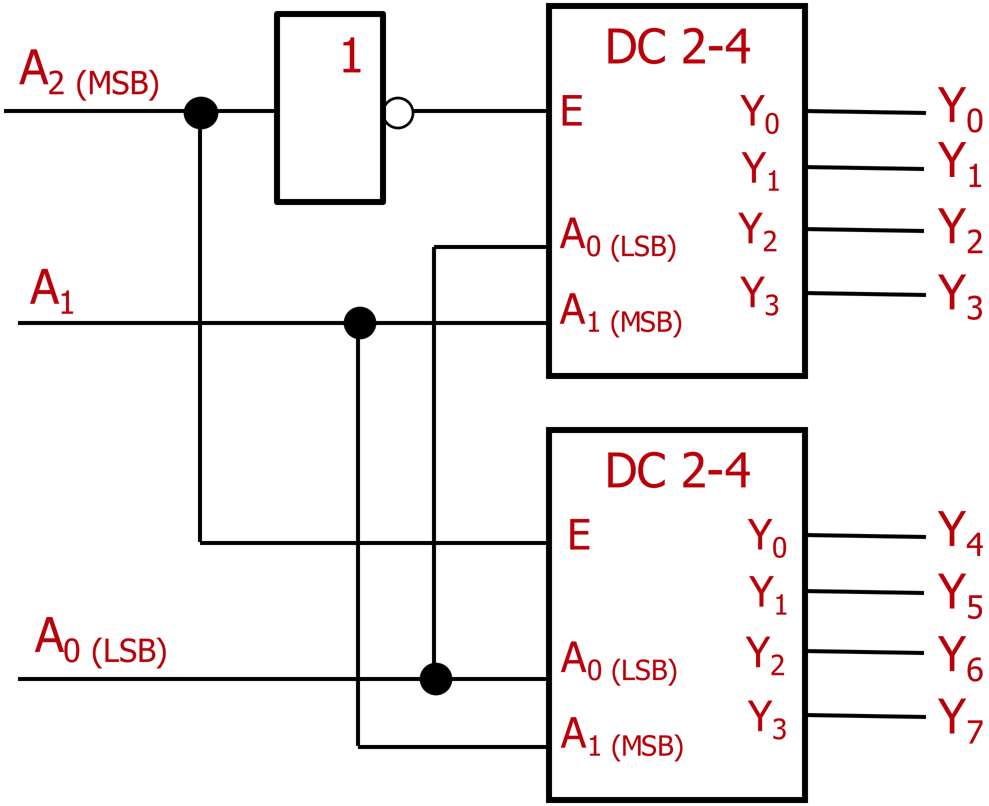
Dekodér 4-16 (1 ze 16)

- Maticová struktura
  - Vhodné pro velký počtu výstupů
  - Typické použití pro paměťové dekodéry

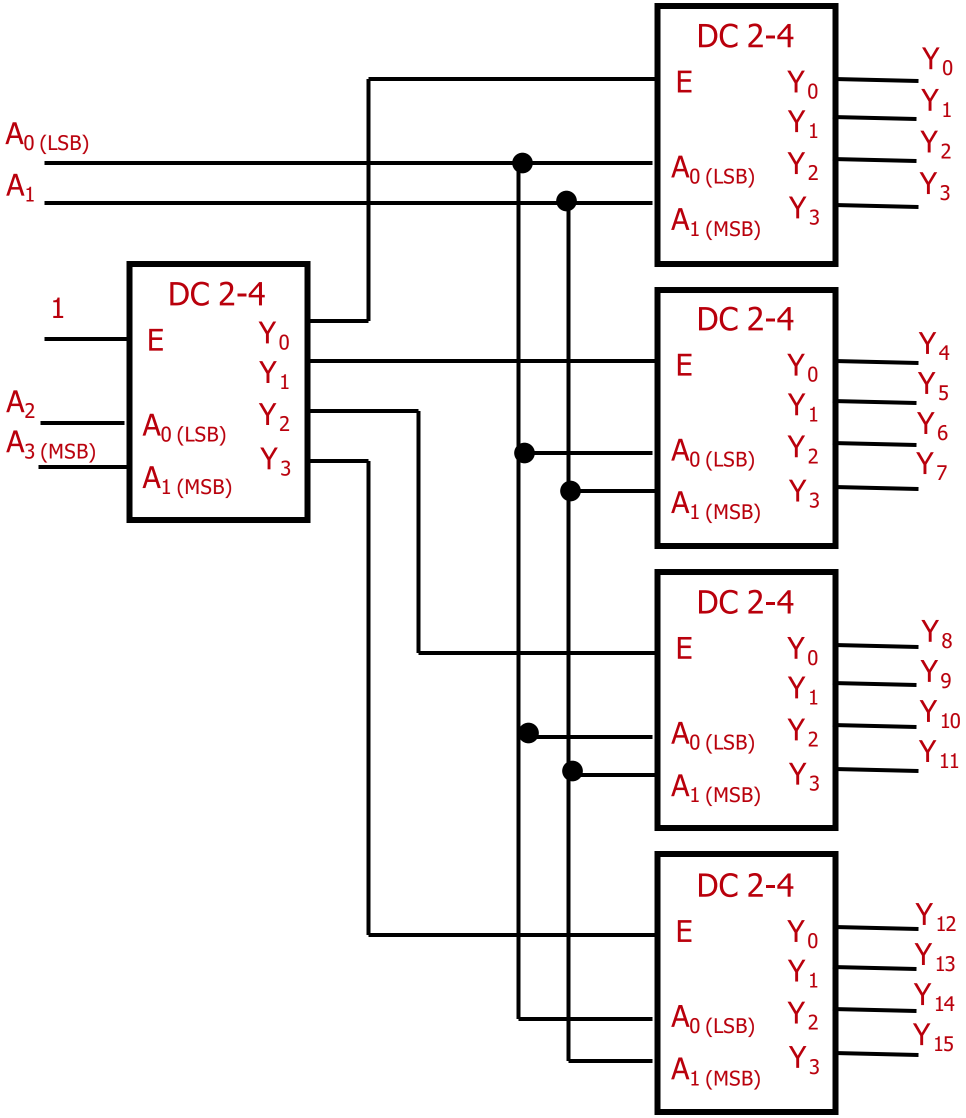


Dekodér: Více výstupů

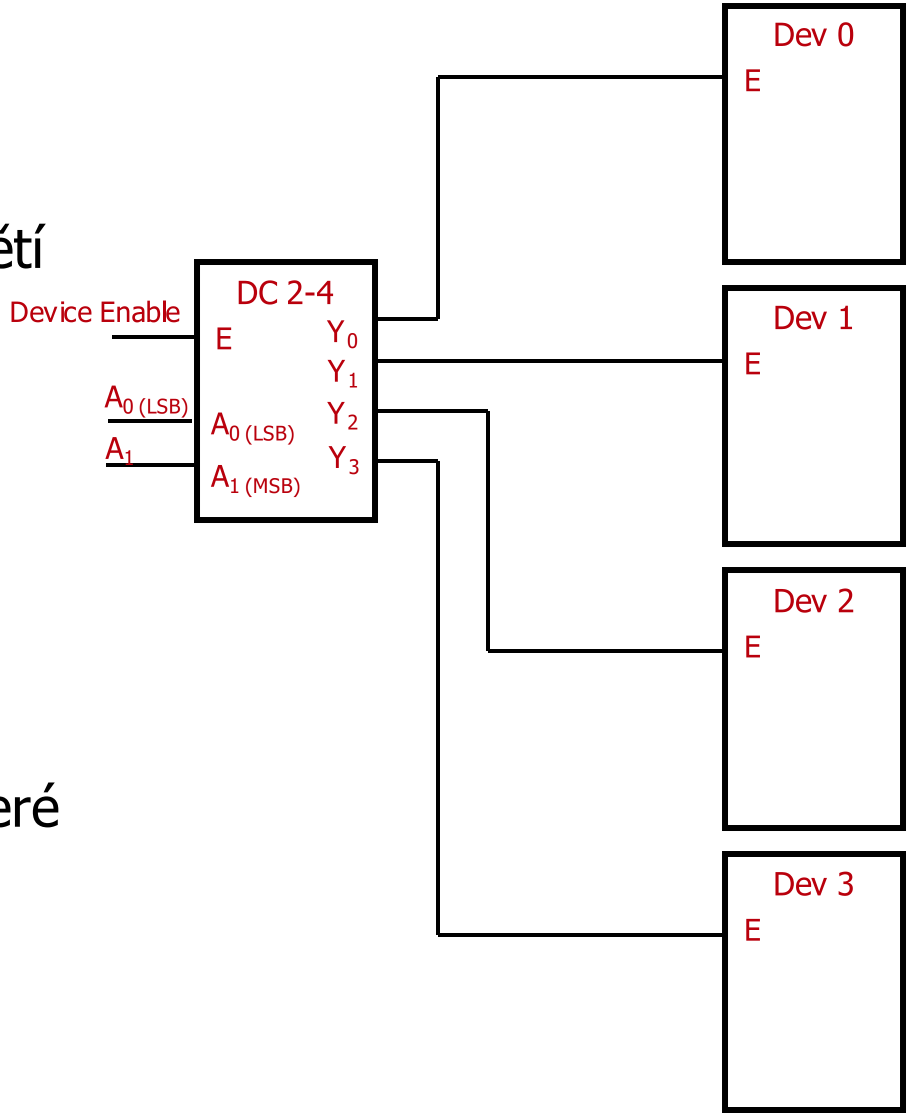
- Dekodér s povolovacím (datovým) vstupem (= demultipexor)
  - Lze využít k budování rozsáhlejších dekodérů z jednodušších
- Ukázka postupu při návrhu zdola-nahoru
  - Z jednodušších komponent se staví složitější
- Příklad
  - Dekodér 3-8 (1 z 8) ze dvou dekodérů 2-4 (1 ze 4)



- Příklad
  - Dekodér 4-16 (1 ze 16) ze čtyř dekodérů 2-4 (1 ze 4)

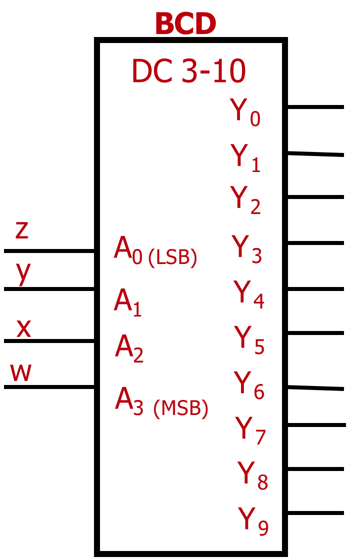


- Adresový dekodér
  - N-bitová adresa je dekódována pro výběr jednoho z 2^n vstupně-výstupních zařízení či pamětí
- Příklad
  - 4bitový dekodér umožňuje vybrat jedno ze 4 zařízení připojených např. v adresovém prostoru procesoru
  - Signál „Device Enable” povoluje výběr zařízení, které je určeno adresou A_1A_0



Dekodér: Dekadický (Binary Coded Decimal - BCD)

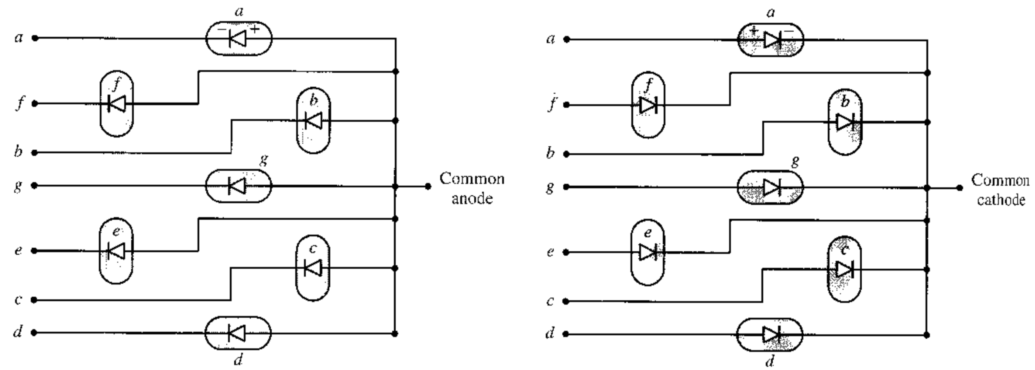
- Dekóduje kód BCD (prvních deset binárních čísel) na kód „1 z 10” (vždy pouze jeden z deseti výstupů je aktivní)

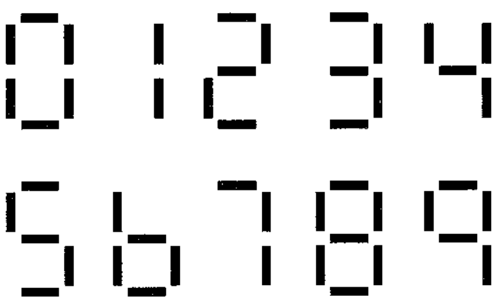


#	A_3	A_2	A_1	A_0	Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
1	0	0	0	1	0	0	0	0	0	0	0	0	**1**	0
2	0	0	1	0	0	0	0	0	0	0	0	**1**	0	0
3	0	0	1	1	0	0	0	0	0	0	**1**	0	0	0
4	0	1	0	0	0	0	0	0	0	**1**	0	0	0	0
5	0	1	0	1	0	0	0	0	**1**	0	0	0	0	0
6	0	1	1	0	0	0	0	**1**	0	0	0	0	0	0
7	0	1	1	1	0	0	**1**	0	0	0	0	0	0	0
8	1	0	0	0	0	**1**	0	0	0	0	0	0	0	0
9	1	0	0	1	**1**	0	0	0	0	0	0	0	0	0
10	1	0	1	0	0	0	0	0	0	0	0	0	0	0
11	1	0	1	1	0	0	0	0	0	0	0	0	0	0
12	1	1	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0	0	0	0	0	0
14	1	1	1	0	0	0	0	0	0	0	0	0	0	0
15	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Dekodér: Kód BCD na 7-segmentový displej

- Sedmisegmentový displej LED
  - Se společnou anodou (dekodér s výstupy aktivními v nule)
  - Se společnou katodou (dekodér s výstupy aktivními v jedničce)



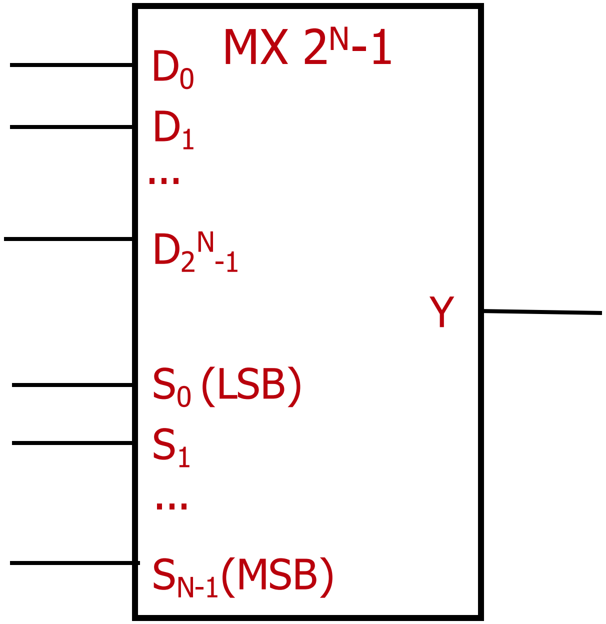


- Pravdivostní tabulka
  - Výstupy aktivní v jedničce

Číslo	**BCD kód**				**Segmenty displeje**						
#	A_3	A_2	A_1	A_0	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	0	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	0	0	1	1

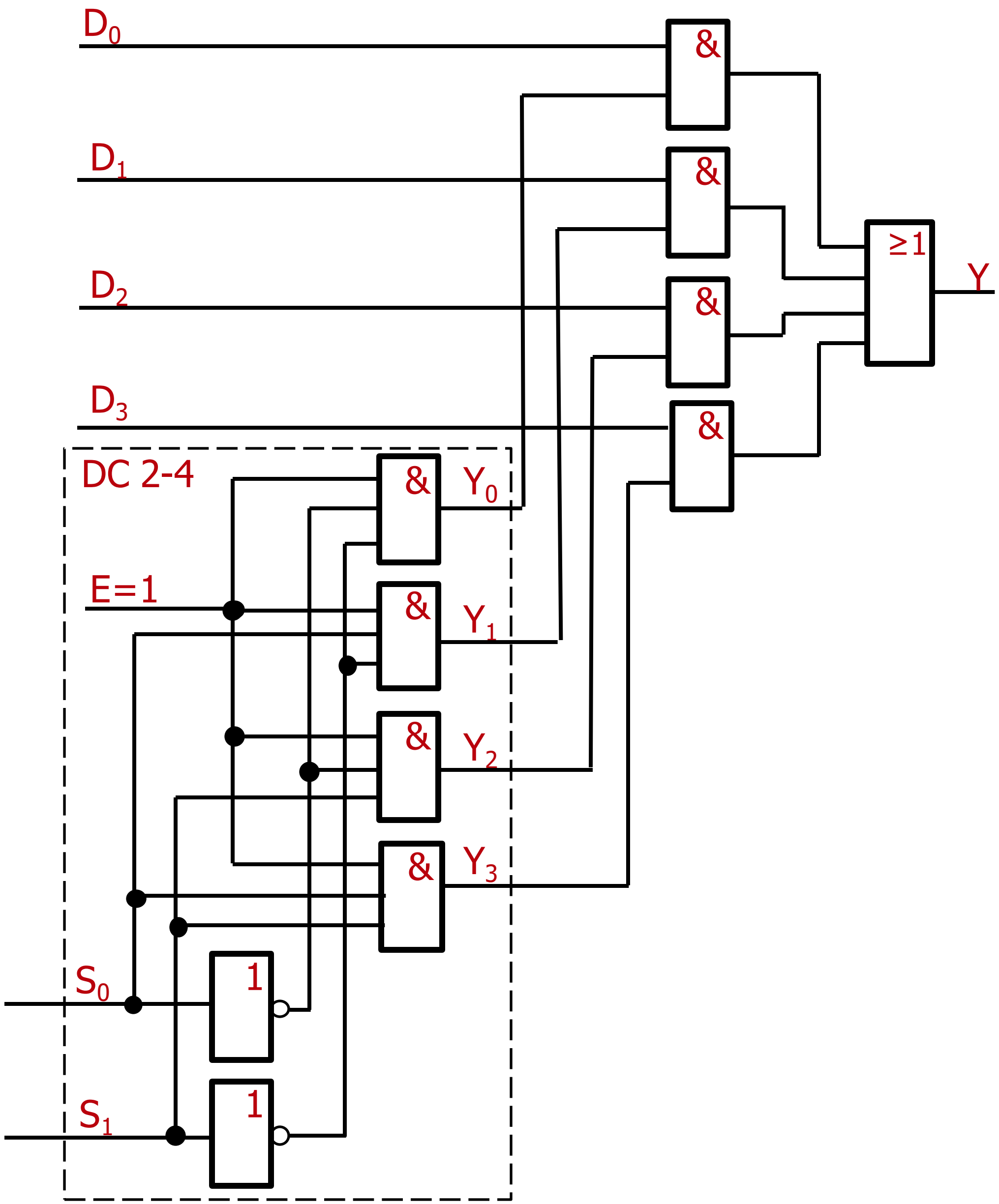
- Charakteristika
  - Kombinační log. síť, která přepíná log. úrovně z více vstupů na jeden výstup
  - Značí se MX či MUX
  - Pomocí binární kombinace na řídicím vstupu S se vybere log. úroveň z příslušného vstupu D_i a kopíruje se (generuje se shodná log. úroveň) na výstup Y
  - Může přepínat i více bitů naráz
- Použití
  - Data selector
  - Generátor logických funkcí atd.

$$Y = \sum_{i=0}^{j} m_i \cdot D_i$$

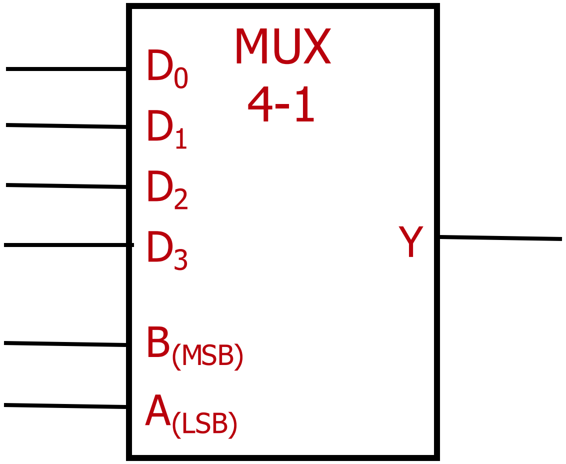


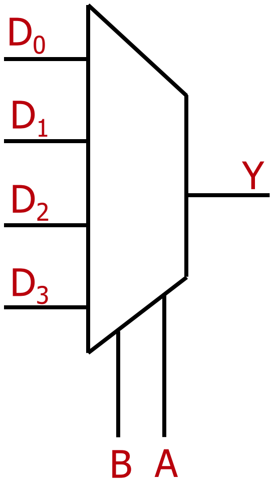
Multiplexor 4-1 (1 ze 4)

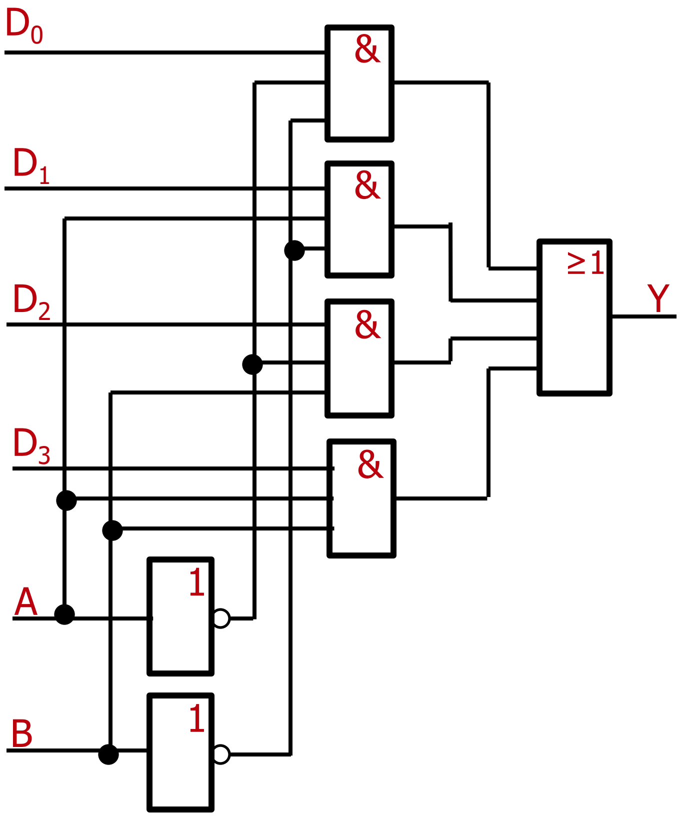
- Konstrukce pomocí dekodéru s povolovacím (datovým) vstupem E=1



- Logické značky, schéma, výraz
  - Pozor na pořadí (váhy) řídicích vstupů B (MSB), A (LSB)

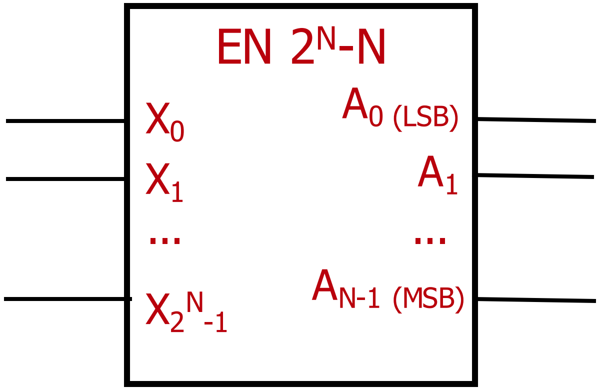






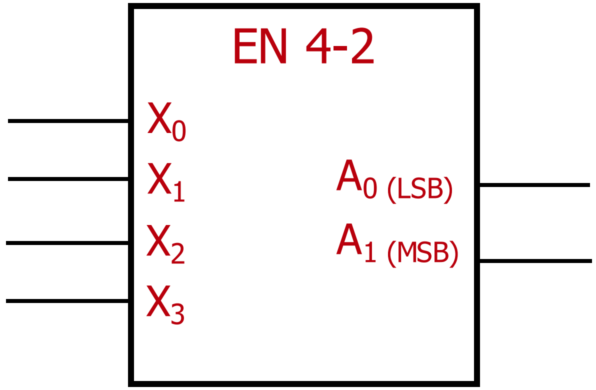
$$Y = \overline{B} \cdot \overline{A} \cdot D_0 + \overline{B} \cdot A \cdot D_1 + B \cdot \overline{A} \cdot D_2 + B \cdot A \cdot D_3$$

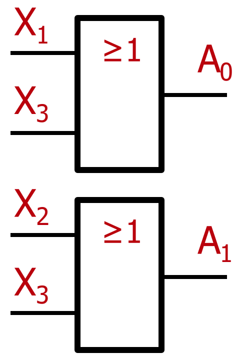
- Charakteristika
  - Kodér je kombinační log. síť, která převádí kód „1 z M”, kde $M=2^N$ na N bitový binární kód (číslo aktivního vstupu)
  - Má opačnou funkci k dekodéru
  - Značí se EN 2^N-N
  - Na výstupu je binární kód (číslo) aktivního vstupu (právě jen jeden může být aktivní v daném čase)
  - Pokud nelze zajistit, aby byl vždy jen jeden vstup aktivní, je třeba použít tzv. prioritní kodér (viz dále)
- Použití
  - Převody kódů
  - Kódování
  - Stanovení priority



- Příklad
  - Funkce je opačná k dekodéru 2-4
  - Výstupem je binární číslo odpovídající váze (číslu) aktivního vstupu
  - X - (don't care): hodnota není dle definice kodéru platná
  - Dle definice může být platná právě jen jedna proměnná – neurčené hodnoty nemohou nastat
  - Pokud může být více vstupů aktivních, je třeba tuto situaci ošetřit (detekovat – viz dále)

Vstupy				**Výstupy**	
X_3	X_2	X_1	X_0	A_1	A_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	X	X
0	1	0	0	1	0
0	1	0	1	X	X
0	1	1	0	X	X
0	1	1	1	X	X
1	0	0	0	1	1
1	0	0	1	X	X
1	0	1	0	X	X
1	0	1	1	X	X
1	1	0	0	X	X
1	1	0	1	X	X
1	1	1	0	X	X
1	1	1	1	X	X

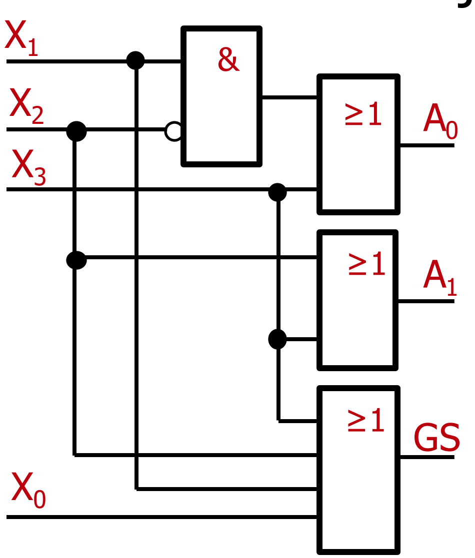


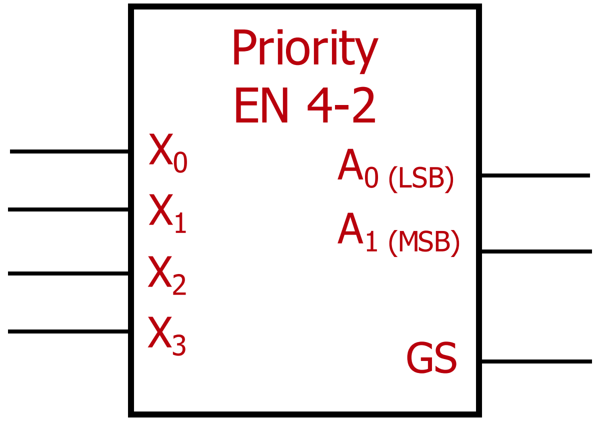


Prioritní kodér (anglicky Priority Encoder) 4-2

- Výstupní binární číslo určuje, který ze vstupních signálů s nejvyšší prioritou je aktivní
  - Pokud je více vstupů aktivních, výstupem bude binární číslo toho, který má nejvyšší prioritu
  - A1, A0 – číslo aktivního vstupu s nejvyšší prioritou
  - GS – detekce - jeden či více vstupů je aktivní

Vstupy				**Výstupy**		
x_3	x_2	x_1	x_0	A_1	A_0	GS
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1





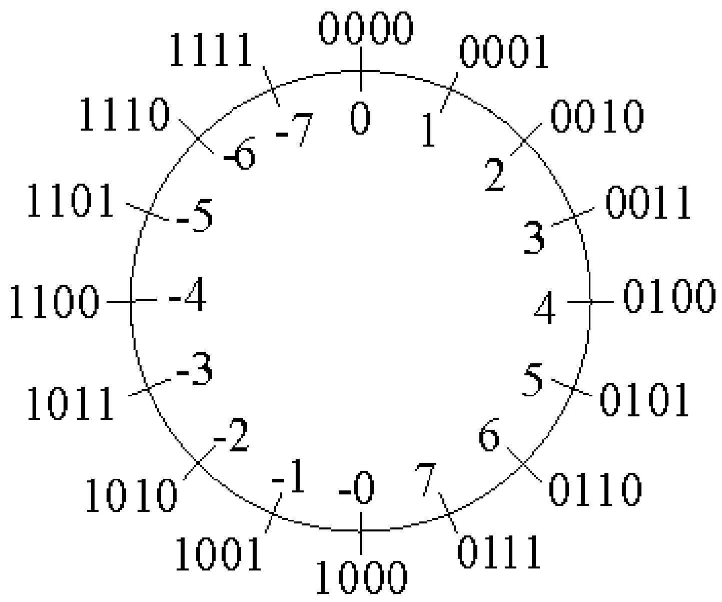
- Záporná čísla
  - Pro vyjádření čísel záporných je třeba definovat, jak bude příslušné znaménko určeno
  - Číslicový systém si může informaci o znaménku daného čísla pamatovat obecně různým způsobem
  - Typicky se však dodržuje konvence - znaménku se vyhradí jeden bit (nejčastěji MSB) příslušného binárního čísla, které má celkem *k* bitů, viz:

MSB							LSB
s	n_{k-2}	n_{k-3}	•	•	•	n_1	n_0

  - Čísla kladná mají ve znaménkovém bitu hodnou 0
  - Čísla záporná mají ve znaménkovém bitu hodnotu 1

Přirozený binární kód se znaménkem

- Přirozený (přímý) binární kód
  - Hodnota binárního čísla odpovídá převedené hodnotě čísla dekadického
  - Pokud chceme vyjádřit číslo záporné, prostě jej na místě MSB doplníme znaménkem
- Příklad
  - Binární číslo se znaménkem reprezentované v přirozeném kódu na čtyřech bitech
- Kód má dvě nuly – kladnou a zápornou
  - Tuto situaci je třeba vhodně ošetřit, což komplikuje počítání v tomto kódu



Doplňkové kódy

- Doplněk číselné soustavy o základu *r*
  - Anglicky radix-complement
  - Platí: $[N]_r = r^n - (N)_r$
  - Rozsah zobrazených čísel: $-r^{n-1} .. r^{n-1} - 1$

- Doplněk číselné soustavy o základu *r* snížený o 1
  - Anglicky diminished radix-complement
  - Doplněk lze též vyjádřit následovně:

$$[N]_r = r^n - (N)_r = ((r^n - 1) - (N)_r) + 1$$

  - Kde $(r^n - 1) - (N)_r$ je doplněk snížený o 1
  - Tento krok se často používá jako mezioperace při výpočtu doplňku přičtením jedničky

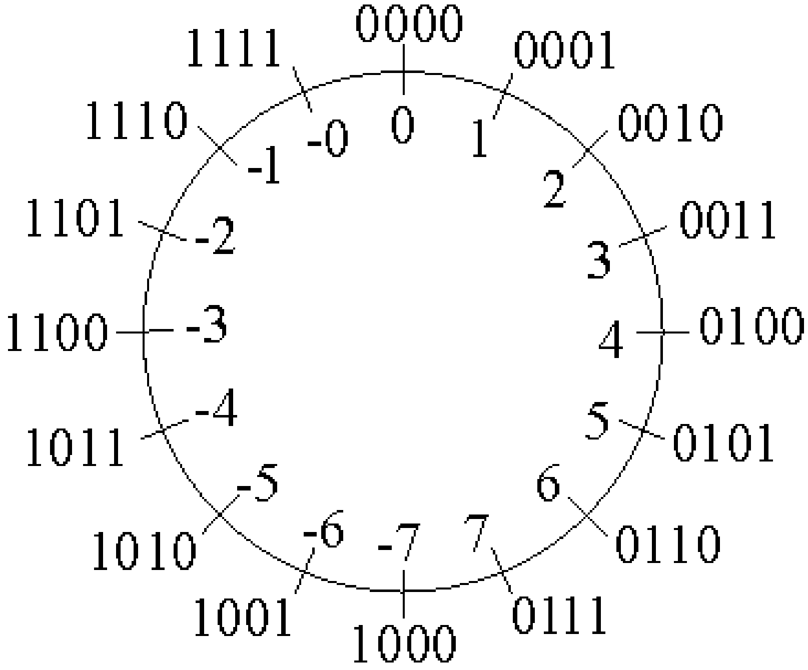
Jedničkový a dvojkový doplněk

- Jedničkový doplněk (anglicky one's complement)
  - Doplněk číselné soustavy o základu r = 2 snížený o 1
  - Lze spočítat prostou negací jednotlivých bitů daného čísla (komplement 1 = 0 a naopak) = odečítání jedničky resp. nuly od jedničky
- Dvojkový doplněk (anglicky two's complement)
  - Doplněk číselné soustavy o základu r = 2
  - Lze spočítat spočtením jedničkového doplňku daného čísla a přičtením jedničky
- Příklad:

$$
\begin{aligned}
[0101]_2 &= ((2^4 - 1) - 0101) + 1 \\
&= ((10000 - 1) - 0101) + 1 \\
&= (1111 - 0101) + 1 \\
&= ((1-0)\cdot 2^3 + (1-1)\cdot 2^2 + (1-0)\cdot 2^1 + (1-1)\cdot 2^0) + 1 \\
&= (1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0) + 1 \\
&= (1010) + 1 \\
&= 1011
\end{aligned}
$$

Jedničkový doplněk

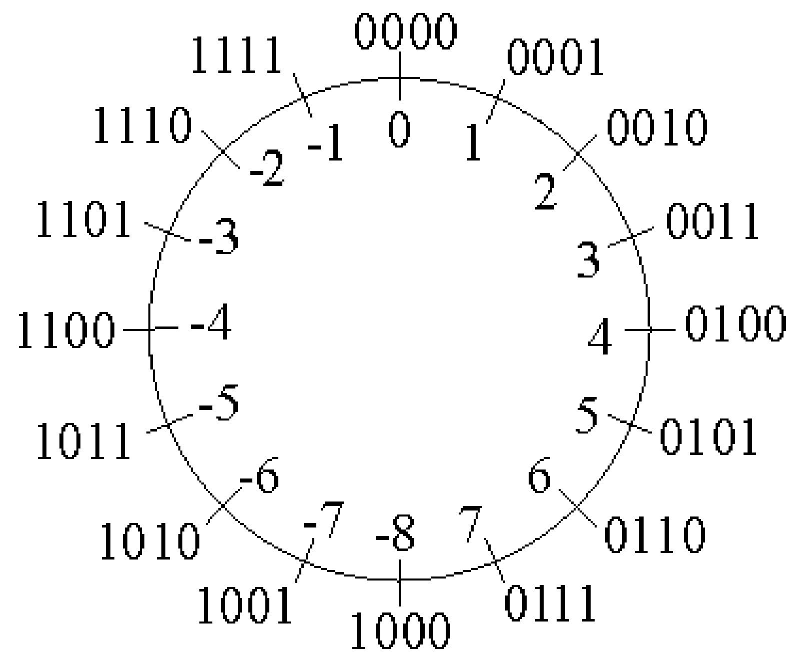
- Zobrazení čtyřbitových čísel
  - Kladná i záporná nula
  - Používá se převážně jako mezioperace při počítání dvojkového doplňku
- Příklad
  - Binární číslo se znaménkem reprezentované v jedničkovém doplňku na čtyřech bitech



Dvojkový doplněk

- Příklad - zobrazení čtyřbitových čísel
  - Nemá dvě nuly
  - Kód je nesymetrický – má o jednu více záporných hodnot než kladných (na sudém počtu čísel nelze zobrazit lichý počet hodnot – čísla kladná, záporná a nula)
  - Nesymetričnost není na závadu při provádění většiny aritmetických operací (kromě speciálního případu násobení dvou nejzápornějších čísel)
  - Lze relativně snadno realizovat logickými obvody
  - Představuje základ aritmetiky většiny číslicových systémů

- Příklad
  - Binární číslo se znaménkem reprezentované ve dvojkovém doplňku na čtyřech bitech



- Aritmetické operace
  - Sčítání, odčítání, násobení a dělení
- Příklad - tabulka součtů dekadických čísel (složité)

+	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

- Příklad - tabulka násobení dekadických čísel (složité)

x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

Sčítání a násobení binárních čísel

- V případě binárních čísel
  - Jsou definiční tabulky mnohem jednodušší, než pro jiné soustavy

  - Sčítání

+	**0**	**1**
0	0	1
1	1	(1)0

Násobení

x	**0**	**1**
0	0	0
1	0	1

- Součet dvou jedniček
  - Dává hodnotu dva
  - V případě dekadických čísel musíme, v případě výsledku většího než devět, provádět tzv. přenos (anglicky Carry)
  - U binárních čísel je to stejné, hodnota (1)0 znamená výsledek nula s přenosem do vyššího řádu (do vyššího významového bitu)

Příklad součtu dvou osmibitových čísel

$$
\begin{array}{ccccccccc}
1 & 1 & 1 & & 1 & & & & \\
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

- Popis
  - V horním řádku jsou znázorněny přenosy z nižších řádů
  - Pod čarou je výsledný součet
  - Výsledek má celkem 9 bitů

$$(10101010)_2 = (170)_{10}$$

$$(11101100)_2 = (236)_{10}$$

- Zkouška

$$(110010110)_2 = (406)_{10} = (170 + 236)_{10}$$

Sčítání binárních čísel se znaménkem

- Existují čtyři možnosti součtu dvou čísel se znaménkem
  - (+) + (+)
  - (-) + (+)
  - (+) + (-)
  - (-) + (-)
- Platí
  - Pokud máme k dispozici libovolný počet bitů, nemusíme tyto případy rozlišovat
  - Pokud máme k dispozici pevný počet bitů, musíme sledovat, zda u výsledku nedošlo k tzv. přetečení (přeplnění) rozsahu (anglicky overflow)
  - Přetečení může nastat tehdy, jestliže nelze výsledek správně zobrazit na daném počtu bitů

Příklad součtu dvou čtyřbitových čísla

- Mějme:
  - Dvě kladná binární čísla v přirozeném kódu

$$0011 = (3)_{10}$$

$$0101 = (5)_{10}$$

- Proveďme:
  - Převod na čísla záporná spočtením dvojkového doplňku
  - Hranaté závorky značí reprezentaci ve dvojkovém doplňku
- Máme tedy celkem čtyři čísla:
  - Dvě kladná (MSB=0)
  - Dvě záporná (MSB=1)

$$[0011]_2 = 1101 = (-3)_{10}$$

$$[0101]_2 = 1011 = (-5)_{10}$$

- Platí:
  - U záporných musíme dávat pozor, jak s nimi pracujeme

Součet dvou kladných čísel

$(+3)+(+3)$ $(+5)+(+5)$

$$\begin{array}{c|c:ccc} & 0 & 0 & 1 & 1 \\ + & 0 & 0 & 1 & 1 \\ \hline & 0 & 1 & 1 & 0 \end{array} \qquad \begin{array}{c|c:ccc} & 0 & 1 & 0 & 1 \\ + & 0 & 1 & 0 & 1 \\ \hline & 1 & 0 & 1 & 0 \end{array}$$

- Popis
  - Svislou čarou je naznačena skutečnost, že máme k dispozici jen 4 bity
  - Čárkovanou čarou je oddělen znaménkový bit

$$(+3)+(+3)=(+6)$$

- Výsledek
  - V prvním případě nám vyšel správný výsledek
  - Ve druhém případě nám při sčítání dvou kladných čísel vyšlo číslo záporné –6 (MSB = 1)

$$[1010]_2 = 0110 = (6)_{10}$$

$$(+5)+(+5)=(+10)$$

- Zkouška
  - Výsledek převeďme na kladné číslo nalezením dvojkového doplňku
  - Zapamatujme si znaménko původního výsledku
  - Doplníme k výsledku převodu
  - Výsledek je nesprávný
    - Došlo k přetečení - výsledek je mimo rozsah zobrazení čtyřbitových čísel ve dvojkovém doplňku $(-8)..(+7)$

Součet dvou záporných čísel

$$(-3)+(-3) \qquad (-5)+(-5)$$

$$\begin{array}{r|c:ccc} & 1 & 1 & 0 & 1 \\ + & 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 \end{array} \qquad \begin{array}{r|c:ccc} & 1 & 0 & 1 & 1 \\ + & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 \end{array}$$

- Popis
  - Při výpočtu sumy zahazujeme případný přenos mimo zobrazení (carry nás nyní nezajímá – nelze zobrazit)
- Výsledek
  - V prvním případě nám vyšel správný výsledek $(-3)+(-3)=(-6)$
  - Ve druhém případě vidíme, že při sčítání dvou záporných čísel vyšlo číslo kladné +6 => došlo k přetečení $(-5)+(-5)=(-10)$
  - Výsledek je mimo rozsah zobrazení čtyřbitových čísel reprezentovaných ve dvojkovém doplňku $(-8)..(+7)$

Součet čísla kladného se záporným

$$(+5) + (-3)$$

$$\begin{array}{c|c:ccc} & 0 & 1 & 0 & 1 \\ + & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 \end{array}$$

$$(+3) + (-5)$$

$$\begin{array}{c|c:ccc} & 0 & 0 & 1 & 1 \\ + & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 0 \end{array}$$

- Popis
  - V prvním případě nám vyšel správný výsledek $(+5) + (-3) = (+2)$
  - Ve druhém si pro kontrolu převeďme výsledek na číslo kladné $[1110]_2 = 0010 = (2)_{10}$
  - Vidíme, že i druhý výsledek je správný $(+3) + (-5) = (-2)$

Součet čísla záporného s kladným

$$(-5)+(+3)$$

$$\begin{array}{c|c:ccc}
 & 1 & 0 & 1 & 1 \\
+ & 0 & 0 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 0
\end{array}$$

$$(-5)+(+3)=(-2)$$

$$(-3)+(+5)$$

$$\begin{array}{c|c:ccc}
 & 1 & 1 & 0 & 1 \\
+ & 0 & 1 & 0 & 1 \\
\hline
1 & 0 & 0 & 1 & 0
\end{array}$$

$$(-3)+(+5)=(+2)$$

- V obou případech nám opět vyšly správné výsledky

Sčítání čísel ve dvojkovém doplňku: Shrnutí

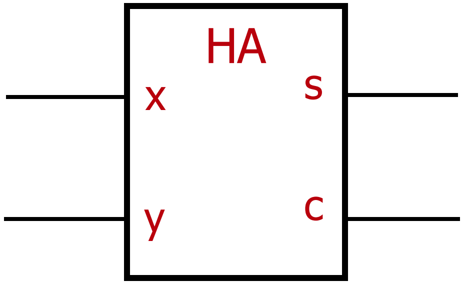
Součet	Znaménko	Přenos	Přetečení	Podmínka
$(+X)+(+Y)$	0 1	0 0	0 1	$(X+Y) \leq 2^{n-1}-1$ $(X+Y) > 2^{n-1}-1$
$(-X)+(-Y)$	1 0	1 1	0 1	$-(X+Y) \geq -2^{n-1}$ $-(X+Y) < -2^{n-1}$
$(+X)+(-Y)$ $(-X)+(+Y)$	0 1	1 0	0 0	$X \leq Y$ $X > Y$

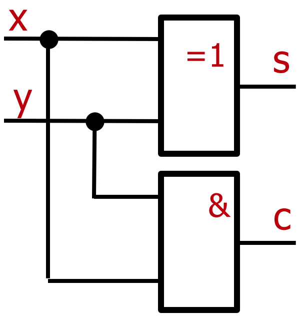
- Při operacích typu (+) + (+) a (-) + (-) může dojít k přetečení v případě, že se výsledek dostane mimo rozsah zobrazení daný počtem platných bitů
- Při operacích typu (-) + (+) a (+) + (-) k přetečení dojít nemůže
- Poznámka
  - Ve většině počítačů se přetečení detekuje příslušným logickým obvodem a uschovává se. V případě, že se přetečení nedetekuje, je na programátorovi, aby zkontroloval, zda nedošlo k přetečení (porovnáním znamének operandů a výsledku)

Binární sčítačka poloviční

- Poloviční sčítačka (Half Adder - HA)
  - Nemá vstup carry (přenos z nižšího řádu)
  - Výraz pro sumu
    $$s = \overline{x} \cdot y + x \cdot \overline{y} = y \oplus x$$
  - Výraz pro přenos do vyššího řádu carry $c = x \cdot y$
  - Pravdivostní tabulka
  - Logická značka
  - Logické schéma

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



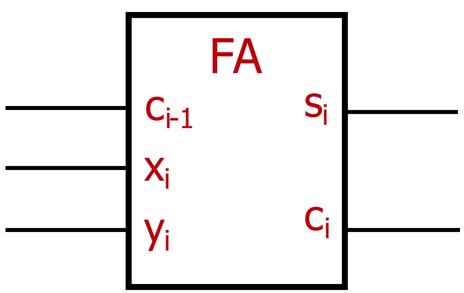


Binární sčítačka úplná

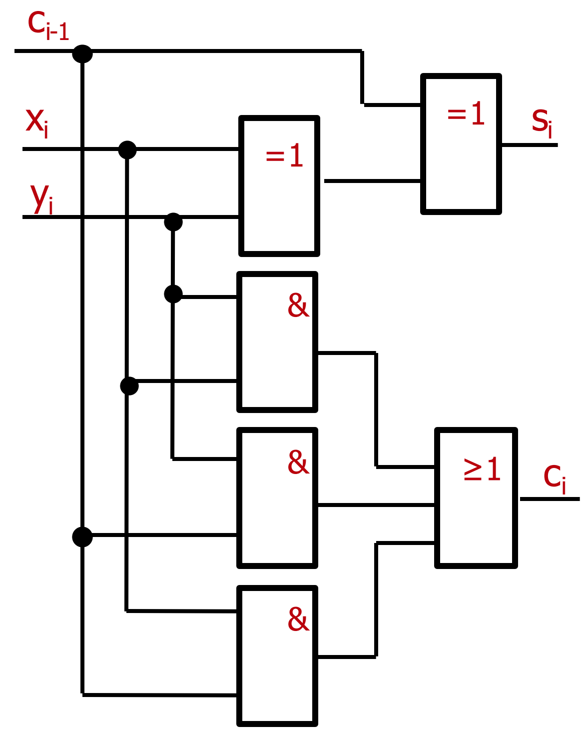
- Úplná sčítačka (Full Adder - FA)
  - Vstup c_{i-1} (carry in - přenos z nižšího řádu)
  - Výraz pro sumu
  - Výraz pro c_i (carry out - přenos do vyššího řádu)

$$s_i = \bar{y}_i \cdot \bar{x}_i \cdot c_{i-1} + \bar{y}_i \cdot x_i \cdot \bar{c}_{i-1} + y_i \cdot \bar{x}_i \cdot \bar{c}_{i-1} + y_i \cdot x_i \cdot c_{i-1}$$
$$= \bar{y}_i \cdot (\bar{x}_i \cdot c_{i-1} + x_i \cdot \bar{c}_{i-1}) + y_i \cdot (\bar{x}_i \cdot \bar{c}_{i-1} + x_i \cdot c_{i-1})$$
$$= \bar{y}_i \cdot (x_i \oplus c_{i-1}) + y_i \cdot (\overline{x_i \oplus c_{i-1}})$$
$$= y_i \oplus x_i \oplus c_{i-1}$$

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

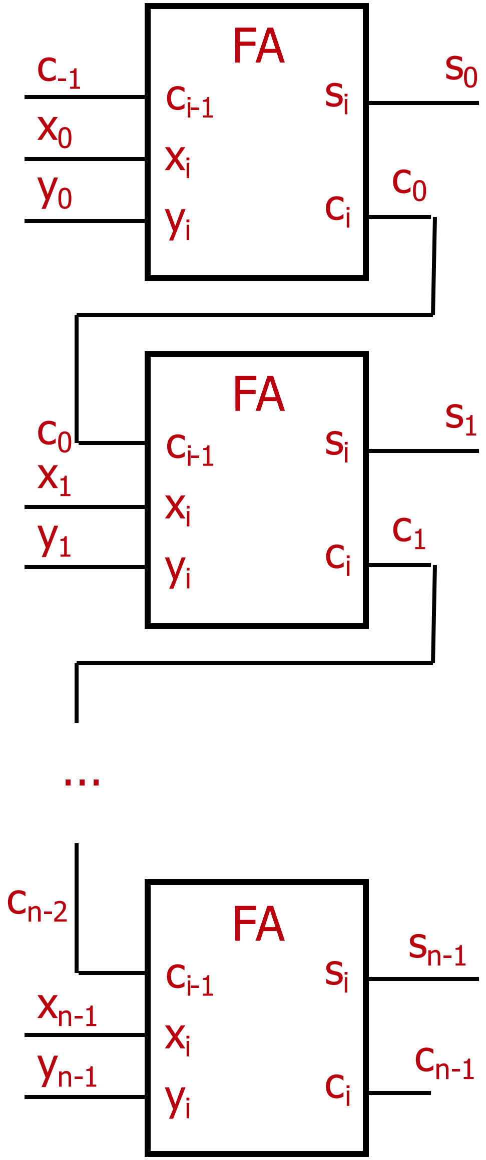


c_{i-1}	x_i	y_i	s_i	c_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



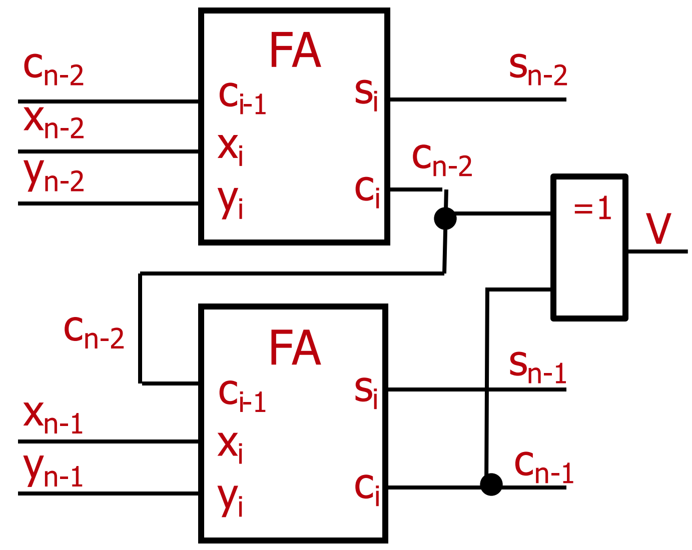
Binární sčítačka vícebitová

- Vícebitová sčítačka
  - Kaskádní zapojení jednobitových sčítaček
  - Anglicky „ripple carry” (přenos se šíří jednotlivými FA)
  - Carry musí procházet (propagate) přes všechny stupně sčítačky
  - Cenově výhodné řešení
  - Pomalé - zpoždění je úměrné počtu bitů
- Urychlení
  - Paralelní struktura
  - Se zrychleným přenosem (Carry Look Ahead) atd.



Detekce přetečení: Verze 1

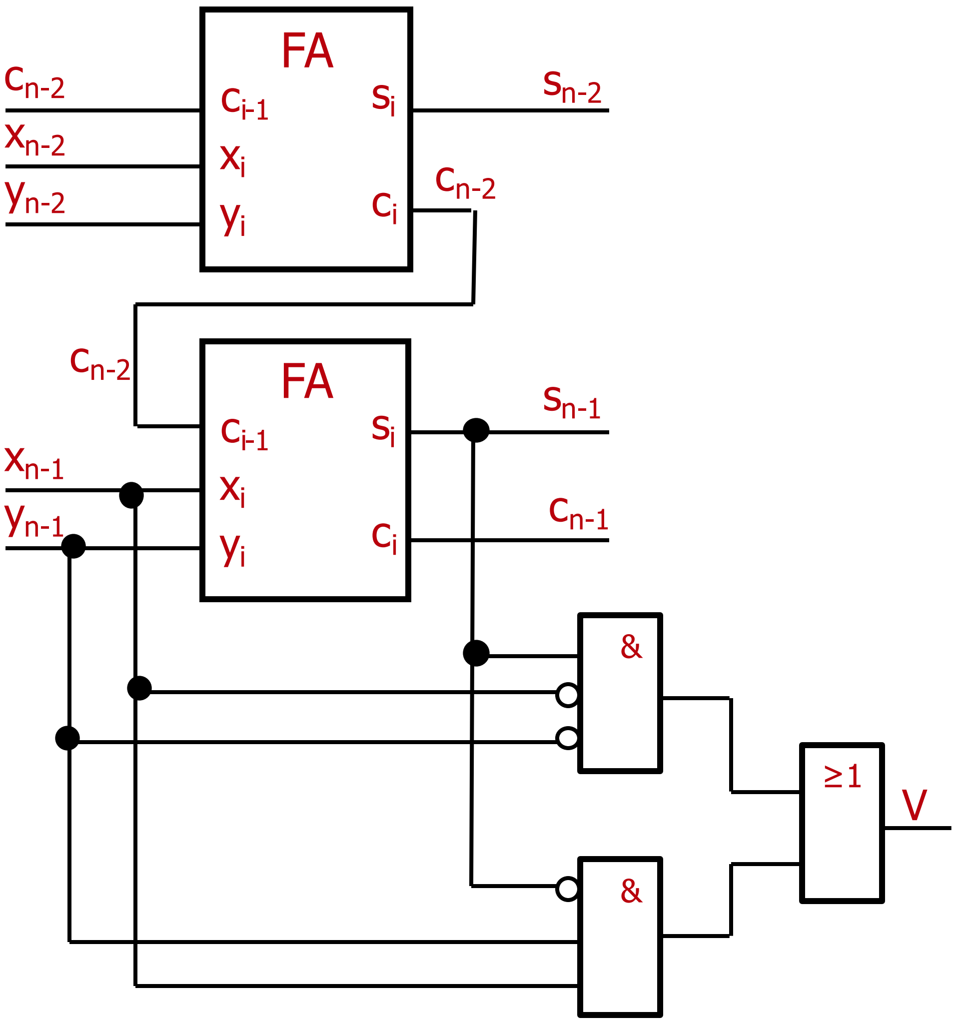
- Detekce přetečení (overflow) na základě shody přenosu do a ze znaménkového bitu
- Poznámka
  - Předpokládáme, že čísla jsou reprezentována ve dvojkovém doplňku
- Tabulka shrnující stav po provedení součtu dvou operandů



Součet	Znaménko	Přenos	Přetečení	Podmínka
$(+X)+(+Y)$	0 1	0 0	0 1	$(X+Y) \leq 2^{n-1}-1$ $(X+Y) > 2^{n-1}-1$
$(-X)+(-Y)$	1 0	1 1	0 1	$-(X+Y) \geq -2^{n-1}$ $-(X+Y) < -2^{n-1}$
$(+X)+(-Y)$ $(-X)+(+Y)$	0 1	1 0	0 0	$X \leq Y$ $X > Y$

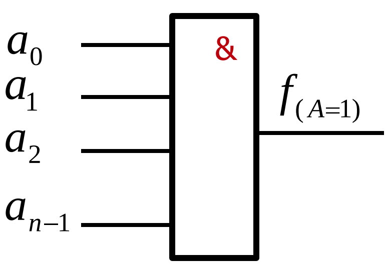
Detekce přetečení: Verze 2

- Detekce přetečení na základě porovnání znaménkových bitů sčítanců a výsledku součtu
  - Součet kladných, resp. záporných, operandů nemůže být záporný, resp. kladný
- Poznámka
  - Předpokládáme, že čísla jsou reprezentována ve dvojkovém doplňku



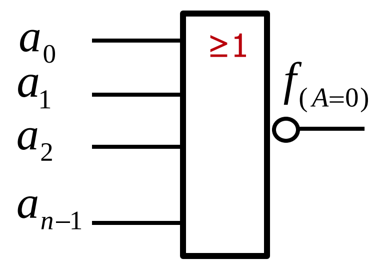
- Test na jedničkovou hodnotu
  - Pokud jsou všechny bity vstupního vektoru rovny log. 1, pak je výstup roven log. 1

$$f_{(A=1)} = a_0 \cdot \ldots \cdot a_{n-1}$$



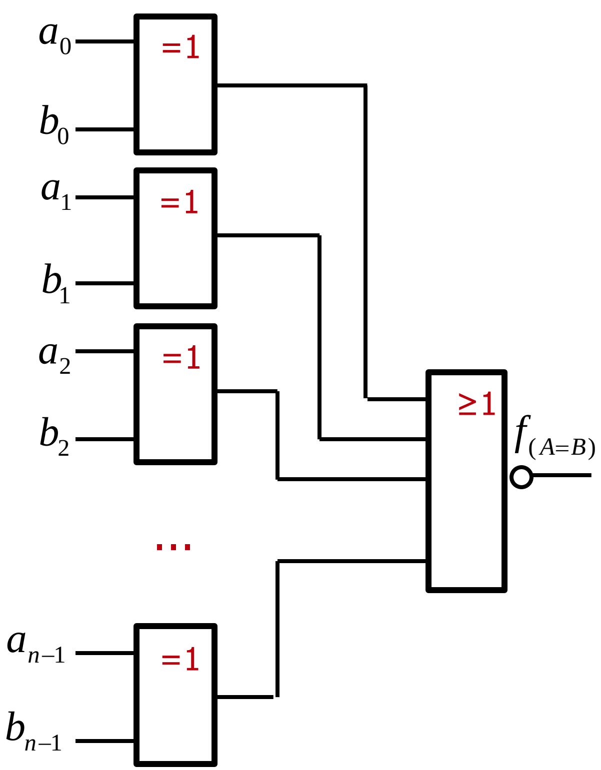
- Test na nulovou hodnotu
  - Pokud jsou všechny bity vstupního vektoru rovny log. 0, pak je výstup roven log. 1

$$f_{(A=0)} = \overline{a_0} \cdot \overline{a_1} \cdot \ldots \cdot \overline{a_{n-1}}$$
$$= \overline{a_0 + \ldots + a_{n-1}}$$

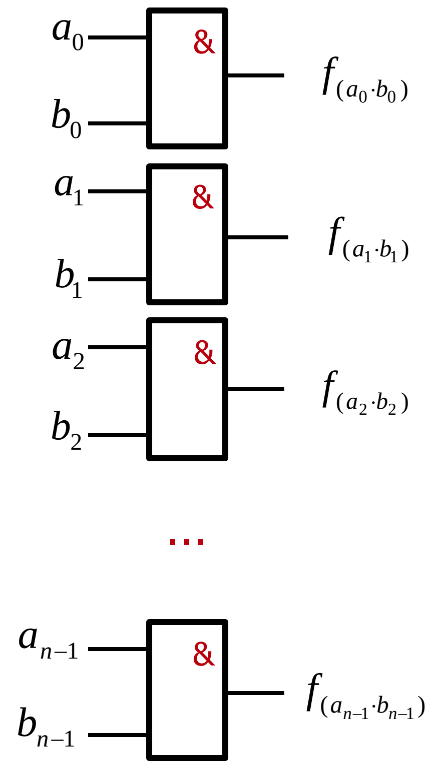


- Test na shodu
  - Porovnání dvou N-bitových vektorů

$$f_{(A=B)} = \overline{a_0 \oplus b_0 + \ldots + a_{n-1} \oplus b_{n-1}}$$



- N-bitové logické operace
  - Provádí se bit po bitu
  - Příklad N-bitový AND



- Relace
  - C – přenos (carry)
  - Z – nula (zero)
  - N – záporné (negative)
  - V – přetečení (overflow)

Relace	Bez znaménka	Se znaménkem
$A = B$	Z	Z
$A \neq B$	$\bar{Z}$	$\bar{Z}$
$A < B$	$\overline{C + Z}$	$\overline{(N \oplus V) + Z}$
$A > B$	$\bar{C}$	$(N \oplus V)$
$A \leq B$	C	$\overline{(N \oplus V)}$
$A \geq B$	$\bar{C} + Z$	$(N \oplus V) + Z$

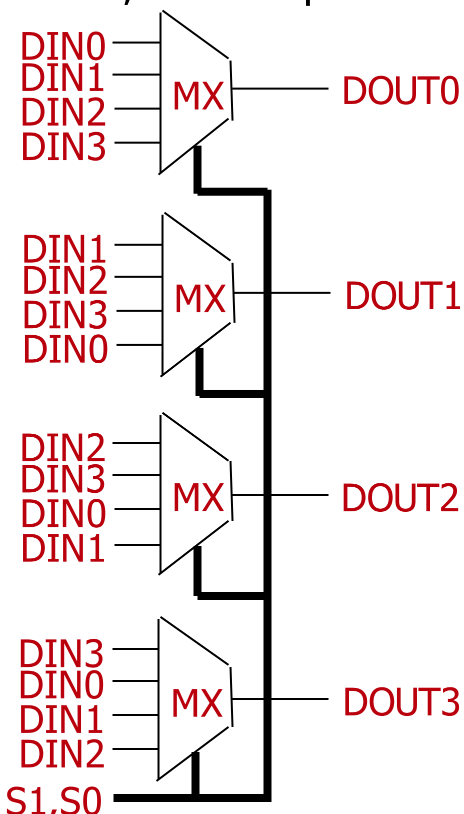
Aritmetické a logické operace

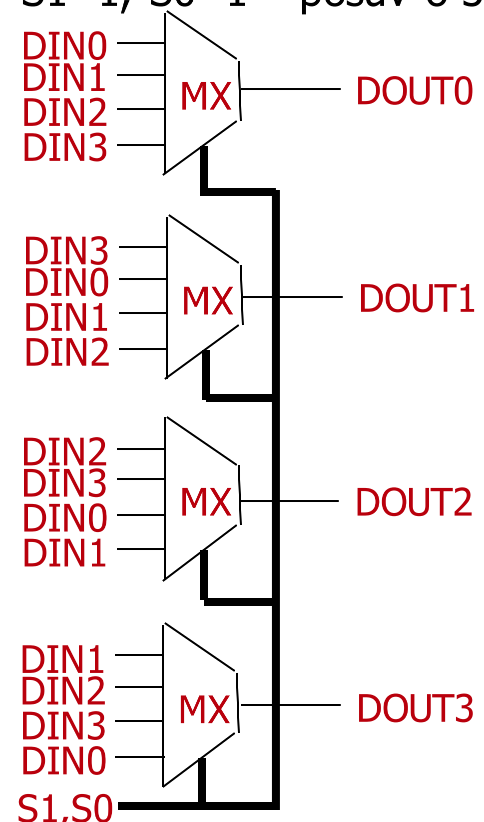
- Posuvy a rotace

- Logický posuv vlevo
  - SLL - Shift Left Logical
  - LSB←0
  - Např.: SLL(1011) = 0110
- Aritmetický posuv vlevo
  - SLA - Shift Left Arithmetic
  - LSB←0
  - Např.: SLA(1011) = 0110
- Rotace vlevo
  - ROL - Rotate Left
  - LSB ←MSB
  - Např.: ROL(1011) = 0111

- Logický posuv vpravo
  - SRL - Shift Right Logical
  - 0→MSB
  - Např.: SRL(1011) = 0101
- Aritmetický posuv vpravo
  - SRA - Shift Right Arithmetic
  - MSB→MSB (šíření znaménka)
  - Např.: SRA(1011)=1101
- Rotace vpravo
  - ROR - Rotate Right
  - LSB→MSB
  - Např.: ROR(1011) = 1101

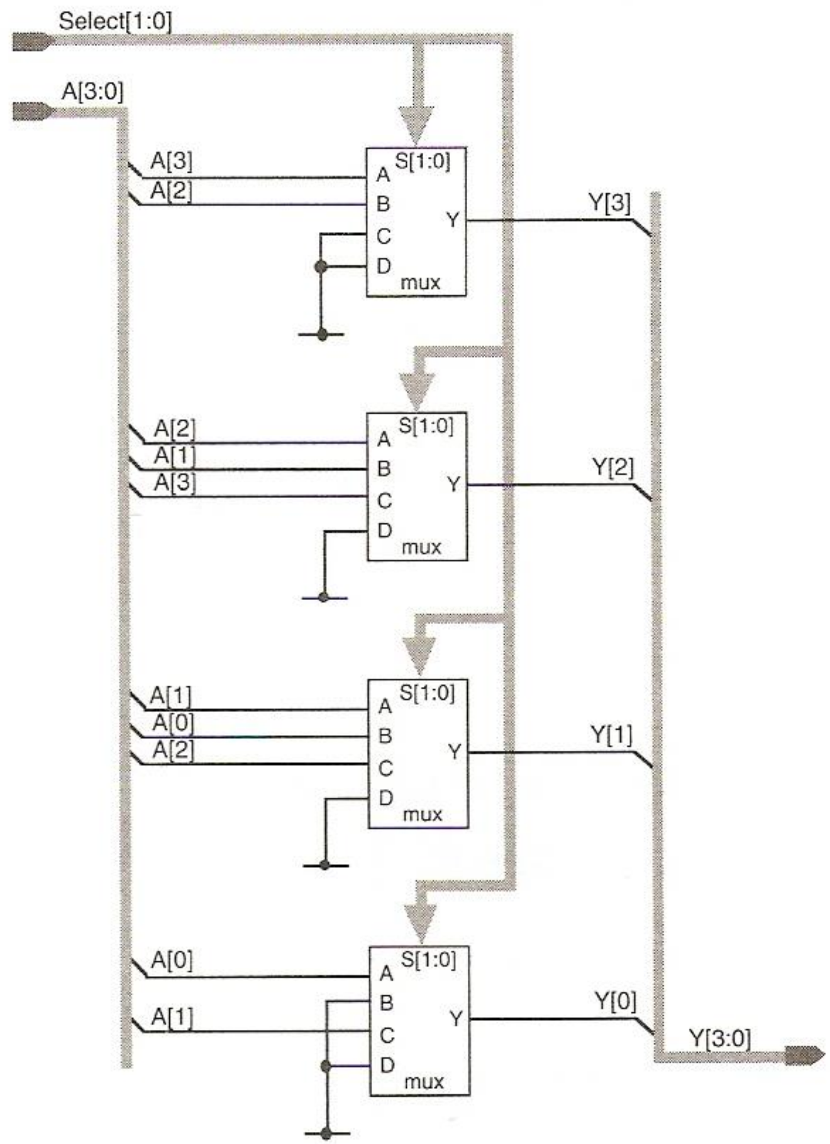
- Válcový posouvač - N-bitů v jednom kroku (barrel shifter)
- Rotace vpravo v jednom kroku
  - S1=0, S0=0 – posuv o 0 bitů vpravo
  - S1=0, S0=1 – posuv o 1 bit vpravo
  - S1=1, S0=0 – posuv o 2 bity vpravo
  - S1=1, S0=1 – posuv o 3 bity vpravo
- Rotace vlevo v jednom kroku
  - S1=0, S0=0 – posuv o 0 bitů vlevo
  - S1=0, S0=1 – posuv o 1 bit vlevo
  - S1=1, S0=0 – posuv o 2 bity vlevo
  - S1=1, S0=1 – posuv o 3 bity vlevo





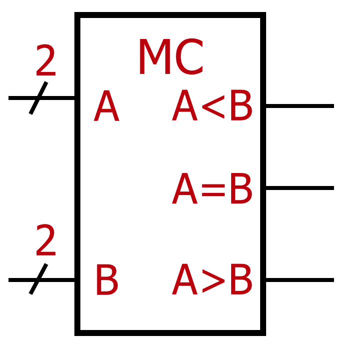
- Posouvač – 1 bit v jednom kroku
  - SLL - Shift Logical Left
  - SRL - Shift Logical Right

Select	Operace
00	Y←A
01	Y←SLL(A)
10	Y←SRL(A)
11	Y←0



[D.J Smith: „HDL Chip Design"]

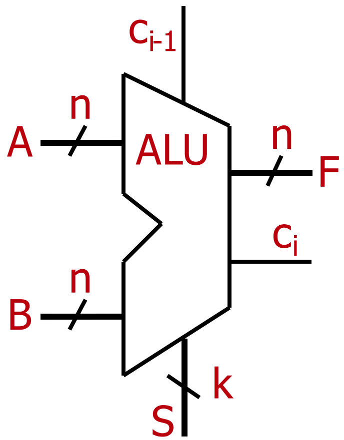
- Komparátor (magnitude comparator)
- Příklad 2bitového komparátoru
  - Porovnání hodnot vstupních dvoubitových operandů A a B
- Poznámka:
  - Porovnávat operandy lze též např. pomocí sčítačky přičtením záporného operandu a testováním hodnoty výsledku:
  - Výsledek operace (A-B)=0 => A=B
  - Znaménko výsledku operace (A-B)=1 => $A<B$



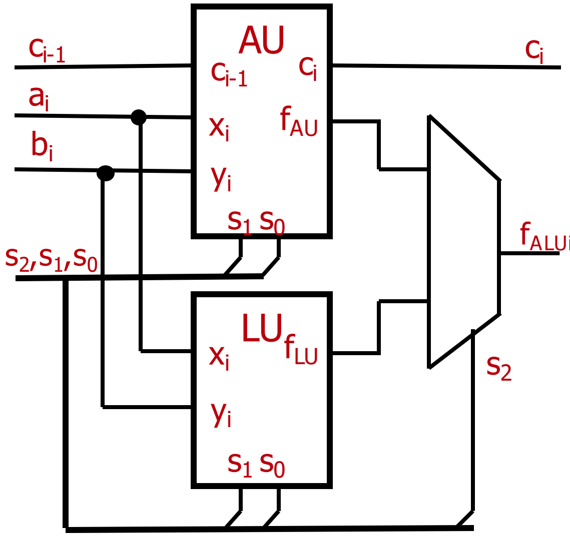
a_1	a_0	b_1	b_0	$A<B$	$A=B$	$A>B$
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

- Arithmetic Logic Unit
  - Základní stavební komponenta počítačů
- Příklad návrhu
  - Dva N-bitové operandy A a B
  - K-bitový vektor S, který vybírá operaci nad operandy A a B
- Funkční tabulka
- Logická značka

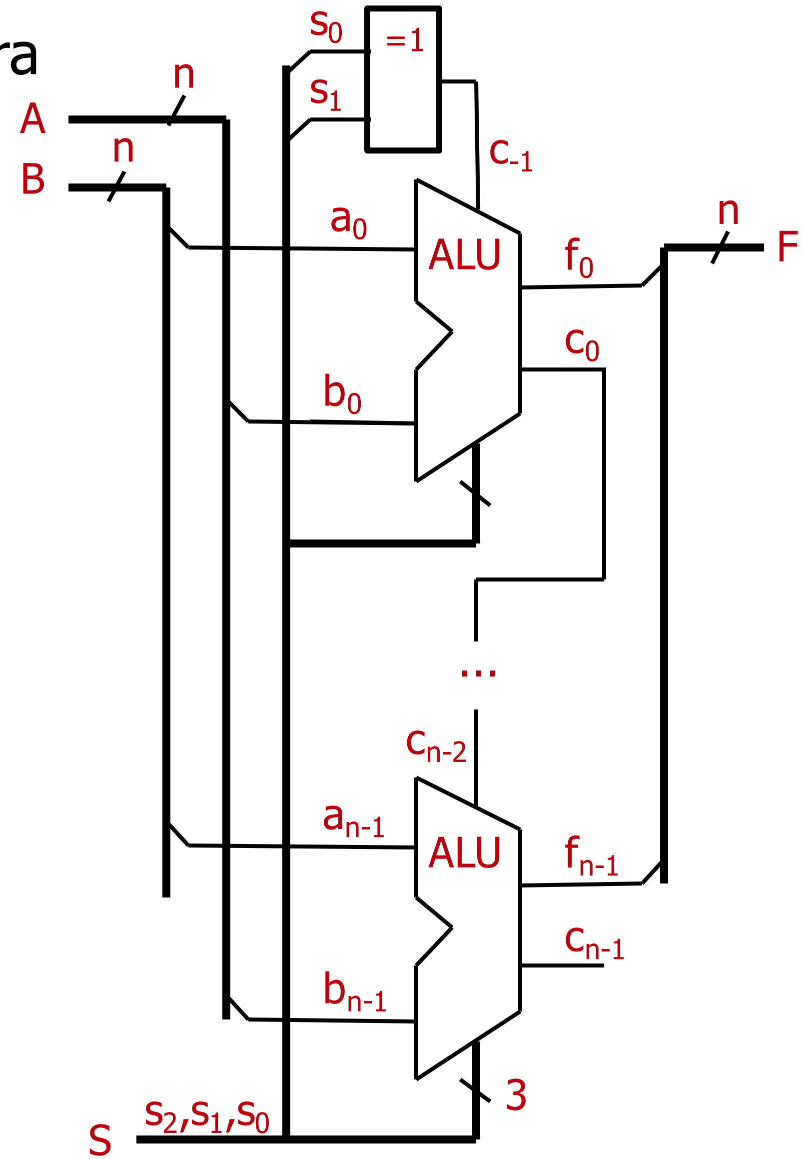
s_2	s_1	s_0	Funkce	Popis
0	0	0	F=A+B	Součet (Add)
0	0	1	F=A-B	Rozdíl (Subtract)
0	1	0	F=A+1	Přičtení jedničky (Increment)
0	1	1	F=A-1	Odečtení jedničky (Decrement)
1	0	0	F=A AND B	Logický součin
1	0	1	F=A OR B	Logický součet
1	1	0	F=NOT(A)	Negace
1	1	1	F=A XOR B	Nonekvivalence



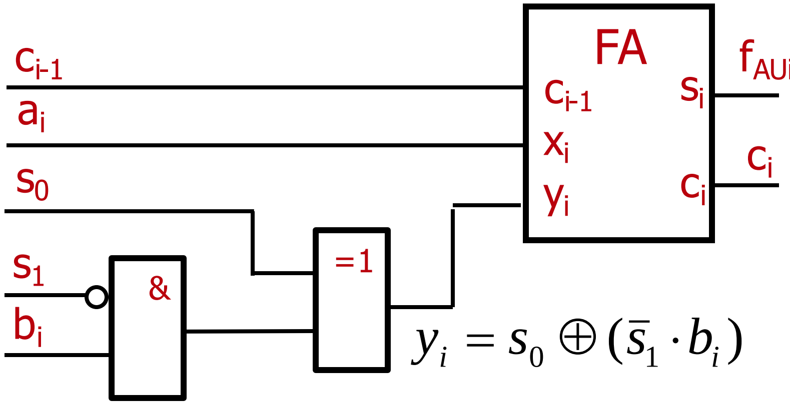
- Při návrhu budeme postupovat shora dolů
  - N-bitovou ALU sestavíme z N 1bitových ALU
  - AU – aritmetická jednotka
  - AL – logická jednotka



  - Poznámka: Carry in do nejnižšího řádu ALU (c_{-1}) se generuje pomocí log. členu XOR, viz následující slajd



- Jednobitová aritm. jednotka. (AU)
  - Odčítání = přičítání dvojkového doplňku (Subtract, Decrement)
  - Dvojkový doplněk = jedničkový doplněk (negace všech bitů) + 1
  - Bitová negace se realizuje členem XOR
  - Přičítání +1 = carry in (c_{-1}=1)



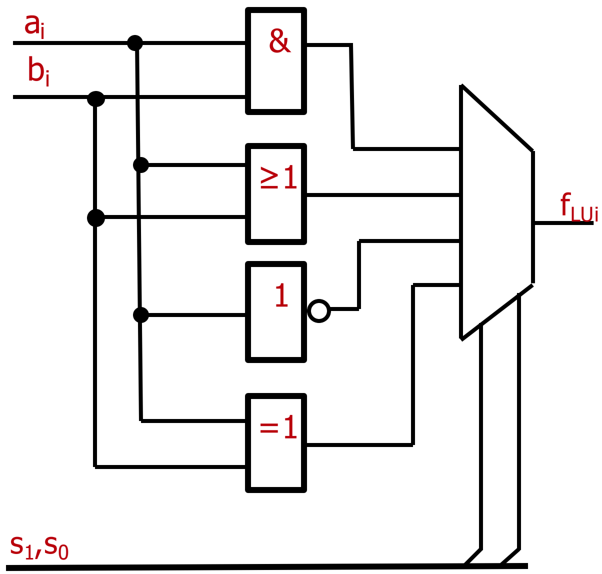
$$y_i = s_0 \oplus (\bar{s}_1 \cdot b_i)$$

$$F = A - B = A + [B]_2$$
$$= A + (\bar{b}_{n-1} \ldots \bar{b}_0) + 1$$

$$F = A - 1 = A + (-1)$$
$$= A + [0 \ldots 1]_2 = A + (1 \ldots 1) + 0$$

s_1	s_0	b_i	y_i	c_{-1}	Funkce	Popis
0 0	0 0	0 1	0=b_i 1=b_i	0 0	F=A+B	Add (A+B+0)
0 0	1 1	0 1	1=b'_i 0=b'_i	1 1	F=A+$[B]_2$	Subtract (A-B=A + dvojkový doplněk B =A + (not B) + 1)
1 1	0 0	0 1	0 0	1 1	F=A+1	Increment (A+0+1)
1 1	1 1	0 1	1 1	0 0	F=A+$[1]_2$	Decrement (A-1=A + dvojkový doplněk (+1) =A+ (not 1 + 1) + 0)

- Jednobitová logická jednotka (LU)
  - Pravdivostní tabulka
  - Logické schéma s multiplexorem



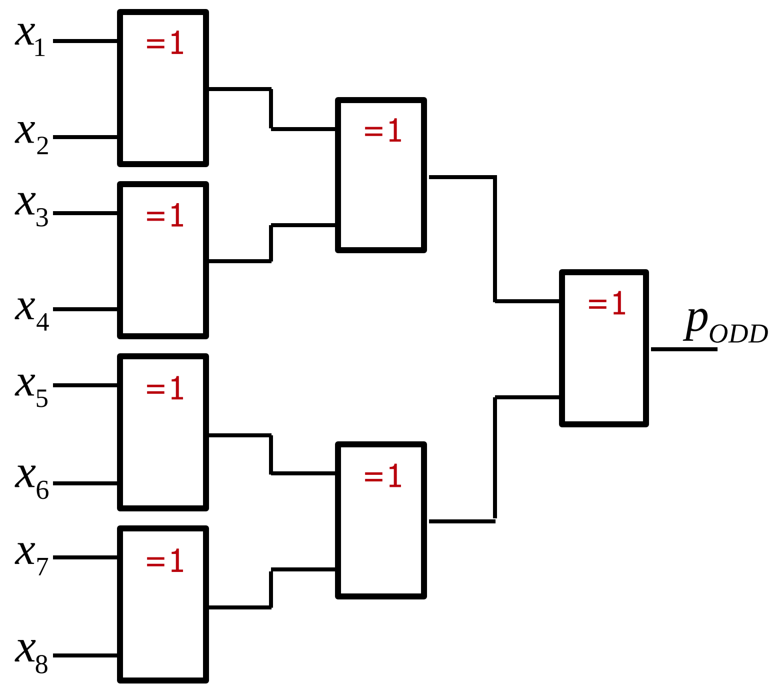
s_1	s_0	a_i	b_i	f_{LUi}	Funkce
0	0	0	0	0	F=A AND B
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	1	
0	1	0	0	0	F=A OR B
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	1	
1	0	0	0	1	F=NOT(A)
1	0	0	1	1	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	F=A XOR B
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	0	

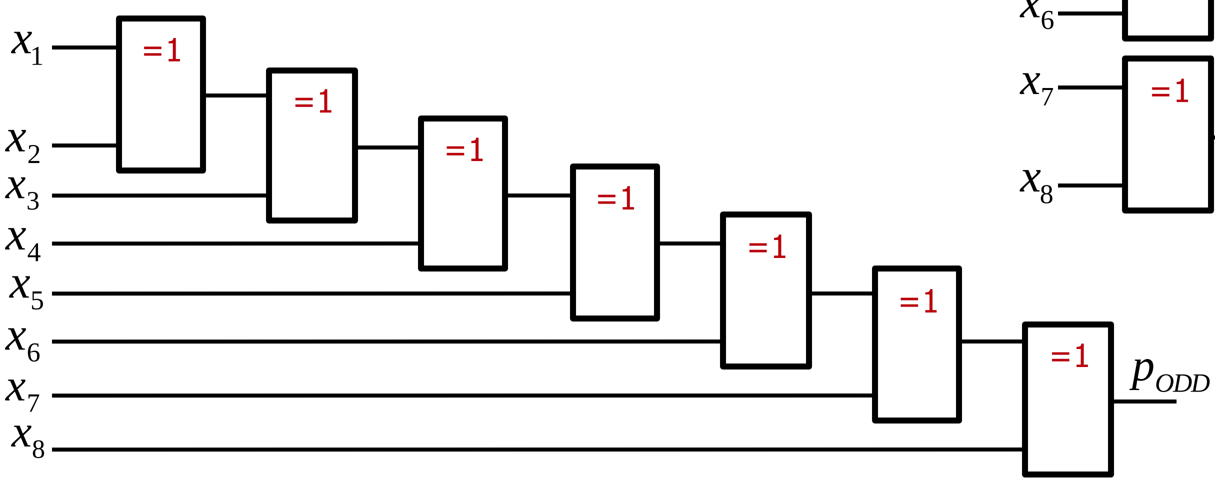
Zabezpečení informace: Parita

- Určuje zda sudý, resp. lichý počet vstupních proměnných nabývá hodnoty log. 1
  - Sudá - počet jedniček ve vstupním vektoru je sudé číslo
  - Lichá - počet jedniček ve vstupním vektoru je liché číslo
- Použití
  - Zabezpečení informace
- Příklad
  - Zabezpečení BCD kódu sudou paritou

#	BCD				Sudá parita
0	0	0	0	0	**1**
1	0	0	0	1	**0**
2	0	0	1	0	**0**
3	0	0	1	1	**1**
4	0	1	0	0	**0**
5	0	1	0	1	**1**
6	0	1	1	0	**1**
7	0	1	1	1	**0**
8	1	0	0	0	**0**
9	1	0	0	1	**1**

- Příklad generování liché parity
  - Anglicky „odd” parity
  - Stromovou strukturou
  - Kaskádní strukturou
- Poznámka
  - Sudá (anglicky „even”) parita je negací liché





$$p_{ODD}(x_1,\ldots,x_8) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8$$