Prvky elektronických počítačů

Logické obvody a systémy

texty pro distanční studium

Doc. Ing. Cyril Klimeš, CSc.

Ostravská univerzita v Ostravě, Přírodovědecká fakulta
Katedra informatiky a počítačů

OBSAH

1 BOOLEOVA ALGEBRA 4

1.1 LOGICKÉ PROMĚNNÉ A LOGICKÉ FUNKCE 5
1.1.1 Základní logické operátory 5
1.1.2 Zákony a pravidla Booleovy algebry 7
1.2 DEFINICE LOGICKÉ FUNKCE 8
1.2.1 Pravdivostní tabulka 8
1.2.2 Zápis logické funkce 9
1.3 LOGICKÁ FUNKCE N-PROMĚNNÝCH 11
1.3.1 Funkce jedné proměnné 11
1.3.2 Funkce dvou proměnných 11
1.3.3 Funkce více proměnných 13
1.4 ZJEDNODUŠOVÁNÍ ZÁPISU LOGICKÉ FUNKCE 14
1.4.1 Algebraická minimalizace 14
1.4.2 Grafická – Karnaughova metoda 14
1.4.3 Zjednodušení úplně zadané funkce 15
1.4.4 Zjednodušení neúplně zadané funkce 17
1.5 OBVODOVÉ ZNÁZORNĚNÍ BOOLEOVY ALGEBRY 18
1.5.1 Shefferova algebra 19
1.5.2 Pierceova algebra 19
1.5.3 Převod Booleovy algebry na Shefferovu algebru 20

2 REPREZENTACE ZÁKLADNÍCH LOGICKÝCH FUNKCÍ ELEKTRONICKÝMI OBVODY 21

2.1 FYZIKÁLNÍ PODSTATA SIGNÁLŮ 21
2.2 TECHNOLOGIE TTL (TRANZISTOR-TRANZISTOR LOGIC) 23
2.3 SYSTÉMY MOS/CMOS 27
2.4 LOGICKÁ HRADLA S TŘEMI STAVY 31

3 ZÁKLADNÍ LOGICKÉ ČLENY 33

3.1 INVERTOR 33
3.2 AND 34
3.3 OR 34
3.4 NAND 35
3.5 NOR 35
3.6 OSTATNÍ LOGICKÉ ČLENY 36
3.6.1 Nonekvivalence – XOR 36
3.6.2 Ekvivalence - NOXOR 36

4 KOMBINAČNÍ LOGICKÉ OBVODY 38

4.1 KÓDOVÁNÍ, DEKÓDOVÁNÍ A PŘEVODY KÓDŮ 38
4.1.1 Převod desítkových číslic do kódu 8421 38
4.1.2 Převod z kódu 8421 na desítkové číslice 40
4.1.3 Převod do Grayova kódu 42
4.1.4 Multiplexor 44
4.1.5 4-vstupý multiplexor 44
4.1.6 Dekodér 45
4.2 POROVNÁNÍ DVOJKOVÝCH INFORMACÍ 46
4.2.1 Rovnost dvou proměnných 46

4.2.2 Jednobitový porovnávací obvod 47
4.2.3 Dvoubitový porovnávací obvod 47
4.3 PARITA 48
4.4 SEČÍTAČKY A ODEČÍTAČKY 50
4.4.1 Sčítačka MODULO 2 50
4.4.2 Jednobitová neúplná sečítačka 51
4.4.3 Jednobitová úplná sečítačka 51
4.4.4 Jednobitová úplná odečítačka 52
4.4.5 Polosčítačka 53
4.4.6 Úplná sčítačka 54
5 SEKVENČNÍ LOGICKÉ OBVODY 55
5.1 KLOPNÝ OBVOD RS 56
5.2 KLOPNÝ OBVOD J-K 58
5.3 KLOPNÝ OBVOD TYPU T(RS-T) 59
5.4 KLOPNÝ OBVOD TYPU D 59
5.5 DVOJČINNÝ KLOPNÝ OBVOD J-K TYPU MASTER-SLAVE 60
5.6 FUNKCE POSOUVÁNÍ A ČÍTÁNÍ 61
5.6.1 Čítače kmitočtu 61
5.6.2 Posuvné registry 65
5.6.3 Kruhový registr 67
6 PRVKY MIKROPOČÍTAČE 68
6.1 ZÁKLADNÍ ARCHITEKTURA POČÍTAČE 68
6.2 OPERAČNÍ PAMĚŤ 69
6.2.1 Paměti typu ROM 69
6.2.2 Paměti typu RAM 71
6.3 JEDNOČIPOVÉ MIKROPOČÍTAČE 76

1 Booleova algebra

V této kapitole se dozvíte:

- Proč je Booleova algebra základem popisu logických funkcí počítačů?
- Jaké jsou základní logické proměnné a funkce Booleovy algebry?
- Jak jsou definovány logické funkce?
- Jaké jsou metody zjednodušování zápisů logických funkcí?
- Z jakých značek se vytváří obvodové značení logických funkcí?

Po jejím prostudování byste měli být schopni:

- Charakterizovat Booleovu algebru.
- Definovat zákony Booleovy algebry.
- Porozumět základním logickým proměnným a funkcím.
- Znát způsoby zápisu logických funkcí pomocí pravdivostních tabulek a Karnaughových map.
- Popsat obvodové znázorňování logických funkcí.
- Znát způsoby zjednodušování (minimalizaci) zápisu logických funkcí.

Klíčová slova této kapitoly:

Booleova algebra, logická negace, logický součn, logický součet, zákony Booleovy algebry, Karnaughova mapa, minimalizace.

Doba potřebná ke studiu: 8 hodiny

Průvodce studiem

Studium této kapitoly je poměrně náročné zejména pro ty z Vás, kteří dosud nemají žádné znalosti z logiky a Booleovy algebry. V takovém případě Vám zřejmě některé příklady budou připadat obtížně pochopitelné, ovšem nenechte se tím odradit, neboť pochopením této části se Vám usnadní studium následujících kapitol.
Na studium této části si vyhraďte alespoň 8 hodin. Doporučujeme studovat s přestávkami vždy po pochopení jednotlivých podkapitol. Po celkovém prostudování a vyřešení všech příkladů doporučujeme dát si pauzu, třeba 1 den, a pak se pusťte do vypracování korespondenčních úkolů.

Matematický prostředek vytvořil George S. Boole jako pomůcku pro znázornění filozofických problémů pomocí matematického aparátu, založeného na dvou pravdivostních hodnotách.

V technice bylo první využití Booleho algebry při popisu a návrhu reléových obvodů. Značného uplatnění však dosáhla Booleova algebra při návrhu logických obvodů sestavených z hradel „logický součin “, „logický součet “ a „ negace “ jimiž lze základní operace Booleovy algebry přímo realizovat. Boolova algebra nám slouží k matematickému popisu zákonů a pravidel výrokové logiky, která řeší vztahy mezi pravdivými (I) a nepravdivými (0) výroky. Jiná tvrzení nejsou povolena. Pravdivý výrok označujeme logickou hodnotou I a nepravdivý výrok logickou hodnotou 0. Nositelem elementární informace o pravdivosti a nepravdivosti výroků je logická proměnná, která

může nabývat hodnoty I nebo 0. Boolova algebra studuje dvě proměnné a funkce těchto proměnných. Je to algebra vztahů, nikoliv čísel.

1.1 Logické proměnné a logické funkce

Logická proměnná je veličina, která může nabývat pouze dvou hodnot, označených 0 a I (tedy dvojková proměnná), a nemůže se spojitě měnit.Tuto proměnnou označujeme x. Platí :

Logická funkce *n* proměnných $x_1, x_2, x_3, \ldots x_n$ je funkce, která může nabývat, stejně jako všechny logické proměnné, pouze dvou hodnot.

$y_1 = x_1, x_2, x_3, \ldots x_n$ $\qquad$ $y_2 = a_1, a_2, a_3, \ldots a_n$
$V = y_1, y_2$

Funkce rovnosti platí, když dvě logické proměnné A, B se sobě rovnají, tzn., jestliže A = I, B = I nebo A = 0, B = 0, což zapisujeme A = B. Dvě veličiny A = $a_1, a_2, a_3, \ldots a_n$; B = $b_1, b_2, b_3, \ldots b_n$ se sobě rovnají, když platí $a_i = b_i$ pro všechna *i.*

1.1.1 Základní logické operátory

V Booleově algebře jsou definovány tři základní operace, jimiž můžeme vyjádřit libovolnou logickou operaci. Jsou to :

- logická negace,
- logický součin,
- logický součet.

Hodnoty proměnné, kterou značíme Y, závislé na jednotlivých kombinací nezávisle proměnných A, B budeme znázorňovat pro základní logické operace (operátory) pravdivostní tabulkou. Počet řádků v pravdivostní tabulce je dán všemi možnými kombinacemi nezávisle proměnných hodnot. Ve sloupci, který je napravo, jsou zapsány stavy hodnot výstupně proměnných Y.

1.1.1.1 Logická negace

Tato operace, která se také velmi často nazývá inverze, dává výsledek zvaný negace. Je aplikována jen na jednu proměnnou. Mění hodnotu nezávisle proměnné na opačnou. Označuje se přidáním „ pruhu “ nad proměnnou *x.* Jejím výsledkem je hodnota Y = 1, jestliže A = 0, a naopak Y = 0, jestliže A = 1. Negace je vyjádřena zápisem „ NE “ (NOT).

Tabulka funkce negace :

A	$\overline{A}$
0	1
1	0

Tato definice vede ke vztahům : $\overline{0} = I$

$\overline{I} = 0$

1.1.1.2 Logický součin

Tato operace, nazývána také průnik nebo konjunkce, aplikovaná na dvě proměnné, vytváří součin neboli funkci AND těchto dvou proměnných. Logický součin nabývá hodnotu I jen tehdy, když všechny nezávislé proměnné mají hodnotu I. Jestliže je alespoň jedna hodnota rovna O, potom výsledná hodnota se rovná 0. Značí se symbolem $\wedge$ mezi dvěma proměnnými ($A \wedge B$). V praxi se však používá zápis :

$Y = A \,.\, B$

Logický součin je vyjádřen spojkou „ a “ (v angličtině „AND“ nebo „ & “).
Definice logického součinu může být vyjádřena vztahy :

Tabulka funkce logického součinu :

A	B	A. B
0	0	0
0	I	0
I	0	0
I	I	I

$$0 \,.\, 0 = 0$$
$$0 \,.\, I = I \,.\, 0 = 0$$
$$I \,.\, I = I$$

1.1.1.3 Logický součet

Tato operace, nazývána také sjednocení nebo disjunkce, aplikovaná na dvě proměnné, vytváří součet neboli funkci OR těchto dvou proměnných. Logický součet má hodnotu I, jestliže jedna nezávisle proměnná nebo druhá nezávisle proměnná nebo obě mají hodnotu I. Označuje se symbolem $\vee$ mezi dvěma proměnnými ($A \vee B$). V praxi se používá ovšem zápis : $Y = A + B$

Logický součet je vyjádřen „ nebo “ (v angličtině „OR“).
Definice logického součtu může být vyjádřena vztahy :

A	B	A + B
0	0	0
0	I	I
I	0	I
I	I	I

$$0 + 0 = 0$$
$$0 + I = I + 0 = I$$
$$I + I = I$$

1.1.2 Zákony a pravidla Booleovy algebry

Pro vyhodnocení logických výrazů, různé úpravy a zjednodušení logických výrazů je nutné znát zákony Booleovy algebry :

1. Komutativní zákon :

$$A + B = B + A$$
$$A \,.\, B = B \,.\, A$$

2. Asociativní zákon :

$$A + (B + C) = (A + B) + C$$
$$A \,.\, (B \,.\, C) = (A \,.\, B) \,.\, C$$

3. Distributivní zákon:

$$A + B \,.\, C = (A + B) \,.\, (A + C)$$
$$A \,.\, (B + C) = A \,.\, B + A \,.\, C$$

4. Zákon o agresívnosti prvku I a O :

$$A + I = I$$
$$A \,.\, 0 = 0$$

5. Zákon o neutrálnosti prvku I a O:

$$A + 0 = A$$
$$A \,.\, I = A$$

6. Zákon o vyloučení třetího :

$$A + \overline{A} = I$$
$$A \,.\, \overline{A} = 0$$

7. Zákon dvojité negace :

$$\overline{\overline{A}} = A$$

8.Zákon absorpce:

$$A + A = A$$
$$A \,.\, A = A$$
$$A + A \,.\, B = A$$
$$A \,.\, (A + B) = A$$

9.Zákon absorpce negace:

$$A + \overline{A} = I$$
$$A \,.\, \overline{A} = 0$$
$$A + \overline{A} \,.\, B = A + B$$
$$A \,.\, (\overline{A} + B) = A \,.\, B$$
$$\overline{A} + A.B = \overline{A} + B$$
$$\overline{A}(A + B) = \overline{A}.B$$

Některé zákony vyplívají přímo z definic základních logických operací – jsou to operace součtu a součinu s hodnotami 0 a I. Ze všeobecných zákonů je nejvýznamnější de Morganův zákon, který se vyjadřuje dvěma rovnostmi.

10. De Morganův zákon :

$$\overline{A + B + C + ...} = \overline{A} \,.\, \overline{B} \,.\, \overline{C} \,.\, ...$$
$$\overline{A \,.\, B \,.\, C \,.\, ...} = \overline{A} + \overline{B} + \overline{C} + ...$$

Tento zákon lze tedy vyjádřit tak, že negaci funkce získáme nahrazením každé proměnné její negací a záměnou značek součtu a součinu navzájem. Při použití tohoto zákona je třeba věnovat velkou pozornost implicitním závorkám :

$$A + B \,.\, C = A + (B \,.\, C)$$

Tedy :

$$\overline{A + B \,.\, C} = \overline{A} \,.\, (\overline{B} + \overline{C})$$

a neplatí :

$$\overline{A \,.\, B} + \overline{C}$$

1.2 Definice logické funkce

1.2.1 Pravdivostní tabulka

Pravdivostní tabulka je tabulka, do které se zapisuje logická (Booleovská funkce). Pravdivostní tabulka má r + n sloupců a 2^n řádků. Číslo r je počet sloupců výsledných funkcí (obyčejně bývá jedna výsledná funkce – tedy jeden sloupec). Číslo n udává počet proměnných. Číslo 2^n udává počet všech možných kombinací proměnných, kde číslo n je počet proměnných. Tyto kombinace reprezentuje počet řádků.

Příklad : Pokud je dána logická funkce, která má tři proměnné a jednu výslednou funkci, tak pravdivostní tabulka bude mít čtyři sloupce
(r + n = 1 + 3 = 4) a osm řádků ($2^n = 2^3 = 8$).

1.2.1.1 Úplně zadaná funkce

Logická funkce je úplně zadaná, jestliže je známa její hodnota I nebo 0 pro všechny možné kombinace hodnot proměnných. Těchto kombinací je pro *n* proměnných 2^n. Lze tak sestavit pravdivostní tabulku.

Příklad : Je dána funkce tří proměnných f (A, B, C), pravdivostní tabulka bude tedy vypadat následovně :

A	B	C	f
0	0	0	0
0	0	I	I
0	I	0	I
0	I	I	0
I	0	0	0
I	0	I	I
I	I	0	0
I	I	I	I

1.2.1.2 Neúplně zadaná funkce

Logická funkce je neúplně zadaná, když její hodnota pro některé kombinace hodnot proměnných je libovolná nebo není určena. S tímto případem se setkáváme, když některé kombinace hodnot jsou fyzikálně nemožné. Hodnotu funkce poté značíme x nebo ∅. Pravdivostní tabulka může vypadat následovně:

A	B	C	f
0	0	0	0
0	0	I	I
0	I	0	x
0	I	I	x
I	0	0	0
I	0	I	I
I	I	0	0
I	I	I	x

1.2.2 Zápis logické funkce

1.2.2.1 Základní tvary funkce

Logickou funkci můžeme zapsat ve dvou tvarech, nazývaných základní součtový a základní součinový tvar. V praxi se ale často používají názvy úplná disjunktivní normální forma – ÚDNF a úplná konjunktivní normální forma – ÚKNF.

ÚDNF – je to součet základních součinů přímých nebo negovaných proměnných. Každý základní součin (minterm – z ang. minimal polynomial term) nabývá hodnoty I pro určitou kombinaci, kdy funkce má hodnotu I, a hodnoty 0 pro všechny ostatní kombinace. ÚDNF vyjadřuje funkci jako součet případů, kdy má hodnotu I.

ÚKNF – je to součin základních součtů přímých nebo negovaných proměnných. Každý základní součet nabývá hodnoty 0 pro určitou kombinaci, kdy funkce má hodnotu 0, a hodnoty I pro všechny ostatní kombinace. ÚKNF vyjadřuje funkci jako součin případů, kdy má hodnotu 0.

1.2.2.2 Výpis logických funkcí z pravdivostní tabulky

Tabulka funkce tří proměnných (příklad – libovolná funkce)

A	B	C	f
0	0	0	I
0	0	I	0
0	I	0	I
0	I	I	0
I	0	0	I
I	0	I	0
I	I	0	I
I	I	I	0

a) ÚDNF – základní součtový tvar

Případy, kdy logická funkce je rovna I: Odpovídající součiny:

$$\text{kombinace } A,B,C \begin{cases} 0\,0\,0 \\ 0\,I\,0 \\ I\,0\,0 \\ I\,I\,0 \end{cases} \qquad \begin{matrix} \overline{A}.\overline{B}.\overline{C} \\ \overline{A}.B.\overline{C} \\ A.\overline{B}.\overline{C} \\ A.B.\overline{C} \end{matrix}$$

Výslednou funkci dostáváme jako součet základních součinů (algebraicky) :

$$f = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$$

b) ÚKNF – základní součinový tvar
Případy, kdy logická funkce je rovna 0 : Odpovídající součty :

$$\text{kombinace } A,B,C \begin{cases} 0\,0\,I \\ 0\,I\,I \\ I\,0\,I \\ I\,I\,I \end{cases} \qquad \begin{array}{l} A+B+\overline{C} \\ A+\overline{B}+\overline{C} \\ \overline{A}+B+\overline{C} \\ \overline{A}+\overline{B}+\overline{C} \end{array}$$

Výslednou funkci dostáváme jako součin základních součtů (algebraicky) :

$$f=(A+B+\overline{C}).(A+\overline{B}+\overline{C}).(\overline{A}+B+\overline{C}).(\overline{A}+\overline{B}+\overline{C})$$

V praxi se častěji používá součtový tvar, tedy ÚDNF.

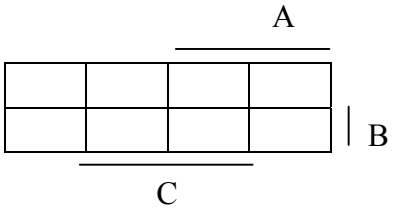
1.2.2.3 Zápis logické funkce do Karnaughovy mapy

Karnaughova mapa umožňuje přehledný zápis všech hodnot logické funkce. Porovnáme-li pravdivostní tabulky (např. tří proměnných), zjistíme, že levá strana tabulky je vždy stejná a nepřináší nové informace. Každé kombinaci nezávisle proměnných je v Karnaughově mapě přidělen jeden čtverec, do kterého zapisujeme výstupní hodnotu funkce. Počet čtverců je tedy roven počtu řádků v pravdivostní tabulce. Přiřazení vstupních proměnných jednotlivým řádkům a sloupcům se provádí úsečkami, někdy algebraickým označením. Zápisu funkce do Karnaughovy mapy se také někdy říká grafická metoda.

Příklad : Pro příklad použijeme dříve uvedenou pravdivostní tabulku tří proměnných.

A	B	C	f
0	0	0	I
0	0	I	0
0	I	0	I
0	I	I	0
I	0	0	I
I	0	I	0
I	I	0	I
I	I	I	0

Postup :
Nejprve si nakreslíme tabulku (Karnaughovu mapu), která bude mít počet políček (čtverců) stejný, jako počet řádků v pravdivostní tabulce (v našem případě 8). Potom přiřadíme vstupní proměnné jednotlivým řádkům a sloupcům např. pomocí úseček (tento způsob je více přehledný). Mapa by měla vypadat následovně :



Poté na základě pravdivostní tabulky, respektive na základě vstupních proměnných, zapíšeme do tabulky hodnotu výstupní funkce. Konečná Karnaughova mapa by měla vypadat následovně (je použita forma ÚDNF):

	A		
I	0	0	I
I	0	0	I

B

C

1.3 Logická funkce n-proměnných

Logická funkce f_n n-proměnných nabývá všech možných hodnot, pro všechny možné kombinace n-proměnných. Počet funkcí je $(2^2)^n$. Toto číslo roste velmi rychle např. pro n = 3 je počet 256 možných kombinací.
Funkce rozdělujeme takto :

- Funkce jedné proměnné, kde je počet funkcí $(2^2)^1 = 4$.
- Funkce dvou proměnných, kde je počet funkcí $(2^2)^2 = 16$.
- Funkce více než dvou proměnných.

1.3.1 Funkce jedné proměnné

Funkce jedné proměnné je kombinace hodnot A. Pro jednu proměnnou jsou funkce uvedeny v následující tabulce :

A	f_0	f_1	f_2	f_3
0	0	0	I	I
I	0	I	0	I

Funkce jsou označeny f_0 až $f_{1,}$ kde index představuje hodnotu dvojkového čísla umístněného v odpovídajícím sloupci, přičemž váhy rostou po řádcích směrem nahoru. Platí:

Konstanty : $f_0 = 0$ a $f_3 = I$

Proměnná sama : $f_1 = A$

Negace proměnné : $f_2 = \overline{A}$

1.3.2 Funkce dvou proměnných

Pro dvě proměnné je počet funkcí 16. Jsou dány pravdivostní tabulkou, kde 16 funkcí f_n nabývá všech možných hodnot pro všechny možné kombinace dvou proměnných. Funkce jsou označeny f_0 až f_{15} , kde index představuje hodnotu dvojkového čísla umístněného v odpovídajícím sloupci, přičemž váhy rostou po řádcích směrem nahoru.

A	B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	I	I	I	I	I	I	I	I
0	I	0	0	0	0	I	I	I	I	0	0	0	0	I	I	I	I
I	0	0	0	I	I	0	0	I	I	0	0	I	I	0	0	I	I
I	I	0	I	0	I	0	I	0	I	0	I	0	I	0	I	0	I

Funkce jedné proměnné :

Konstanty : $f_0 = 0$ a $f_{15} = I$

Proměnná sama : $f_3 = A$ a $f_5 = B$

Negace proměnné : $f_{12} = \overline{A}$ a $f_{10} = \overline{B}$

Funkce odpovídající základním operátorům :

Logický součet ve významu slučovacím : $f_7 = A + B$

L Lgický součin : $f_1 = A \,.\, B$

Osm nových funkcí :

Součet ve významu vylučovacím neboli součet modulo 2 neboli nonekvivalence, též XOR, běžně zapisována

$A \oplus B$ $f_6 = A\overline{B} + \overline{A}B$

Funkce ekvivalence, zapisovaná A ≡ B, nebo též

$A \otimes B$ $f_9 = AB + \overline{A}\,\overline{B}$

Funkce ani jeden není nebo také NOR (negace logického součtu) – Piercova funkce

$$f_8 = \overline{A}\,\overline{B} = \overline{A + B}$$

Funkce alespoň jeden není neboli Schefferova funkce, nazývaná také NAND (negace logického součinu)

$$f_{14} = \overline{A} + \overline{B} = \overline{AB}$$

Negace implikace – inhibice, zábrana $f_2 = A\overline{B}$

Negace obrácené implikace $f_4 = \overline{A}B$

Obrácená implikace $f_{11} = A + \overline{B}$

Implikace $f_{13} = \overline{A} + B$

Znalost logických funkcí je nutná zejména při návrhu, minimalizaci a konstrukci logických obvodů. Vynecháním některé funkce může dojít ke konstrukční závadě. Pomocí funkcí NAND a NOR lze vyjádřit jakoukoliv funkci.

1.3.3 Funkce více proměnných

Pro n proměnných je možné napsat $(2^2)^n$ určitých logických funkcí Například když si vezmeme funkci tří proměnných. Pro tři proměnné existuje 256 funkcí. Vyjádření takové funkce by bylo již nepřehledné. Pro vyjádření můžeme však využít již známých funkcí dvou proměnných, protože :

$$f(A, B, C) = A \,.\, f(I, B, C) + \overline{A} \,.\, f(0, B, C)$$

Kontrola : Pro ověření tohoto vztahu stačí položit A = I nebo A = 0. Z toho plyne, že

$$f(I, B, C) \text{ a } f(0, B, C)$$

jsou funkce dvou proměnných. Stejná úvaha se může použít pro případ více než tří proměnných.

Příklad:
Chceme určit logickou funkci zařízení které:

- rozsvítí zelenou žárovku – F1, když v nějakém výrobním procesu překročí kritickou hodnotu pouze jedna ze sledovaných veličin, např. tlak (x), nebo teplota (y), nebo vlhkost (z), nebo žádná,
- rozsvítí červenou žárovku - F2, když je překročena kritická hodnota kterýchkoli dvou veličin současně,
- zapne sirénu - F3, když jsou překročeny kritické hodnoty všech tří veličin současně.

Přiřazení hodnot 0 a 1 veličinám x,y,z (které budeme považovat logické proměnné) provedeme následovně. Když se překročí kritická hodnota veličiny je proměnná rovna 1, když se nepřekročí je rovna 0. Jde tedy o tři samostatné funkce, které podle slovního zadání lze přepsat do tabulky pro všechny kombinace hodnot x,y,z tak, že do příslušného řádku dosadíme za F=1 jestliže má žárovka svítit a 0 jestliže ne.

x	y	z	F1	F2	F3
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	0	1

Každou funkci vyjádříme samostatně známým zápisem (ÚDNF):

$$F1 = \overline{x}\overline{y}\overline{z} + \overline{x}\overline{y}z + \overline{x}y\overline{z} + x\overline{y}\overline{z} \qquad F2 = \overline{x}yz + x\overline{y}z + xy\overline{z}$$

$$F3 = xyz$$

1.4 Zjednodušování zápisu logické funkce

Zjednodušování, často také minimalizace, logické funkce je určitý postup, kterým lze získat jednodušší vyjádření logické funkce. Minimalizací získáme jednodušší strukturu logické funkce při zachování stejného výsledku. Minimalizace se proto používá k nejjednodušší technické realizaci, tzn., že minimalizace je důležitá při konstrukci logických obvodů – stejná funkce může být zhotovena z menšího počtu logických členů = ekonomičnost. Nejpoužívanější jsou dva způsoby, a to algebraická minimalizace – pomocí Booleovy algebry, nebo grafická minimalizace. Grafických metod je více, například Quinova-McCluskeyho – metoda, ale nejpoužívanější je Karnaughova metoda.

1.4.1 Algebraická minimalizace

Algebraická minimalizace logických funkcí je upravování logického výrazu podle zákonů a pravidel Booleovy algebry. Výsledná funkce je vyjádřená co nejjednodušeji se stejným chováním, což při realizaci znamená jednodušší konstrukci logického obvodu.

Příklad algebraické minimalizace demonstrujeme na jednoduchém příkladě :

A	B	f
0	0	I
0	I	0
I	0	I
I	I	I

Podle již uvedených a tedy i známých pravidel určíme výslednou funkci (ÚDNF):

$$f = \overline{A}\,\overline{B} + A\overline{B} + AB$$

Tuto funkci dle Booleových zákonů a pravidel můžeme upravit na tvar:

$$f = \overline{A}B + A\overline{B} + AB = A(B + \overline{B}) + \overline{A}\,\overline{B} = A + \overline{A}\,\overline{B} = A + \overline{B}$$

Jak vidíme, výsledná zjednodušená logická funkce je omnoho jednodušší než předešlá funkce. Ovšem tato metoda je velice riskantní a úpravy pomocí této metody jsou obtížné. Pro více než tři proměnné je téměř nepoužitelná.

1.4.2 Grafická – Karnaughova metoda

Pro zjednodušení funkce pomocí algebraické minimalizace spojujeme součiny (mintermy), které se liší v jediné proměnné. Tyto součiny se nazývají sousední.

Příklad: $A\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D = A\,B\,\overline{C}\,(\overline{D} + D) = A\,B\,\overline{C}$

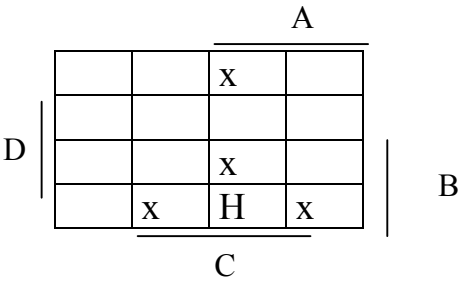
Karnaughova metoda se díky jasnému geometrickému postupu vyhýbá hledání sousedním součinů složitým algebraickým způsobem, který je navíc velmi nespolehlivý. Tato metoda se používá velmi dobře pro 3, 4 a 5 proměnných.

Minimální logickou funkci stanovíme tak, že v Karnaughově mapě vytváříme tzv. **podmapy**. Podmapou rozumíme sjednocení 2^k sousedních stavů, ve kterých nabývá logická funkce hodnoty 1 pro k = 0, 1, 2, ... , n-1. Každou podmapou vyloučíme k proměnných z dvou, čtyř až 2^{n-1} základních součinů. Snažíme se vytvářet co největší podmapy, abychom vyloučili co největší počet proměnných. Využíváme k tomu také **neurčité stavy**.

Výběr podmap provádíme podle následujících **pravidel**:

- vybranými podmapami musí být pokryty všechny jednotkové stavy logické funkce,
- do podmapy spojujeme stejné stavy, které spolu sousedí hranou, a to i přes okraje mapy. Rohy mapy jsou též sousedními stavy. Členy dvou sousedních polí se od sebe liší jednou proměnnou a tuto proměnnou můžeme vyloučit,
- podmapu pravidelného tvaru (čtverec, obdélník) vytváříme co největší, aby se ze skupiny stavů vyloučilo co nejvíce proměnných,
- podmapy se mohou prolínat,
- nevytváříme zbytečné podmapy, tzn. Že nespojujeme ty stavy, které už byly předtím pokryty jinou podmapou,
- čím větší bude podmapa, tím jednodušší bude výsledný výraz.

Příklad : Hledání sousedních součinů pomocí Karnaughovy metody, příklad je pro čtyři proměnné.



Jak vidíme, najít sousední součiny x zadaného políčka H je pomocí Karnaughovy metody jednoduché.

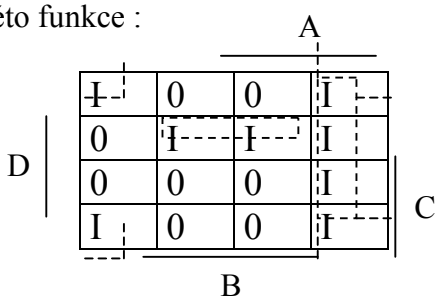
1.4.3 Zjednodušení úplně zadané funkce

Do políčka zapíšeme pomocí logických hodnot I a 0 pro všechny kombinace proměnných odpovídající hodnoty funkce. Zjednodušování spočívá v hledání pro funkci v disjunktivním tvaru smyčky dvou, čtyř a osmi sousedních políček tak, aby se ze skupiny součinů (mintermů) vyloučila jedna, dvě nebo tři proměnné. Pro vyjádření smyček se používají pouze jen políčka obsahující I.

Příklad : Libovolná pravdivostní tabulka úplně zadané funkce čtyř proměnných

A	B	C	D	f
0	0	0	0	I
0	0	0	I	0
0	0	I	0	I
0	0	I	I	0
0	I	0	0	0
0	I	0	I	I
0	I	I	0	0
0	I	I	I	0
I	0	0	0	I
I	0	0	I	I
I	0	I	0	I
I	0	I	I	I
I	I	0	0	0
I	I	0	I	I
I	I	I	0	0
I	I	I	I	0

Karnaughova mapa této funkce :



Úplná disjunktní normální forma této funkce je :

$$f = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + A\,\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D + A\,\overline{B}\,C\,\overline{D} + A\,\overline{B}\,C\,D + A\,B\,\overline{C}\,D$$

Z této mapy můžeme udělat tři smyčky :

Smyčka se dvěma políčky

$$\overline{A}\,B\,\overline{C}\,D + A\,B\,\overline{C}\,D = B\,\overline{C}\,D\,(\overline{A} + A) = B\,\overline{C}\,D$$

Smyčka se čtyřmi políčky na pravé straně

$$A\,\overline{B}\,\overline{C}\,(D + \overline{D}) + A\,\overline{B}\,C\,(D + \overline{D}) = A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C = A\,\overline{B}\,(\overline{C} + C) = A\,\overline{B}$$

Smyčka s políčky v rozích mapy

$$\overline{A}\,\overline{B}\,\overline{D}\,(\overline{C} + C) + A\,\overline{B}\,\overline{D}\,(\overline{C} + C) = \overline{A}\,\overline{B}\,\overline{D} + A\,\overline{B}\,\overline{D} = \overline{B}\,\overline{D}\,(\overline{A} + A) = \overline{B}\,\overline{D}$$

Celkový výsledek logické funkce po minimalizaci – zjednodušení je tedy :

$$f = B\,\overline{C}\,D + A\,\overline{B} + \overline{B}\,\overline{D}$$

Tento způsob upravování je náročný a zbytečně zdlouhavý. Nejpoužívanější způsob je vyloučit proměnné, které mění svůj stav a ponechají se pouze proměnné, které se nemění.

Příklad:
Smyčka se čtyřmi políčky v rozích mapy
A – mění stav, takže se vyloučí
B – zůstává stejná s hodnotou 0, zapíšeme $\overline{B}$
C – mění svůj stav, takže se vyloučí
D – zůstává stejná s hodnotou 0, zapíšeme $\overline{D}$
Dostáváme tedy součin $\overline{B}\,\overline{D}$

Jak je vidět, tento součin je naprosto shodný se součinem s předešlé metody. Z toho plyne, že obě metody jsou správné, ale poslední metoda je jednodušší, protože se vyhneme upravování členů.

1.4.4 Zjednodušení neúplně zadané funkce

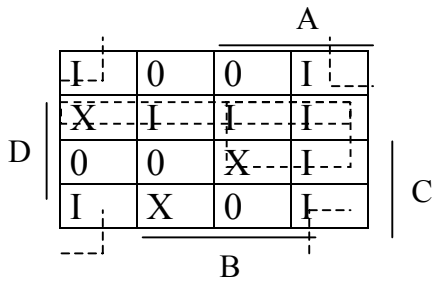
Pro zjednodušení neúplně zadané funkce použijeme funkci stejnou jako pro zjednodušování úplně zadané funkce. Neúplně zadaná funkce může v určitých políčcích mapy nabývat libovolné hodnoty 0 nebo 1 – zapisujeme ji symbolem X. Při minimalizaci funkce se této skutečnosti využívá tak, že vytváříme podmapy s využitím neurčitých stavů, které považujeme za jednotkové, nebo za nulové, jak je to z hlediska minimalizace nejvýhodnější. Pro usnadnění smyček zapíšeme X do políček místo hodnoty I, v ostatních případech 0. Zápis bude tedy vypadat následovně:

$$f = \overline{B}\,\overline{D} + \overline{C}\,D + A\,D$$

Pravdivostní tabulka neúplně zadané funkce:

A	B	C	D	f
0	0	0	0	I
0	0	0	I	X
0	0	I	0	I
0	0	I	I	0
0	I	0	0	0
0	I	0	I	I
0	I	I	0	X
0	I	I	I	0
I	0	0	0	I
I	0	0	I	I
I	0	I	0	I
I	0	I	I	I
I	I	0	0	0
I	I	0	I	I
I	I	I	0	0
I	I	I	I	X

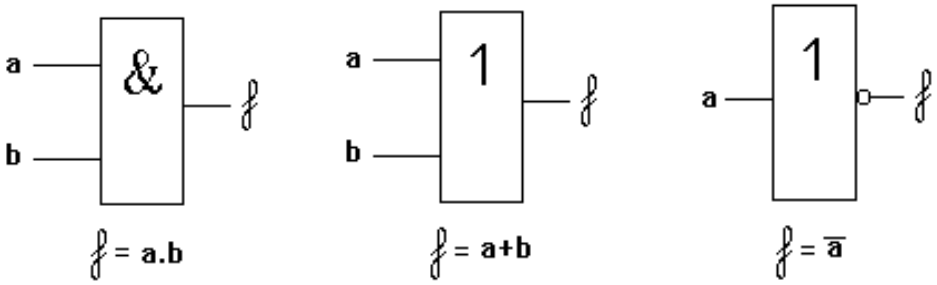
Karnaughova mapa neúplně zadané funkce:



$$f = \overline{B}\overline{D} + \overline{C}D + AD$$

Tyto metody jsou použitelné pro zjednodušování logických obvodů realizovaných pouze pomocí základních logických operátorů – logického součinu a součtu. Pro systémy realizované pomocí složitějších funkcí jsou nepoužitelné.

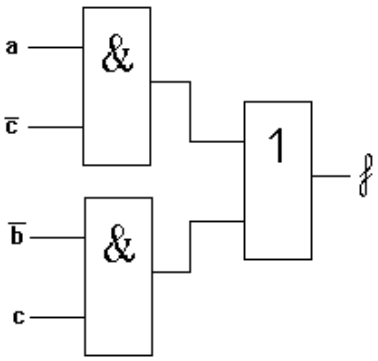
1.5 Obvodové znázornění Booleovy algebry



Pravidla pro kresbu značek:

- Vstup je vždy zleva, výstup zprava.
- Značky se nesmějí otáčet.
- Spoje mají být rovnoběžné s okraji listu.

Př: $f = a.\overline{c} + \overline{b}.c$



1.5.1 Shefferova algebra

Je vybudovaná na jedné logické funkci = negace logického součinu NAND.

- Pro libovolný počet proměnných

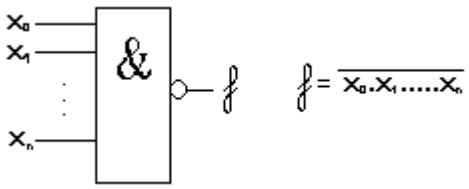
$$f = \overline{x.y}$$

Pravidla:

$$\overline{x.x} = \overline{x}$$
$$\overline{x.0} = 1$$
$$\overline{x.1} = \overline{x}$$
$$\overline{\overline{x.y}.1} = \overline{\overline{x.y}} = x.y$$
$$\overline{\overline{x.x}.\overline{y.y}} = \overline{\overline{x}.\overline{y}} = x + y$$

- Pomocí operace NAND lze realizovat všechny operace Booleovy algebry.
- Neplatí zákon asociativní: $\overline{\overline{x.y}.z} \neq \overline{x.\overline{y.z}} \neq \overline{x.y.z}$

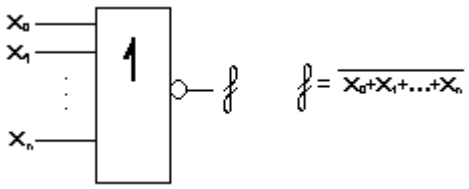
Obvodové znázornění Shafferovy algebry:



1.5.2 Pierceova algebra

Vystavěna na operaci NOR (negace logického součtu) - obdobné jako Shefferova algebra.

Obvodové znázornění Pierceovy algebry:



1.5.3 Převod Booleovy algebry na Shefferovu algebru

Opakovanou aplikací de Morganových pravidel:

$$\overline{a + b} = \overline{a} \,.\, \overline{b}$$

Př.:

$$\begin{aligned} f &= \overline{\overline{b} \,.\, c + \overline{a} \,.\, c + \overline{a} b d} \\ &= \overline{\overline{\overline{b} \,.\, c + \overline{a} \,.\, c + \overline{a} b d}} = \\ &= \overline{\overline{b} \,.\, c} \,.\, \overline{\overline{\overline{a} \,.\, c}} \,.\, \overline{\overline{\overline{a} b d}} \end{aligned}$$

Kontrolní otázky:

1. Čím se liší zápis logické funkce pomocí úplné disjunktivní normální formy od úplné konjunktivní normální formy.
2. Jaký bude počet logických funkcí pro 4 proměnné?
3. Proč je algebraická minimalizace neúplně zadaných funkcí velmi obtížná a téměř nepoužitelná?

Úkoly k zamyšlení:

1. Zamyslete se nad postupem návrhu logických funkcí reálných řídících systémů od popisu funkce slovním zadáním, přes pravdivostní tabulku, Karnaughovu mapu, minimalizaci funkce až po převod minimalizované funkce do Shefferovy algebry a výsledné zapojení logických obvodů realizovaných ze členů NAND.

Korespondenční úkol:

1. Navrhněte zapojení logických obvodů ze členů NAND realizující řízení výtahu. Motor výtahu je zapnut, když dveře v 1. poschodí a 2. poschodí jsou zavřeny, cestující jsou v kabině a je stlačeno tlačítko v kabině., nebo dveře v 1. poschodí a 2. poschodí jsou zavřeny, cestující nejsou v kabině a je stlačeno odesílací tlačítko.

Shrnutí obsahu kapitoly

V této kapitole jste se seznámili se základními principy návrhu logických funkcí pomocí Booleovy algebry. Důraz v této kapitole byl kladen na pochopení způsobu zápisu logických funkcí a jejich minimalizaci. Velká pozornost byla věnována realizaci těchto funkcí pomocí logických obvodů.

2 Reprezentace základních logických funkcí elektronickými obvody

V této kapitole se dozvíte:

- Jaké jsou základní interpretace logických hodnot pomocí elektrického napětí a proudu?
- Jaké konkrétní hodnoty napětí jsou přiřazeny logickým úrovním v logice TTL?

Po jejím prostudování byste měli být schopni:

- Znát způsob interpretace logických hodnot pomocí elektrických signálů.
- Porozumět základům TTL logiky.

Klíčová slova této kapitoly:

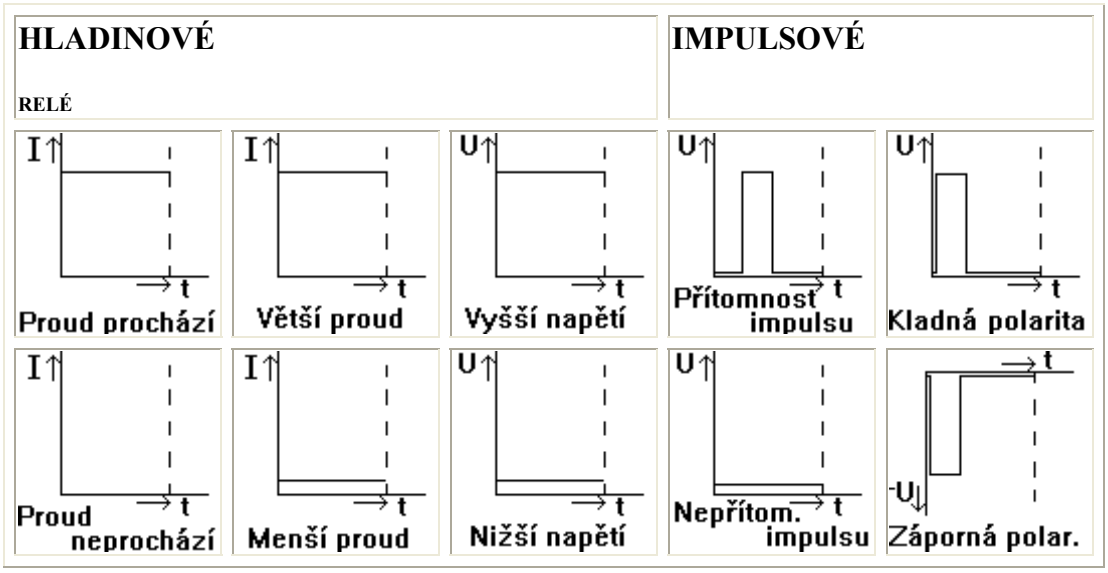
Hladinové signály, impulsové signály, TTL, zakázané pásmo.

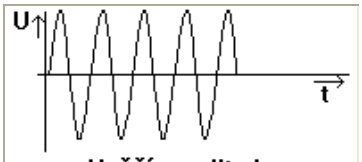
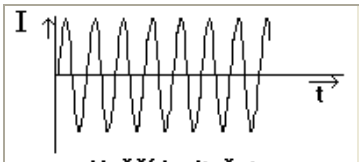
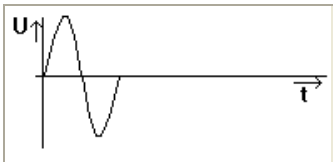
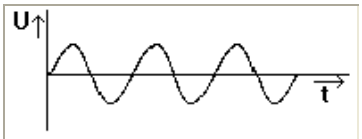
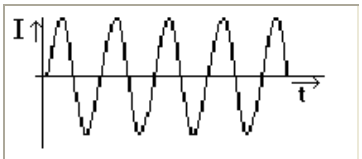
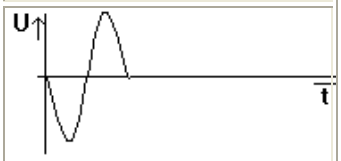
Doba potřebná ke studiu: 1 hodina

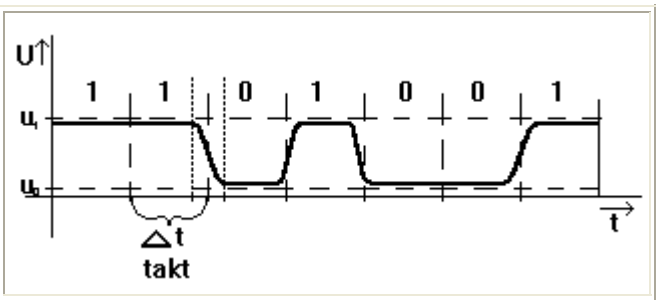
Průvodce studiem

Studium této kapitoly je jednoduché a popisným způsobem zde nastudujete principy zobrazení logických úrovní elektrickými signály.
Na studium této části si vyhraďte 1 hodinu.

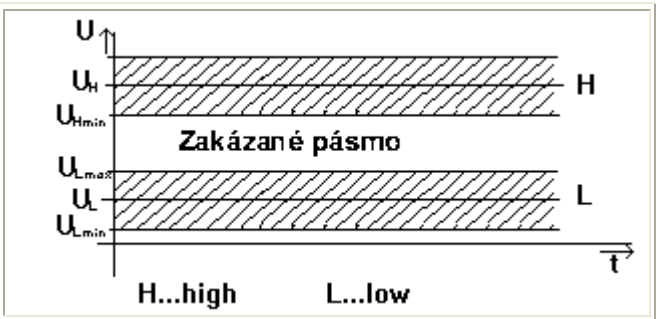
2.1 Fyzikální podstata signálů



AMPLITUDA	KMITOČET	FÁZE
 Vyšší amplituda	 Vyšší kmitočet	
		



Časová závislost logických úrovní



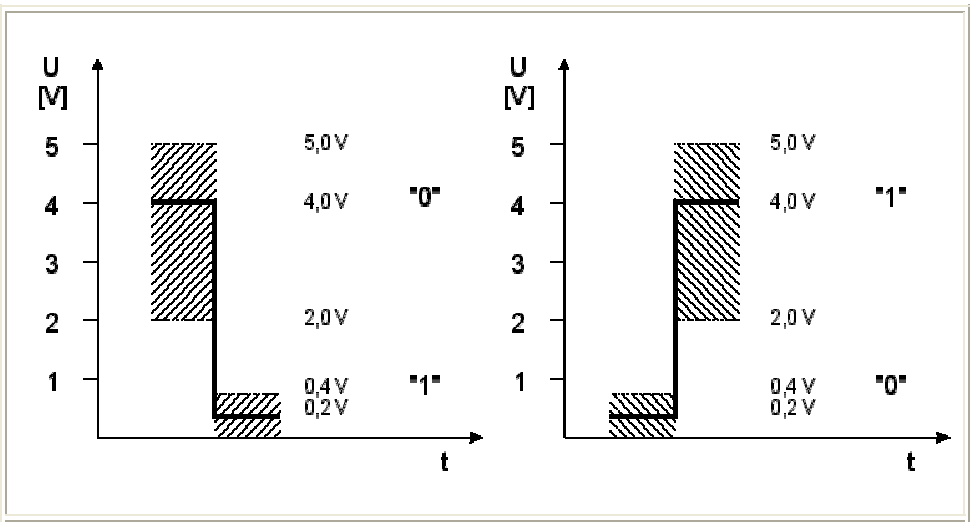
Zakázané pásmo v logických úrovních

Hodnoty jsou stanoveny pro každou výrobní technologii zvlášť.

L ~ 0 H ~ 1......pozitivní logika
L ~ 1 H ~ 0......negativní logika

Vzhledem k tomu, že parametry reálného logického členu se různí kus od kusu (užívají se odpory s určitou tolerancí, tranzistory a diody, které mohou mít různé parametry), není možné stanovit přesnou hodnotu napětí odpovídající logické “0” resp. “1” v té které logické síti. Místo toho se logické členy

konstruují tak, aby nebyly citlivé na změnu napětí vstupních parametrů pokud tyto leží v určitém intervalu napětí.



Tolerance logických hodnot pro negativní a pozitivní logiku

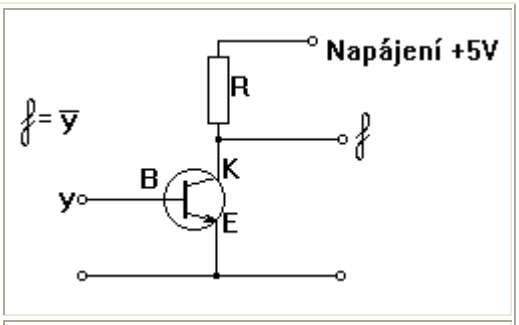
Například pro hradla TTL (transistor-transistor-logic) jsou příslušné intervaly následující:
$U_{vst}(0)$ = max. 0,8 V
$U_{vst}(1)$ = min. 2 V
neboli pro logickou "0" je povolený interval vstupních napětí 0 - 0.8V pro logickou "1" 2 - 5 V. Hradlo samo má zaručovaná výstupní napětí:
$U_{výst}(1)$ = min. 2,4 V
$U_{výst}(0)$ = max. 0,4 V
tj. hluboce v povolené toleranci napětí vstupních. Napájecí napětí je (5± 0,25) V. Uvedené hodnoty jsou typické pro tzv. tranzistorovou logiku a byly implementovány u celé řady výrobců logických obvodů.

2.2 Technologie TTL (tranzistor-tranzistor logic)

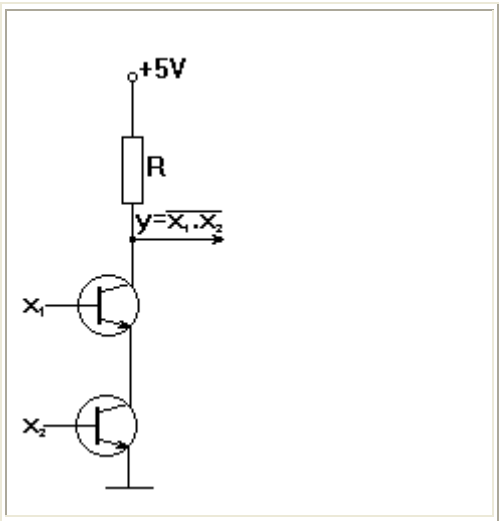
Největšího rozšíření dosáhly ve své době integrované logické systémy TTL (Transistor-Transistor-Logic). V dnešní době jsou však nahrazovány systémy STTL, MOS a CMOS, které mají nižší spotřebu a srovnatelnou rychlost. Zachovala se však definice logických úrovní; pokud má moderní logický systém stejně definované napěťové úrovně logické nuly a jedničky, nazývá se kompatibilní s TTL na logických úrovních (logic level TTL compatible).

Parametry TTL:
- napájecí napětí + 5V
- L < 0,8V L ~ 0,4V
- H > 2,0V H ~ 2,4V

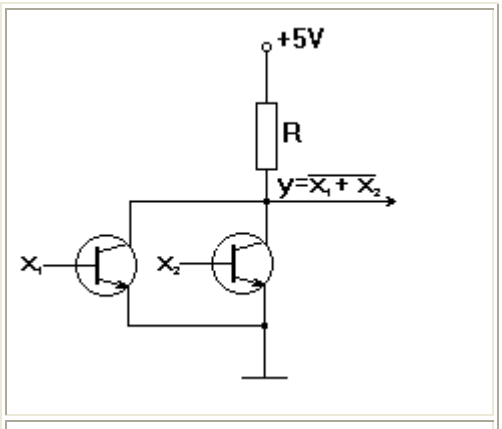
Základní funkce jsou zřejmé z následujících obrázků.



Invertor v TTL



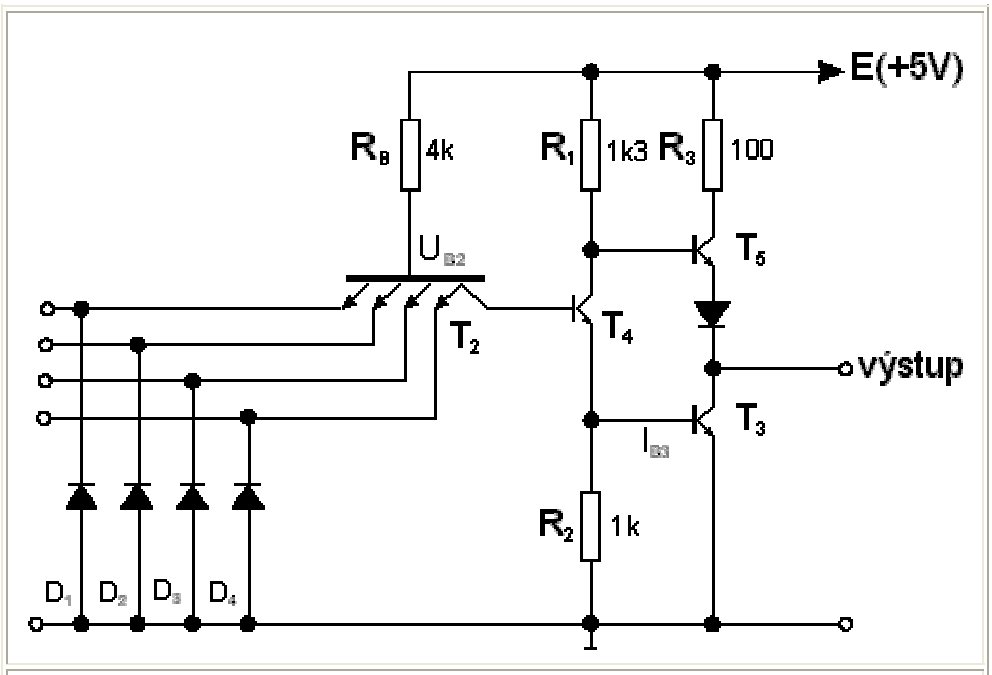
NAND pomocí dvou tranzistorů



NOR pomocí dvou tranzistorů

x_1	x_2	NAND	x_1	x_2	NOR
0	0	1	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0

Na následujícím obrázku je naznačena praktická realizace hradla NAND. Vstupem je víceemitorový tranzistor.



Základní hradlo NAND

Aby byla umožněna vyšší zatížitelnost hradla, je na konci hradla zabudován koncový stupeň. Z důvodů omezení záporných napěťových špiček na vstupech hradla byly vstupy opatřeny ochrannými diodami, které nedovolí vyšší záporné napětí na vstupech než cca -0,6V. Funkce obvodu je následující:

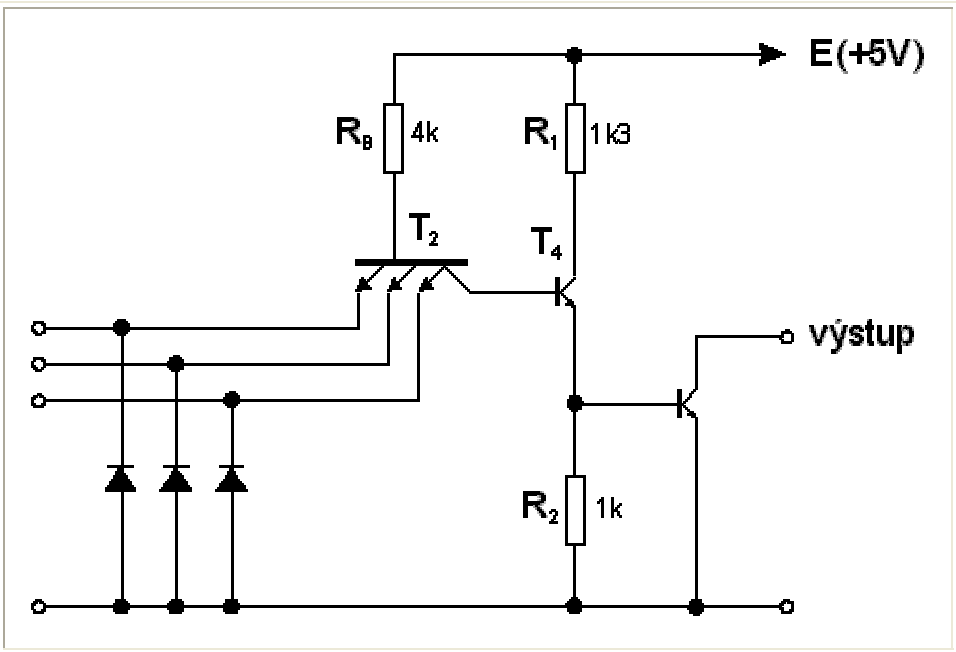
a. Všechny vstupy jsou ve stavu logické jedničky.
V tomto případě tranzistor T_2 pracuje v inverzním režimu. Tranzistor T_4 je saturován a tranzistor T_3 je vybuzen. Napětí na kolektoru tranzistoru T_4 je přibližně $U_D + U_{kesat}$. Aby byl tranzistor T_5 uzavřen (má na bázi napětí U_{k4}) je do obvodu vložena dioda D, která posunuje napětí emitoru T_5 na napětí $U_{kesat}+U_D$. Tranzistor T_3 je tedy otevřen a T_5 uzavřen. Na výstupu je napětí U_{kesat}, což je napětí logické 0.

b. Jeden nebo více vstupů jsou ve stavu logické nuly.
Pak je tranzistor T_4 a rovněž T_3 uzavřen a odpor R_1 zaručuje nasycení tranzistoru T_5. Ten slouží nyní jako emitorový sledovač pro výstupní napětí, takže může do zátěže dodat podstatně větší proud než tomu bylo u předchozího zapojení. Napětí na výstupu je menší než E o spád na diodě báze-emitor tranzistoru T_5 a na diodě D, je tedy $U_1 \approx E - 2\ U_D$. Je užitečné si všimnout, že

napájecí napětí E nemůžeme libovolně zmenšovat. Je-li totiž T_3 otevřen, je napětí na bázi T_2 napětí $U_{B2} \approx 3U_D$. Aby T_2 pracoval v inverzním režimu, je třeba, aby napětí na emitoru T_2 bylo větší než toto napětí. Vzhledem k tomu, že emitorové napětí T_2 je vlastně výstupní napětí předcházejícího hradla stejného typu (ve stavu logické jedničky), musí být $U_1 > U_{B2}$ neboli $E > 5U_D$. Minimální napětí logické jedničky musí tedy být $U_1 > 3U_D$.

Na TTL hradlo v uvedeném zapojení může být zapojeno až 15 dalších hradel stejného typu. Typické zpoždění při průchodu hradla je ≥ 10 ns a výkonová ztráta ≥ 15 mW.

Dvojčinný stupeň neumožňuje propojení několika výstupů popsaných hradel NAND paralelně. K tomu je zapotřebí dalšího hradla typu OR. Toto hradlo však můžeme vypustit, jestliže použijeme hradel s tzv. otevřeným kolektorem.

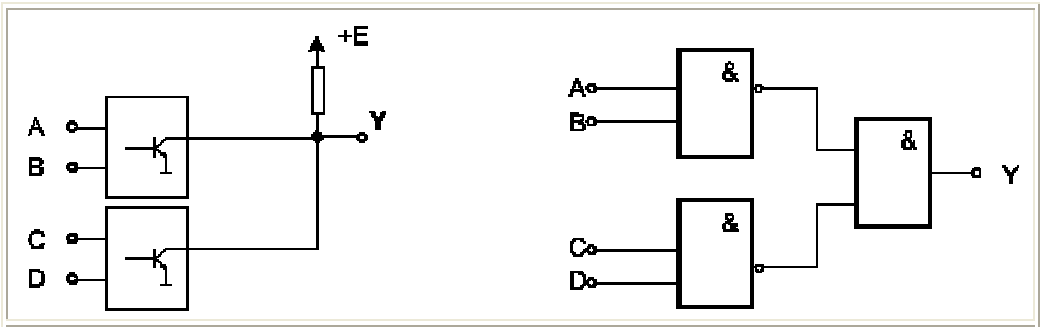


Principiální zapojení hradla NAND s otevřeným kolektorem

Do výstupního obvodu je nutné zapojit vnější odpor, na který však můžeme připojit další hradlo s otevřeným kolektorem.

Spojením několika výstupů s otevřeným kolektorem přes vnější odpor vznikne funkce "montážního AND". Protože však většinou spojujeme negované výstupy hradel (funkce NAND) vytvoří se (použitím transformace pomocí de Morganova teorému) funkce "montážního OR" (wired OR). Příklad paralelního zapojení dvou hradel a ekvivalentní schema zapojení je na následujícím obrázku.Logická funkce je dána následujícím Booleovským výrazem:

$$Y = \overline{AB} \cdot \overline{BD} = \overline{AB + BD}$$

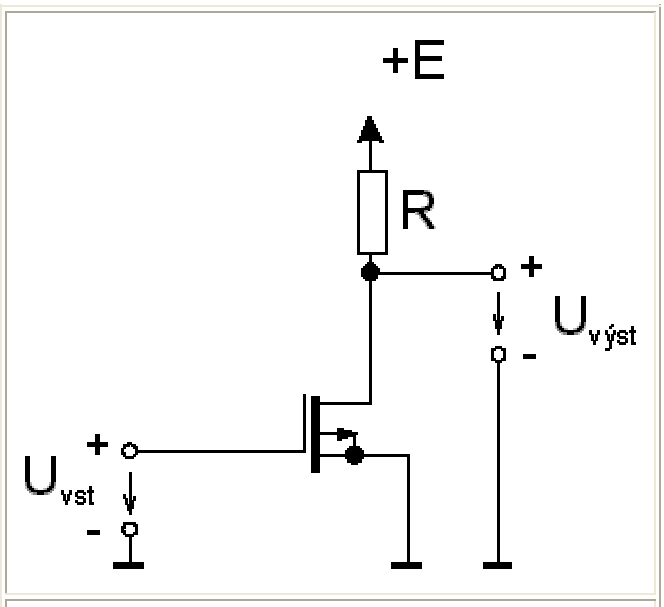


Příklad paralelního zapojení dvou hradel a ekvivalentní schema zapojení

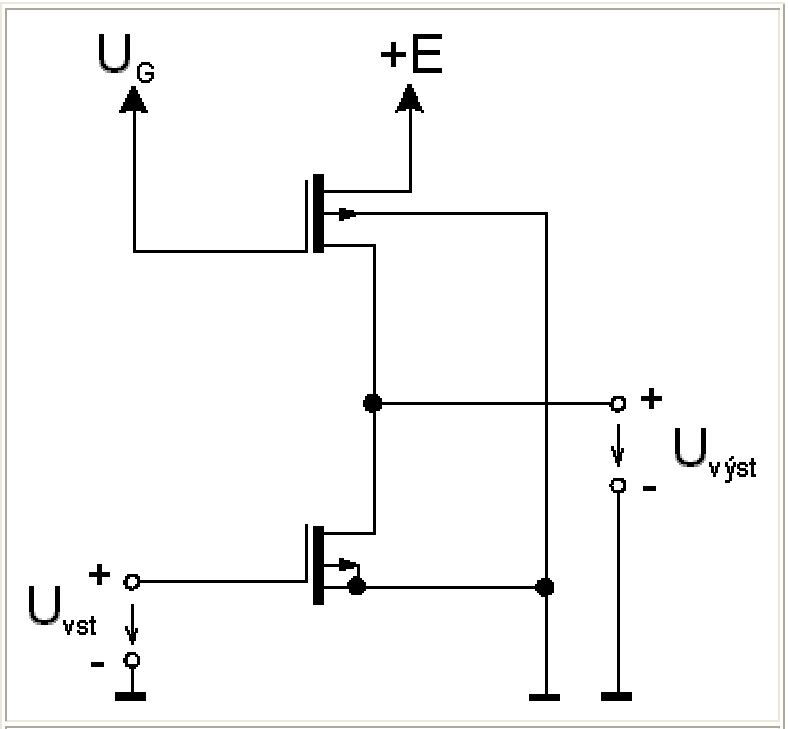
2.3 Systémy MOS/CMOS

Jednou z podmínek k opravdu masovému rozšíření mikroelektroniky je malá spotřeba zařízení, která umožňuje napájet přístroj z baterií. Možnost malé spotřeby otevřely logické systémy s tranzistory řízenými polem MOSFET a zejména systémy CMOS.

Základem systémů MOS je opět invertor, ovšem s tranzistorem řízeným elektrickým polem. Používají se tranzistory s indukovaným kanálem, které mají vhodnou polaritu tzv. prahového napětí. Prahové napětí U_p je napětí na řídicí elektrodě G (hradlu) tranzistoru, při kterém protéká transistorem určitý malý definovaný proud (např. 10 nA); tranzistor je při tom napájen definovaným napětím, např. 10 V. Je zřejmé, že v zájmu zjednodušení napájení zařízení je třeba, abychom tranzistor "zavřeli" napětím stejné polarity, jako je napětí zdroje. Tuto vlastnost mají právě MOS tranzistory s obohaceným kanálem. Invertor je možné realizovat s odporovou zátěží, jak je znázorněno na následujícím obrázku, avšak z hlediska jednoduchosti technologie výroby je mnohem jednodušší realizovat zátěž pomocí dalšího tranzistoru MOS, jehož elektroda G je na pevném napětí U_G, které může být menší nebo rovno napětí zdroje E.

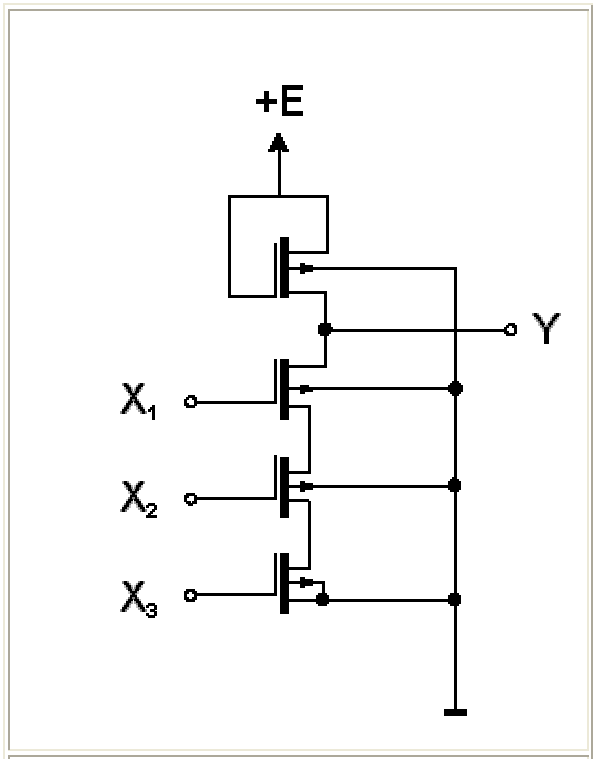


Invertor s odporovou zátěží

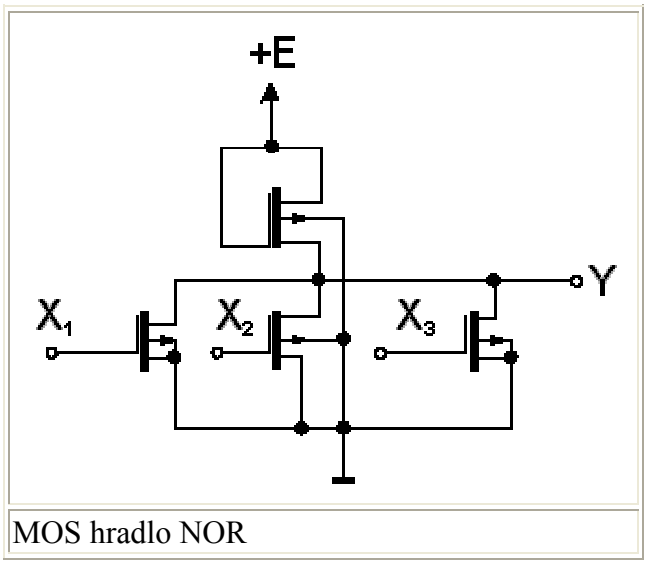


Invertor se dvěma tranzistory MOS

Napájecí napětí bývá 12 V, což je dáno velikostí prahového napětí - okolo 4 V. Aby bylo možno snížit napájecí napětí, byla vypracována řada technologií, které snižují prahové napětí až na 1.5 - 2V (např. technologie MNOS, která pro izolací G elektrody užívá kombinace vrstev nitridu a kysličníku křemíku, technologie "silicon gate MOS", kde se pro ovládací elektrodu používá polykrystalický křemík obohacený bórem, technologie RMOS s molybdenovou ovládací elektrodou apod.). Kombinací MOSFETových spínačů je možné konstruovat jak hradlo NAND, tak hradlo NOR.



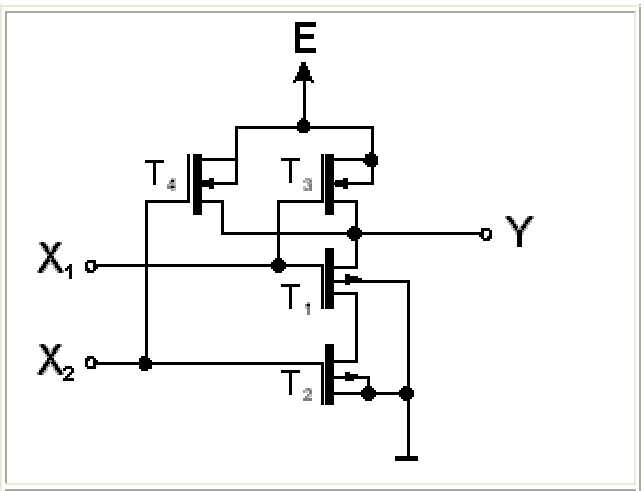
MOS hradlo NAND



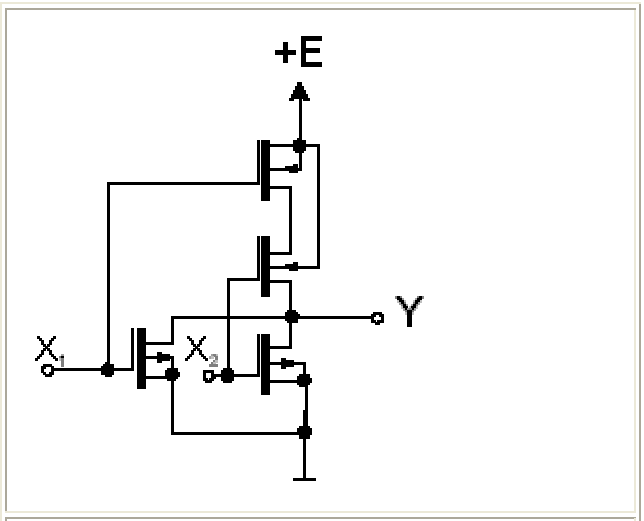
MOS hradlo NOR

Obvody s tranzistory MOS sice nevyžadují k ovládání prakticky žádný proud (kromě proudu nabíjejícího parazitní kapacity mezi řídicí elektrodou a kanálem), avšak v sepnutém stavu odebírají ze zdroje proud, který se bezúčelně tratí v zátěži. Významným krokem ke snížení spotřeby, umožněným zejména rozvojem technologie výroby integrovaných MOSových obvodů proto byla možnost realizace tranzistorů MOS s obohacenými kanály typu P a typu N na jediném čipu a tím i možnost realizace komplementárních MOSových obvodů (CMOS). Invertor CMOS se liší od invertoru typu MOS tím, že jeho zátěž je tvořena MOS tranzistorem opačné polarity a řídicí elektrody obou tranzistorů jsou spojeny. Prahová napětí jsou volena tak, aby při vstupním napětí rovném logické nule nebo jedničce byl vždy otevřen pouze jeden z obou

tranzistorů. Tranzistory tak fungují prakticky jako spínače, které připínají výstup buď na napájecí napětí E, nebo k zemi. Je zřejmé, že pokud nezatěžujeme výstup takového obvodu, je jeho spotřeba v klidovém stavu prakticky nulová. Výstup obvodu má relativně velice malou impedanci v obou stavech (řádově stovky ohmů), což umožňuje propojovat jednotlivá pouzdra s obvody běžnou technikou tištěných spojů. Jednoduchost obvodů je umožňuje sdružovat ve značné hustotě na čipu a vytvářet tak obvody velké a extrémně velké integrace. Např. operační paměti počítačů se dnes vyrábějí téměř výhradně technologií MOS nebo CMOS (průměrná doba vybavení informace z paměti MOS je okolo 20-100 ns). Rovněž naprostá většina dnes vyráběných mikroprocesorů využívá systému MOS nebo CMOS. Jako příklad uvádíme na obr. 7.18.a,b hradla NAND a NOR v systému CMOS.



CMOS hradlo NAND



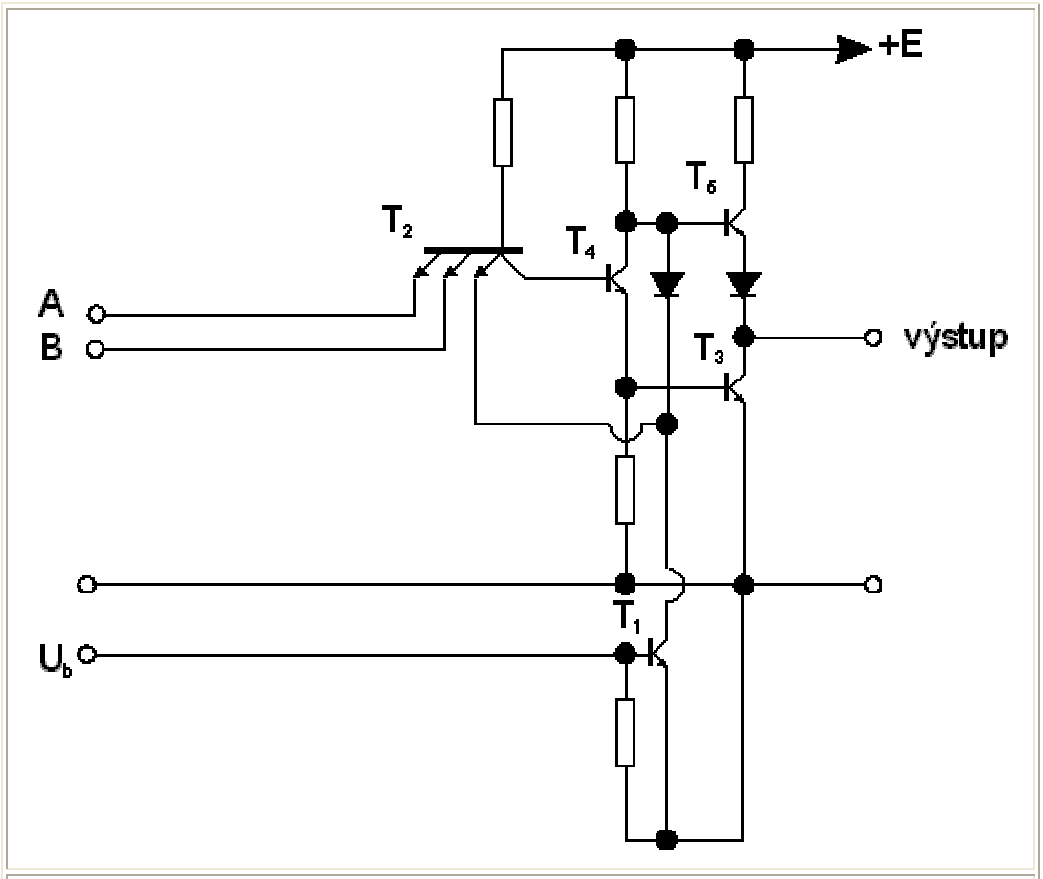
CMOS hradlo NOR

Je třeba ještě poznamenat, že ve skutečných obvodech MOS a CMOS je nutno chránit vstupní elektrodu před průrazem vysokým statickým napětím. Nejběžnější ochrana je pomocí Zenerovy diody, která je zapojena mezi substrát (normálně uzemněný) a řídící elektrodu. Tato dioda v normálním režimu nevede a otevírá se pouze dosáhne-li napětí na řídící elektrodě určité hodnoty.

V integrovaném obvodu je ochrana nutná ovšem pouze pro vstup těch hradel, které jsou vyvedeny ven z pouzdra.

2.4 Logická hradla s třemi stavy

V některých případech při spojování výstupů hradel je výhodné používat tzv. třístavových logických členů, kdy vedle výstupních aktivních stavů na úrovni logické nuly a jedničky existuje ještě třetí stav, kdy výstup hradla je v podstatě od sběrnice odpojen (připojen ke sběrnici přes velkou impedanci). Tento stav umožňuje stejně jako hradlo s otevřeným kolektorem připojení výstupů hradel do jednoho bodu.
Vedle dvou aktivních vstupů A a B má hradlo blokovací vstup U_B. Vybuzením tranzistoru T_1 se uzavřou tranzistory T_3 až T_5 a hradlo má velkou výstupní impedanci.



Principiální zapojení třístavového hradla NAND

Kontrolní otázky:

4. Proč musí být mezi logickými úrovněmi určité zakázané pásmo?
5. Lze kombinovat v jenom logickém obvodu použití pozitivní a negativní logiky?

Úkoly k zamyšlení:

2. Zamyslete se nad použitím amplitudového a kmitočtového způsobu kódování logických úrovní?

Shrnutí obsahu kapitoly

V této kapitole jste se seznámili s principy zobrazení logických úrovní pomocí elektrických signálů. Důraz v této kapitole byl kladen na pochopení možnosti různých způsobů kódování logických úrovní.

3 Základní logické členy

V této kapitole se dozvíte:

- Jaké jsou základní logické členy a jejich schematické značky?

Po jejím prostudování byste měli být schopni:

- Charakterizovat funkce základních logických členů.
- Porozumět způsobu kreslení schematických značek logických členů.

Klíčová slova této kapitoly:

Invertor, AND, OR, NAND, NOR, XOR, NOXOR.

Doba potřebná ke studiu: 1 hodina

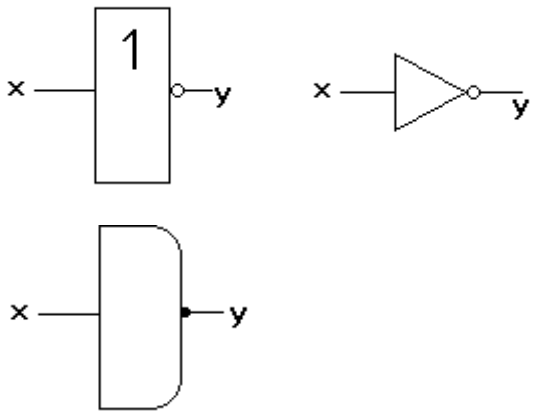
Průvodce studiem

Studium této kapitoly je jednoduché a seznámíte se s logickými členy a jejci způsobu kreslení ve schematech.
Na studium této části si vyhraďte 1 hodinu.

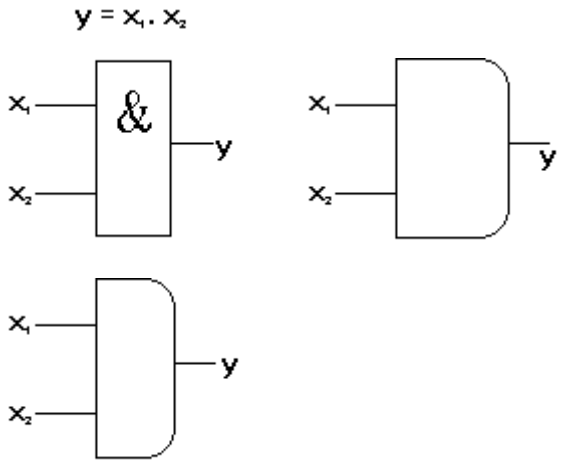
3.1 Invertor

USA **DIN**

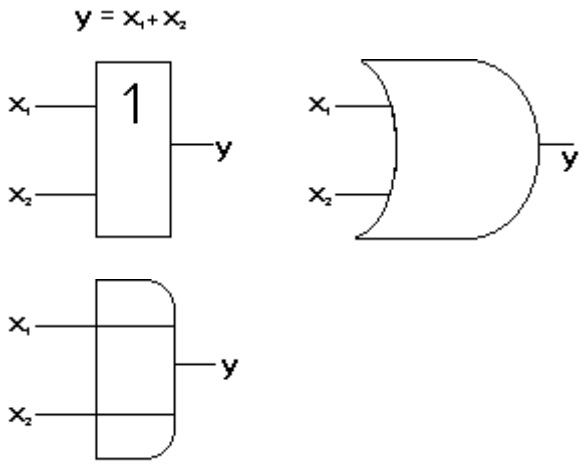
$y = \overline{x}$



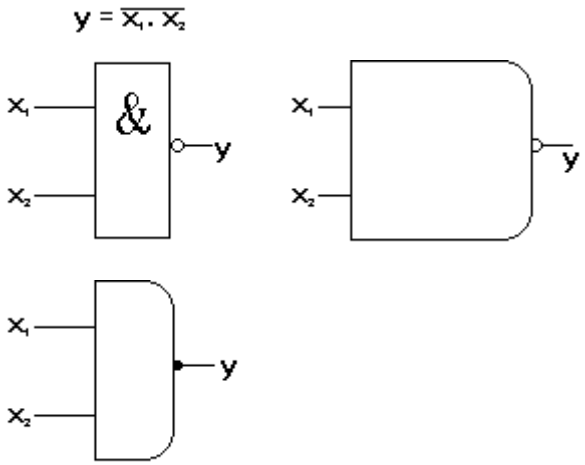
3.2 AND



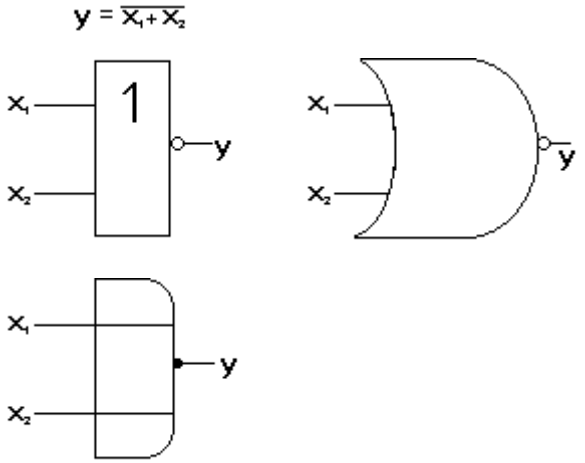
3.3 OR



3.4 NAND

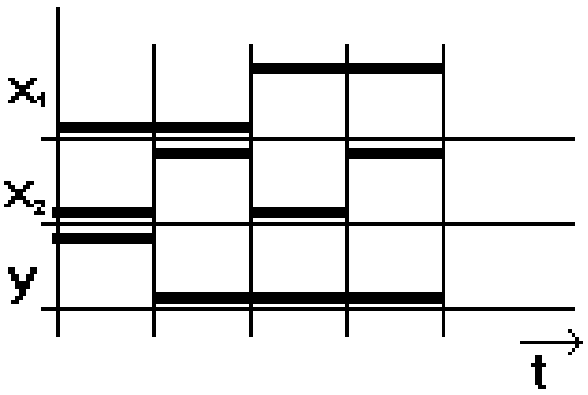


3.5 NOR



Př.: NOR

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

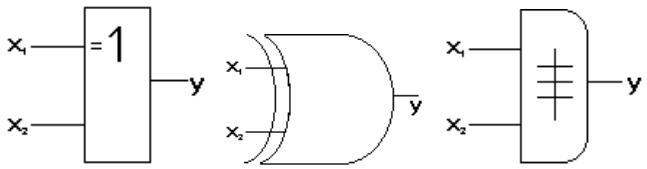


3.6 Ostatní logické členy

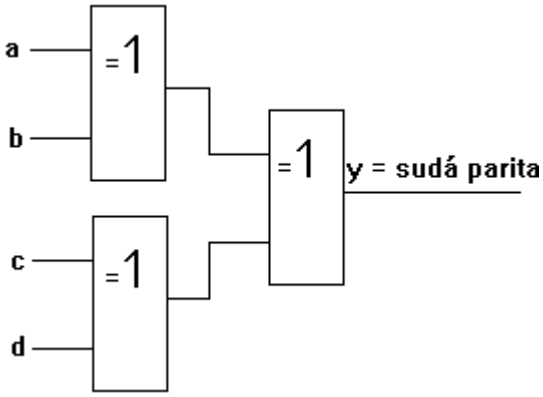
3.6.1 Nonekvivalence – XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

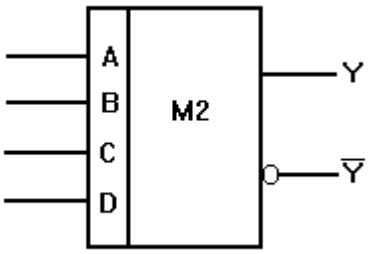
$y = x_1 \oplus x_2$



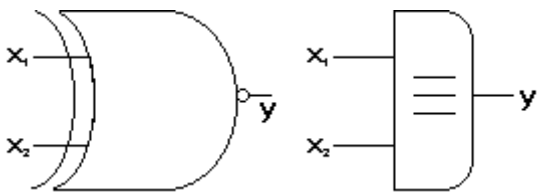
Př. Generátor parity: $y = a \oplus b \oplus c \oplus d = (a \oplus b) \oplus (c \oplus d)$



Schématická značka:



3.6.2 Ekvivalence - NOXOR



Kontrolní otázky:

6. Odvoďte pravdivostní tabulku generátoru parity 4 vstupních proměnných z předchozího příkladu?

Úkoly k zamyšlení:

3. Zamyslete se, které typy základních logických členů stačí pro vytváření logických sítí?

Korespondenční úkol:

2. Nakreslete logické schema obvodu realizující funkci

$$f = B\,\overline{C}\,D + A\,\overline{B} + \overline{B}\,\overline{D}$$

Využijte

a. členů NOT, OR, AND
b. členů NAND

Shrnutí obsahu kapitoly

V této kapitole jste se seznámili se základními logickými členy a jejich schematickými značkami.

4 Kombinační logické obvody

V této kapitole se dozvíte:

- Jak jsou realizovány a jaké mají funkce vybrané kombinační logické obvody?

Po jejím prostudování byste měli být schopni:

- Charakterizovat logické funkce základních kombinačních logických obvodů.
- Znát pravdivostní tabulky a zapojení těchto kombinačních logických obvodů.
- Porozumět způsobu vytváření kombinačních logických obvodů.

Klíčová slova této kapitoly:

Multiplexor, dekodér, sčítačka.

Doba potřebná ke studiu: 2 hodiny

Průvodce studiem

Studium této kapitoly je důležité pro pochopení funkcí základních částí logiky počítače a proto si na studium této části si vyhraďte 2 hodiny.Po celkovém prostudování a vyřešení všech příkladů doporučujeme vypracovat korespondenční úkol.

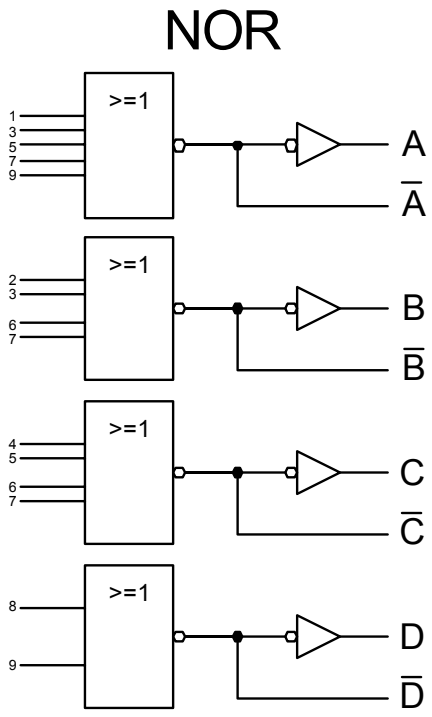
Kombinační logické obvody provádí přímou transformaci vstupních logických proměnných na výstupní. Realizují určitou logickou funkci kterou můžeme popsat např. pomocí pravdivostní tabulky.
Definujme si základní rozdíly mezi kombinačními a sekvenčními obvody:
Kombinační obvod:
– výstupy jsou závislé pouze na vstupních kombinacích a ne na jejich předchozích hodnotách,
– jediné kombinaci vstupních hodnot odpovídá jediná výstupní kombinace.
Sekvenční obvod:
– hodnota výstupní veličiny závisí nejen na okamžité kombinaci hodnot vstupních veličin , ale i na posloupnosti hodnot vstupních veličin jenž jsou uchovány v paměťových členech, v podobě vnitřních signálů logických obvodů, nebo-li v předcházejících časových okamžicích.

4.1 Kódování, dekódování a převody kódů

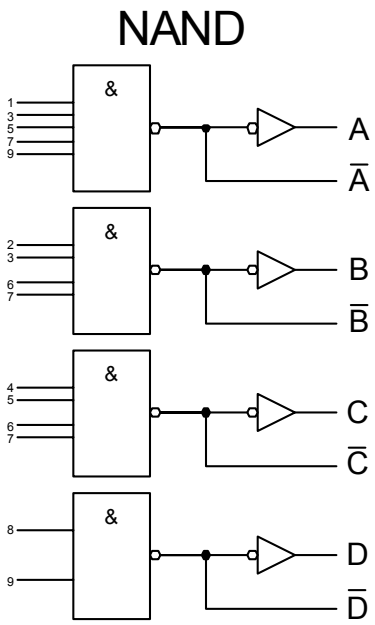
4.1.1 Převod desítkových číslic do kódu 8421

	kód 8 4 2 1				kód 1 z 10									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	1	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0	1	0	0	0	0
6	0	1	1	0	0	0	0	0	0	0	1	0	0	0
7	0	1	1	1	0	0	0	0	0	0	0	1	0	0
8	1	0	0	0	0	0	0	0	0	0	0	0	1	0
9	1	0	0	1	0	0	0	0	0	0	0	0	0	1
10	1	0	1	0	} nepoužité kombinace									
11	1	0	1	1										
12	1	1	0	0										
13	1	1	0	1										
14	1	1	1	0										
15	1	1	1	1										

Obr. Kombinační tabulka

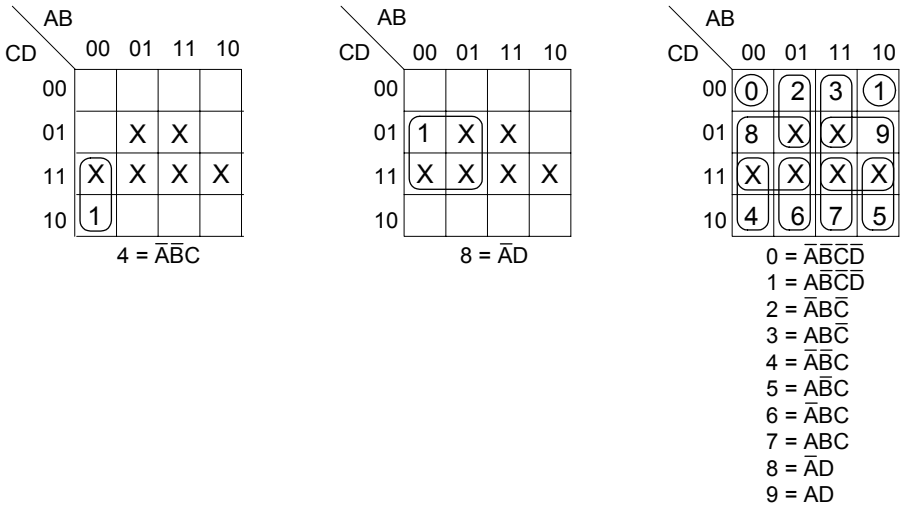


Obr. Výsledné zapojení s obvody NOR

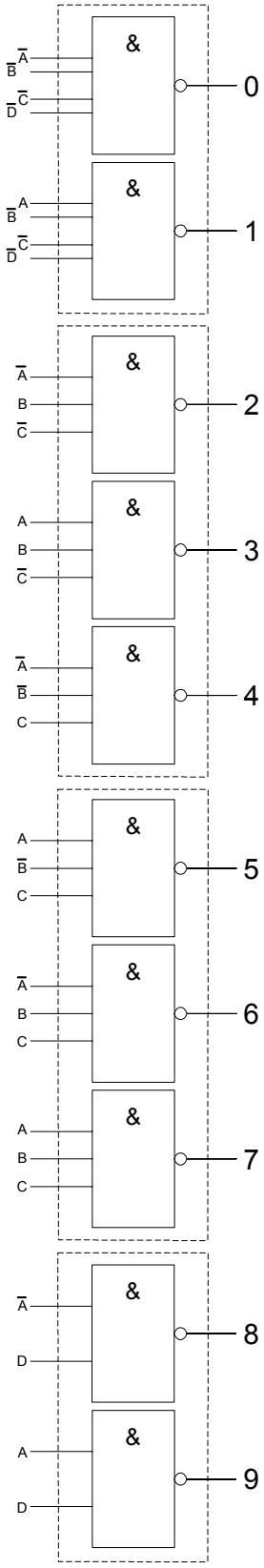


Obr. Výsledné zapojení s obvody NAND

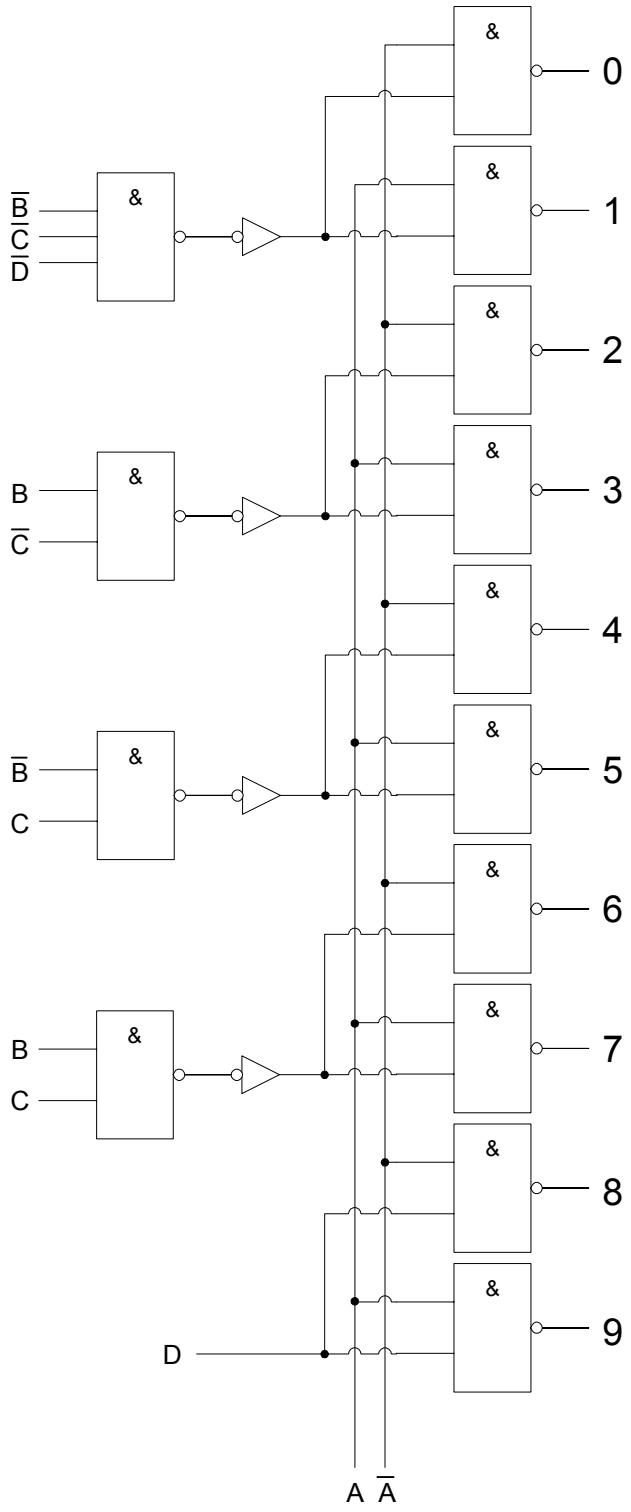
4.1.2 Převod z kódu 8421 na desítkové číslice



Obr. Minimalizace funkcí



Obr. Realizace zapojení s obvody NAND



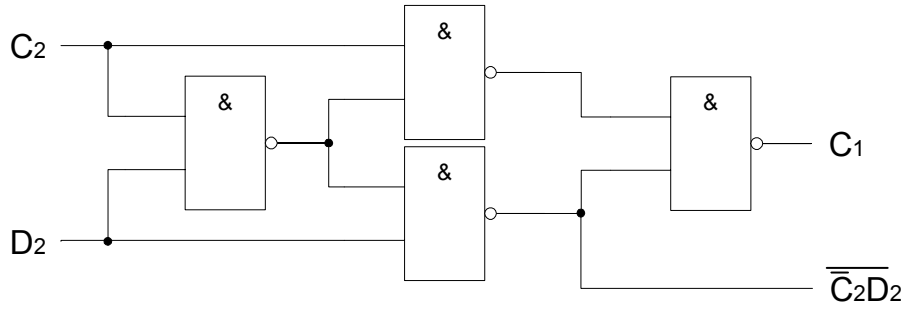
Obr. Upravené zapojení pro obvody NAND s menším počtem vstupů

4.1.3 Převod do Grayova kódu

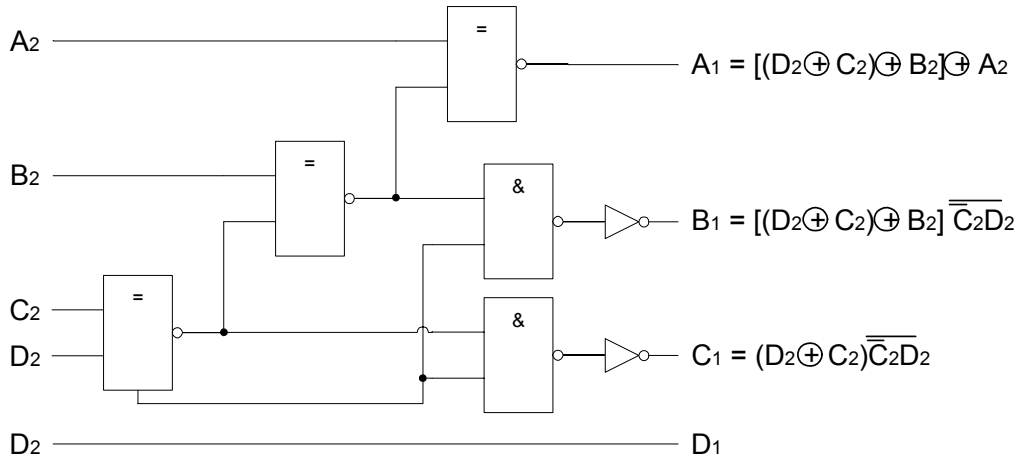
Grayův kód má tu vlastnost , že se při přechodu od jednoho kódového slova ke druhému mění vždy jen hodnota jednoho bitu.

	8	4	2	1	Gray			
	D_1	C_1	B_1	A_1	D_1	C_1	B_1	A_1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	0	0	0

Obr. Kombinační tabulka



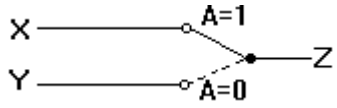
Obr. Upravený obvod EXL-OR použitý ve výsledném zapojení



Obr. Výsledné zapojení s obvody EXL-OR

4.1.4 Multiplexor

$$Z = A.\,X + \overline{A}\,.\,Y$$

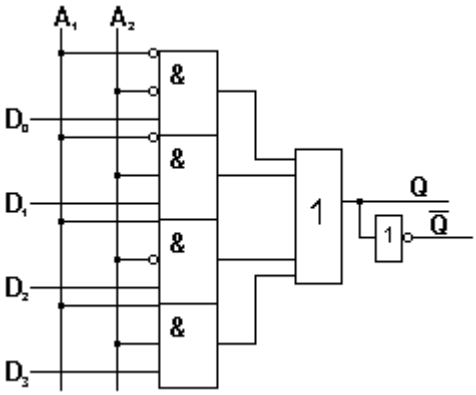


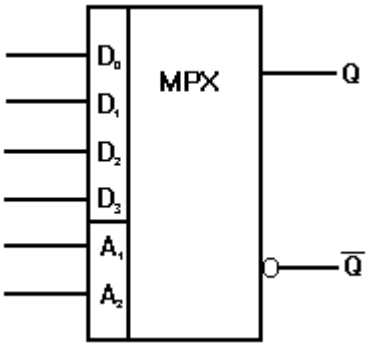
4.1.5 4-vstupý multiplexor

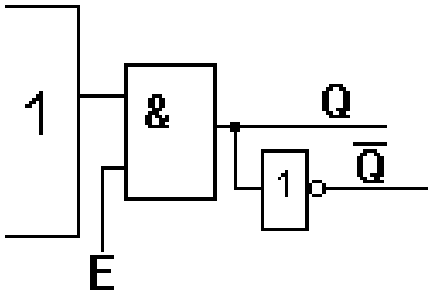
4 datové vstupy, 2 adresové vstupy

A_1	A_2	Q
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

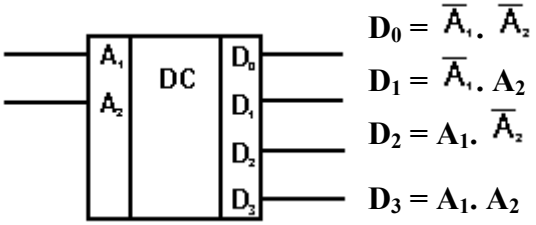
$$Q = \overline{A_1}.\,\overline{A_2}.\,D_0 + \overline{A_1}.\,A_2.\,D_1 + A_1.\,\overline{A_2}.\,D_2 + A_1.\,A_2.\,D_3$$







4.1.6 Dekodér

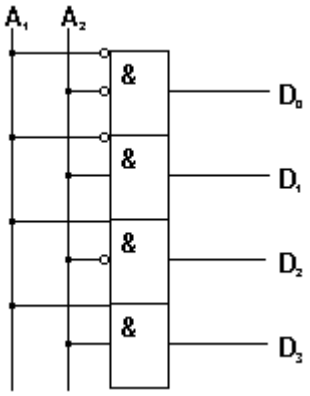


$D_0 = \overline{A_1} . \overline{A_2}$

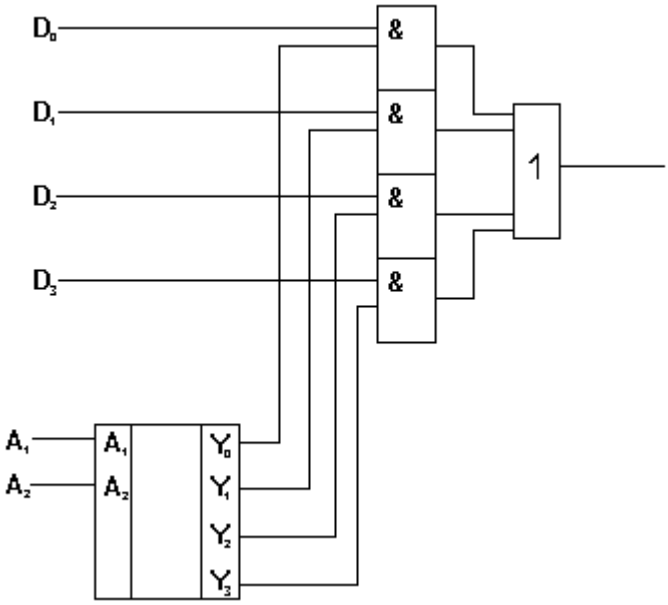
$D_1 = \overline{A_1} . A_2$

$D_2 = A_1 . \overline{A_2}$

$D_3 = A_1 . A_2$



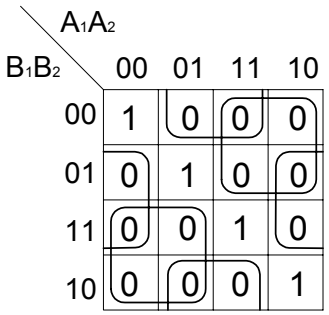
Realizace MPX pomocí dekodéru:



4.2 Porovnání dvojkových informací

4.2.1 Rovnost dvou proměnných

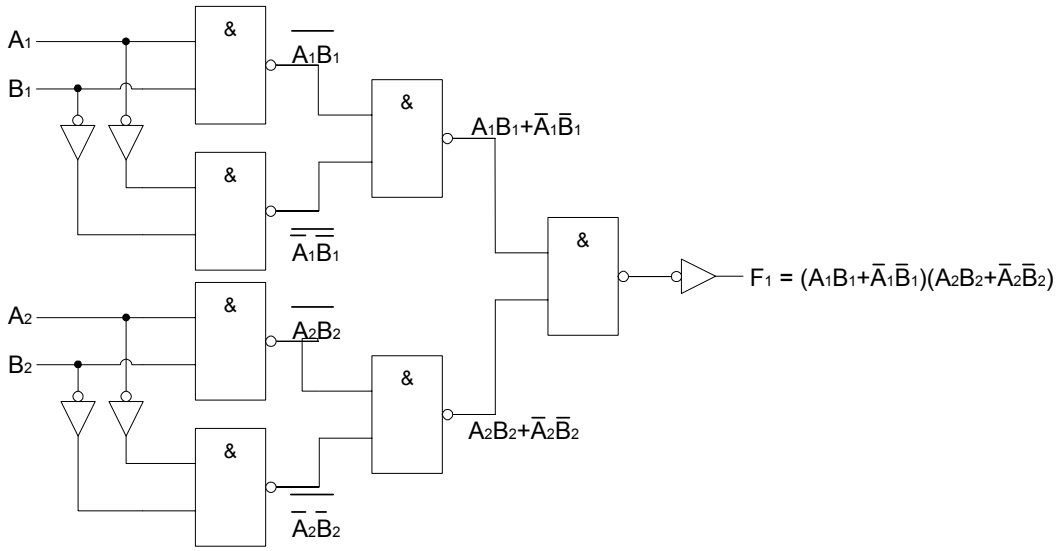
B_2	B_1	A_2	A_1	F_1
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



$\overline{F}_1 = A_1\overline{B}_1+\overline{A}_1B_1+A_2\overline{B}_2+\overline{A}_2B_2$

$F_1 = (A_1B_1+\overline{A}_1\overline{B}_1)(A_2B_2+\overline{A}_2\overline{B}_2)$

Obr. Kombinační tabulka a minimalizace



Obr. Výsledné zapojení

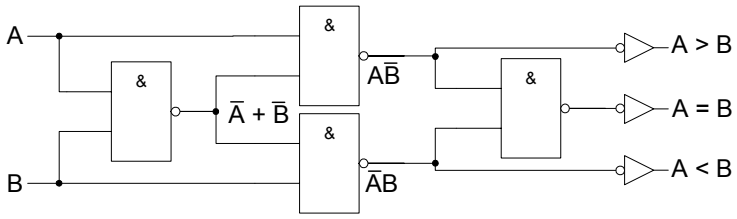
4.2.2 Jednobitový porovnávací obvod

B	A	F1 A = B	F2 A > B	F3 A < B
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

$F_1 = AB + \overline{A}\,\overline{B}$

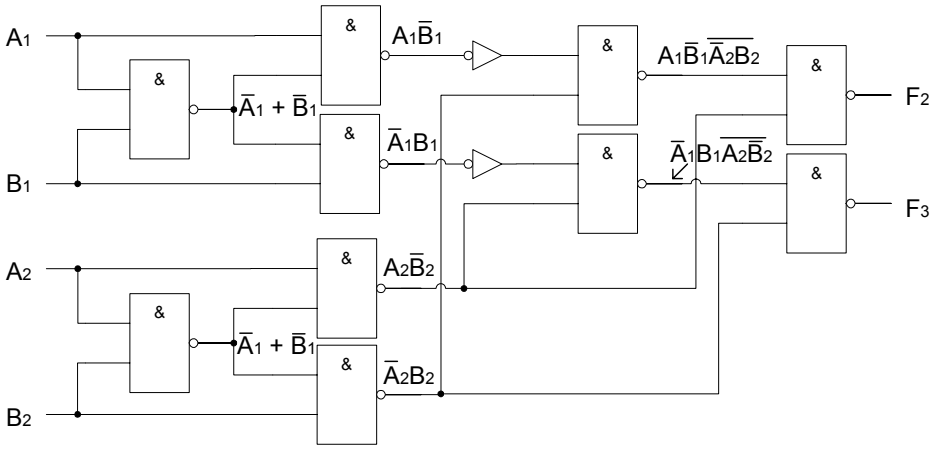
$F_2 = A\overline{B}$

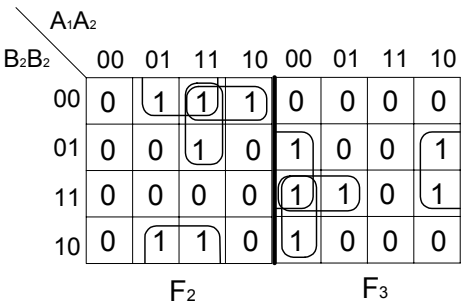
$F_3 = \overline{A}B$



Obr. Kombinační tabulka a zapojení

4.2.3 Dvoubitový porovnávací obvod





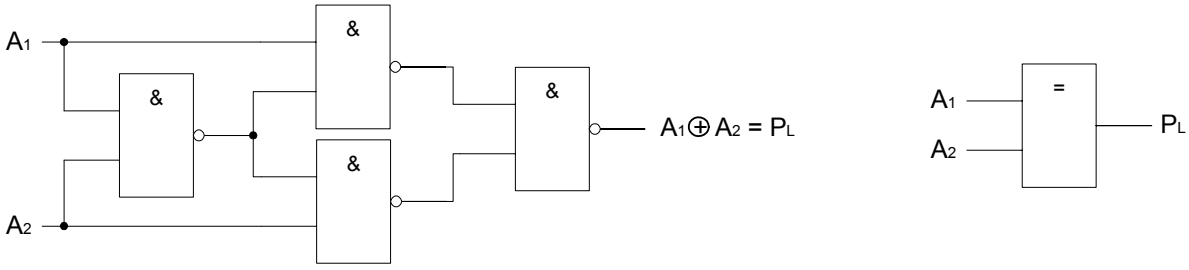
$F_2 = A_2\bar{B}_2 + A_1A_2\bar{B}_1 + A_1\bar{B}_1\bar{B}_2$

$F_3 = \bar{A}_2B_2 + \bar{A}_1\bar{A}_2B_1 + \bar{A}_1B_1B_2$

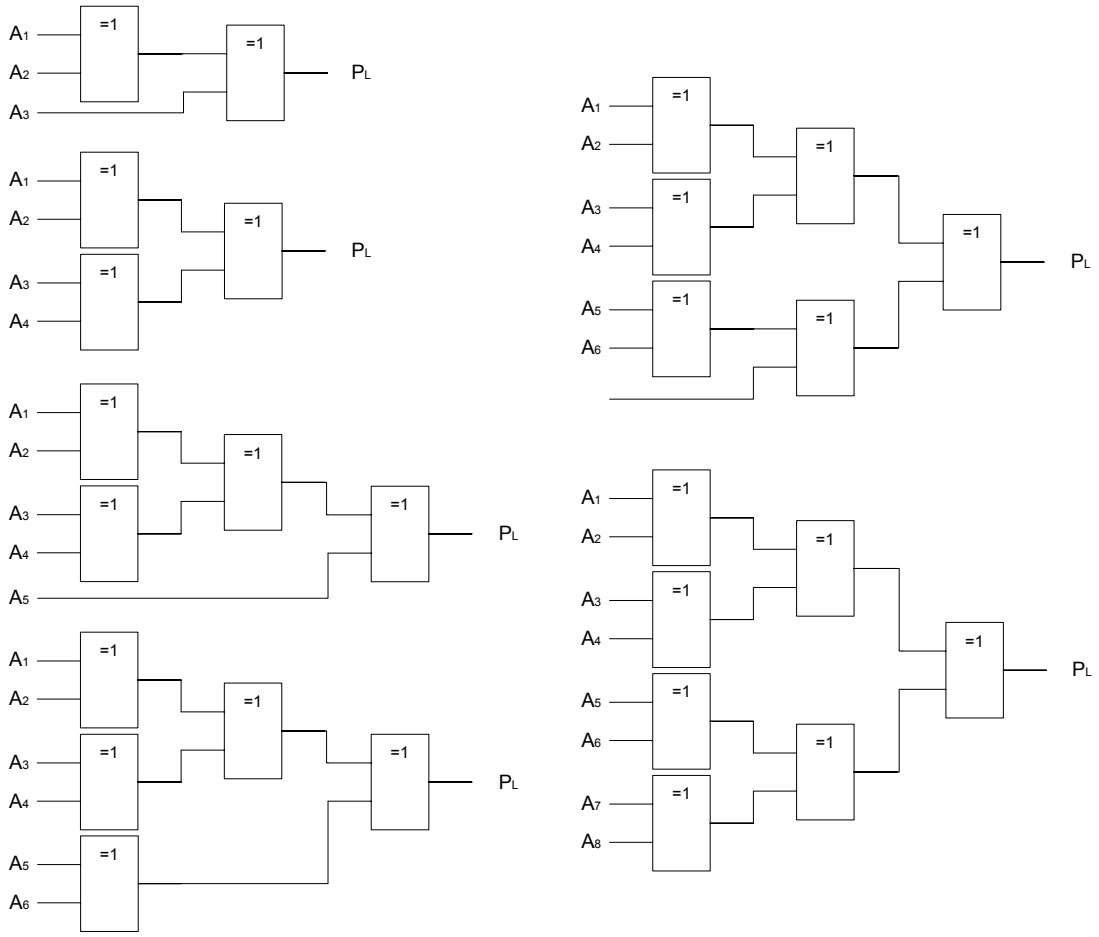
				F_1	F_2	F_3
B_2	B_1	A_2	A_1	A = B	A > B	A < B
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

Obr. Kombinační tabulka, minimalizace a výsledné zapojení

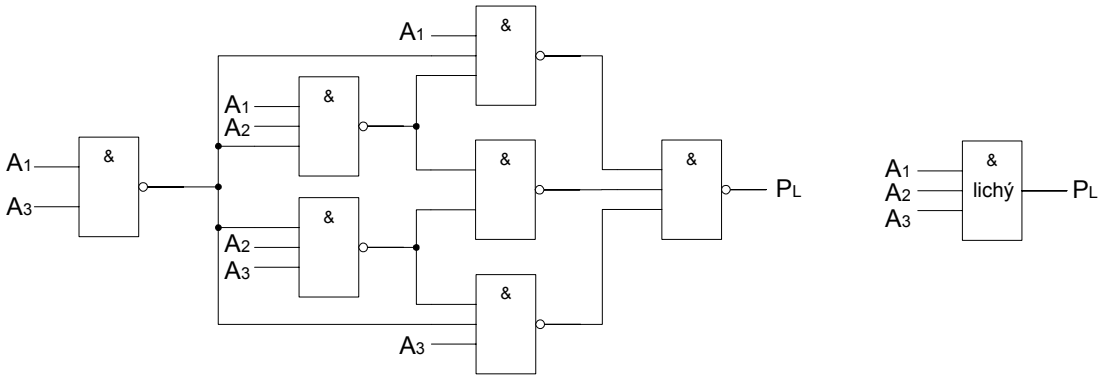
4.3 Parita



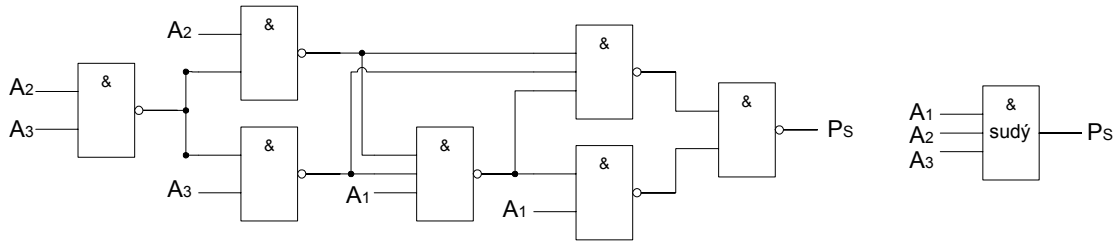
Obr. Zapojení a schematická značka základního zapojení liché parity



Obr. Zapojení pro různý počet bitů



Obr. Základní obvod pro kontrolu liché parity tří bitů



Obr. Základní obvod pro kontrolu sudé parity tří bitů

4.4 Sečítačky a odečítačky

4.4.1 Sčítačka MODULO 2

Činnost sčítačky modulo 2 je dána následující tabulkou. Jednotlivé číslice vstupující do součtu jsou x a y, z je číslice dvojkového aritmetického (nikoli logického) součtu

$$\mathbf{x + y = z.}$$

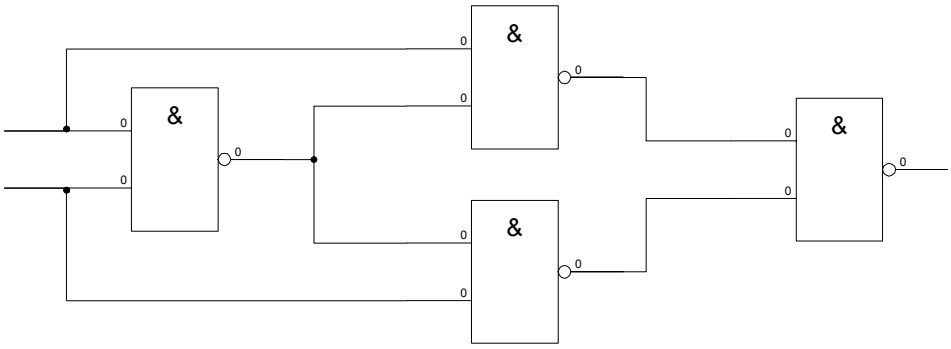
x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Vzhledem k tomu, že číslice x a y považujeme za logické proměnné, můžeme z považovat za výstupní funkci a vytvořit úplnou disjunktní normální formu

$$z = \bar{x}y + x\bar{y}$$

Při realizaci sčítačky z logických členů NAND je vhodné výraz pomocí algebraických úprav změnit na:

$$z = (x + y)\overline{xy}$$



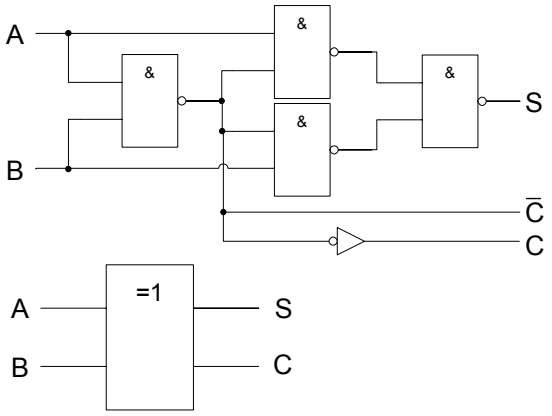
4.4.2 Jednobitová neúplná sečítačka

součet
přenos

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = A\bar{B} + \bar{A}B$

$C = AB$



4.4.3 Jednobitová úplná sečítačka

A	B	C_i	S	C_0
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

C_i \ AB	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

$S = AB\bar{C}_i + A\bar{B}\bar{C}_i + \bar{A}B\bar{C}_i + \bar{A}\bar{B}C_i =$

$= A\overline{(B \oplus C_i)} + A(B \oplus C_i) =$

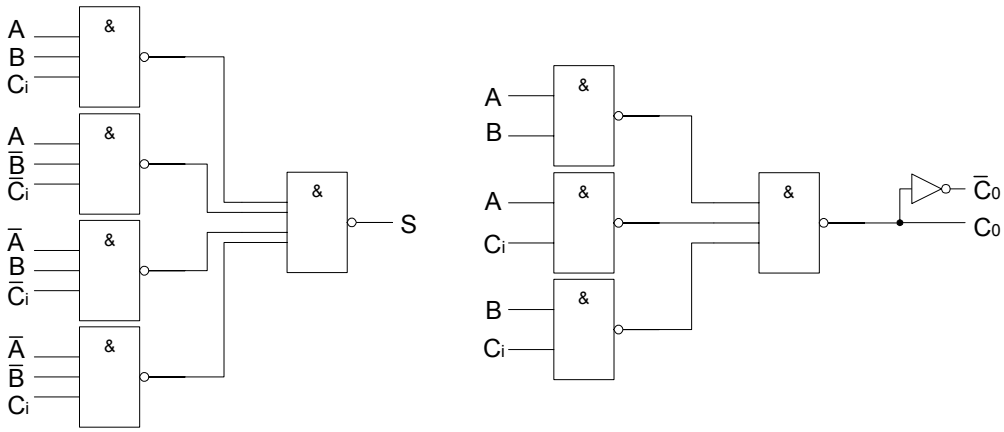
$= A \oplus B \oplus V_i$

V_i \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

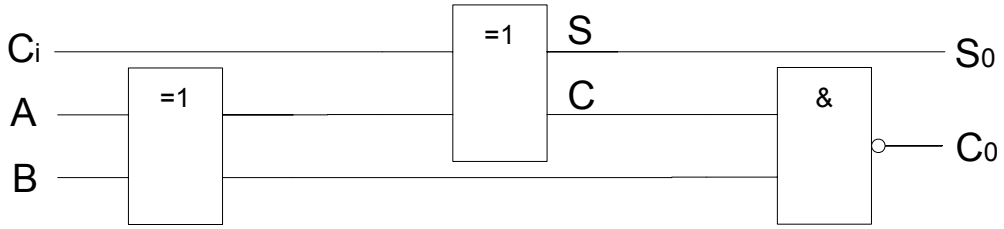
$C_0 = AB + C_i A + C_i B =$

$= AB + C_i (A + B)$

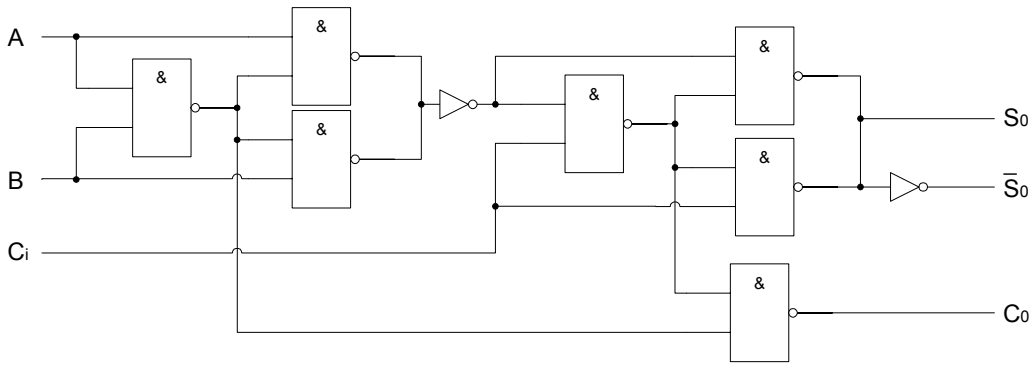
Obr. Kombinační tabulka a minimalizace



Obr. Zapojení s obvody NAND



Obr. Zapojení s neúplnými sečítačkami



Obr. Příklad zapojení

4.4.4 Jednobitová úplná odečítačka

A	B	V_1	D	V_0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

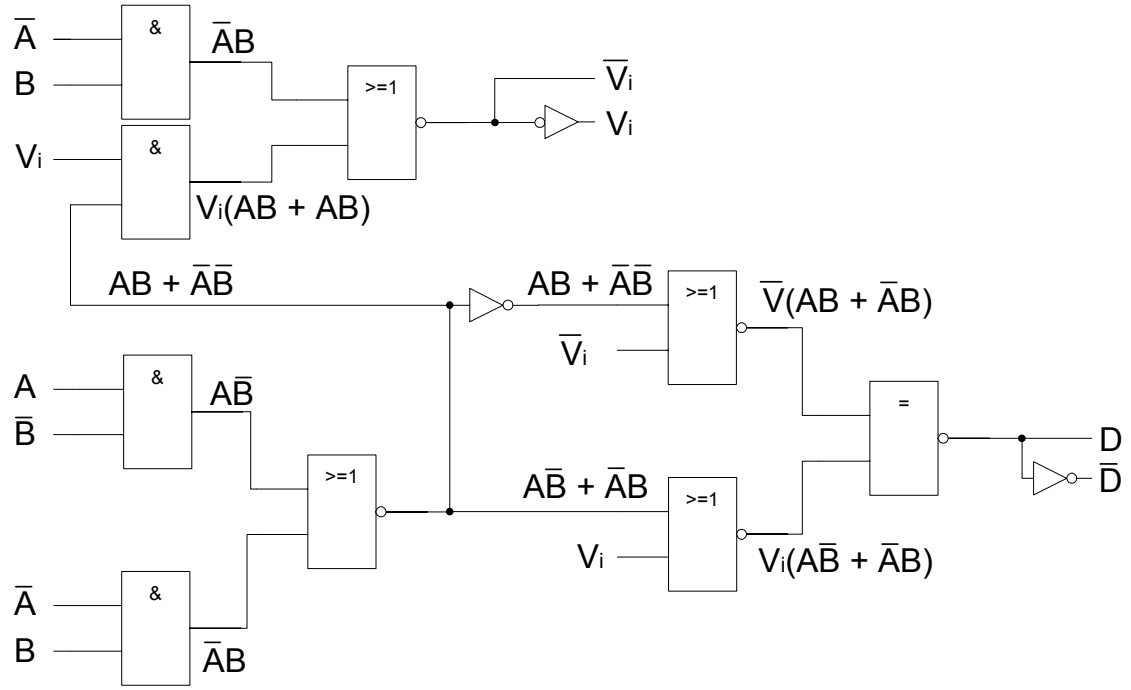
V_i \ AB	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

$D = ABV_i + A\overline{B}\overline{V_i} + \overline{A}B\overline{V_i} + \overline{A}\overline{B}V_i =$
$= A \oplus B \oplus V_i$

V_i \ AB	00	01	11	10
0	0	(1)	0	0
1	(1)	(1)	(1)	0

$V_0 = \overline{A}B + \overline{A}V_i + BV_i$

Obr. Kombinační tabulka a minimalizace



Obr. Příklad zapojení

4.4.5 Polosčítačka

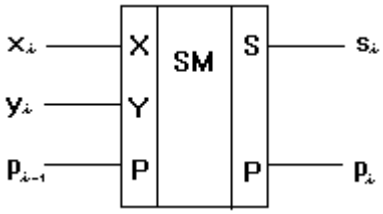
x	y	S	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$\mathbf{S = \overline{x} \cdot y + x \cdot \overline{y}}$

$\mathbf{P = x \cdot y}$

4.4.6 Úplná sčítačka

x_i	y_i	p_{i-1}	s_i	p_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



Kontrolní otázky:

7. Nakreslete logické schema úplné sčítačky?
8. Proč má úplná sčítačka 3 vstupy a 2 výstupy?

Korespondenční úkol:

3. Navrhněte kombinační logický obvod realizující dekodér sedmisegmentového zobrazovače hexadecimálních znaků.

Shrnutí obsahu kapitoly

V této kapitole jste se seznámili se základními kombinačními logickými obvody. Důraz v této kapitole byl kladen na pochopení vytváření logických obvorů. Velká pozornost byla věnována funkcím sčítačky.

5 Sekvenční logické obvody

V této kapitole se dozvíte:

- Jaké jsou základní sekvenční logické obvody?
- Z jakých logických prvků se sestavují registry?
- Jaké funkce a složení mají čítače?
- Jaké jsou realizovány seriové a paralelní vstupy a výstupy?

Po jejím prostudování byste měli být schopni:

- Charakterizovat základní typy klopných obvodů.
- Znát způsob návrhu sekvenčních logických obvodů.
- Porozumět funkcím registrů a čítačů.

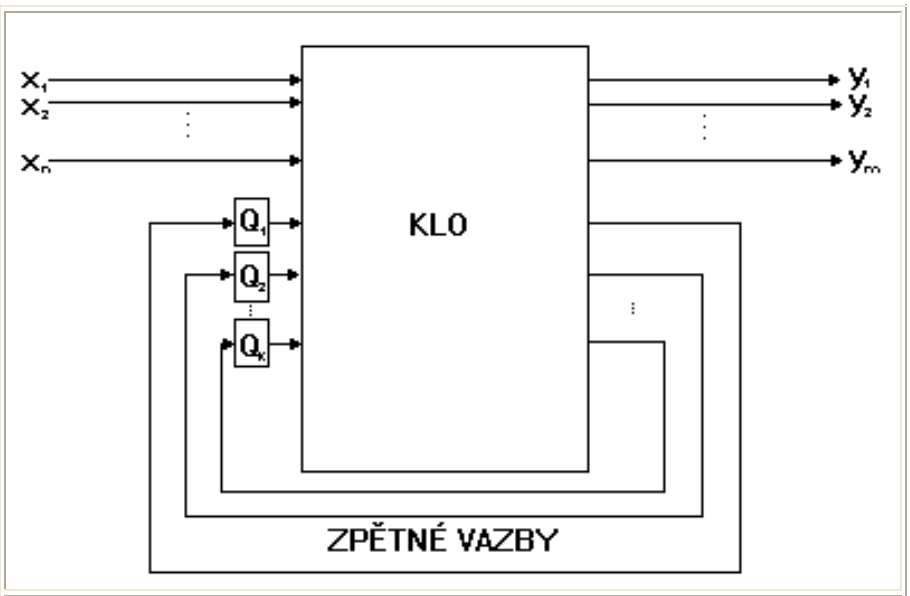
Klíčová slova této kapitoly:

Klopný obvod, registr, čítač, seriový vstup, paralelní vstup.

Doba potřebná ke studiu: 4 hodiny

Průvodce studiem

Studium této kapitoly je složitější oproti předchozím kapitolám. Pochopení funkce sekvenčních logických obvodů je důležité pro znalost dalších částí počítače.
Na studium této části si vyhraďte alespoň 4 hodiny.Po celkovém prostudování a vyřešení všech příkladů doporučujeme vypracovat korespondenční úkol.



Sekvenční obvod

Sekvenční obvody se dělí na:

- Synchronní - Jsou synchronizovány samostatnými signály , které se nazývají synchronizační , nebo hodinové a určují jednotlivé takty (časové intervaly) tj. diskrétní čas

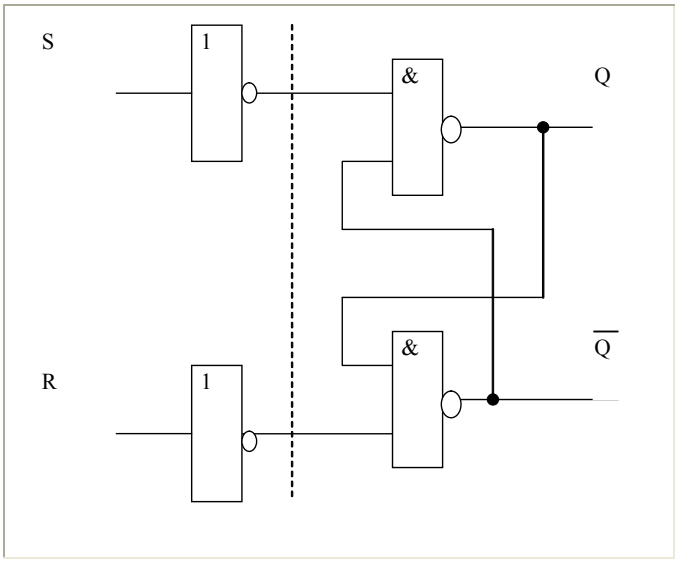
- Asynchronní - U asynchronních logických obvodů nastává okamžitá změna stavu na vstupu . Ze vstupních hodnot se vygeneruje výstupní signál

5.1 Klopný obvod RS

Základním obvodem, který je schopen setrvat v určitém stavu (logické 0 nebo 1) bez aplikace vnějších logických úrovní (mimo napájecí napětí ovšem) je tzv.paměťová buňka, neboli klopný obvod. Nejjednodušší klopný obvod vytvoříme pomocí dvou NAND členů, kterým křížem propojíme vstupy a výstupy. Pravdivostní tabulka takového klopného obvodu je uvedena na následujícím obrázku.

R	S	Q_i	$\overline{Q_i}$
0	1	1	0
1	0	0	1
0	0	Q_{i-1}	$\overline{Q_{i-1}}$
1	1	Zakázaný stav	

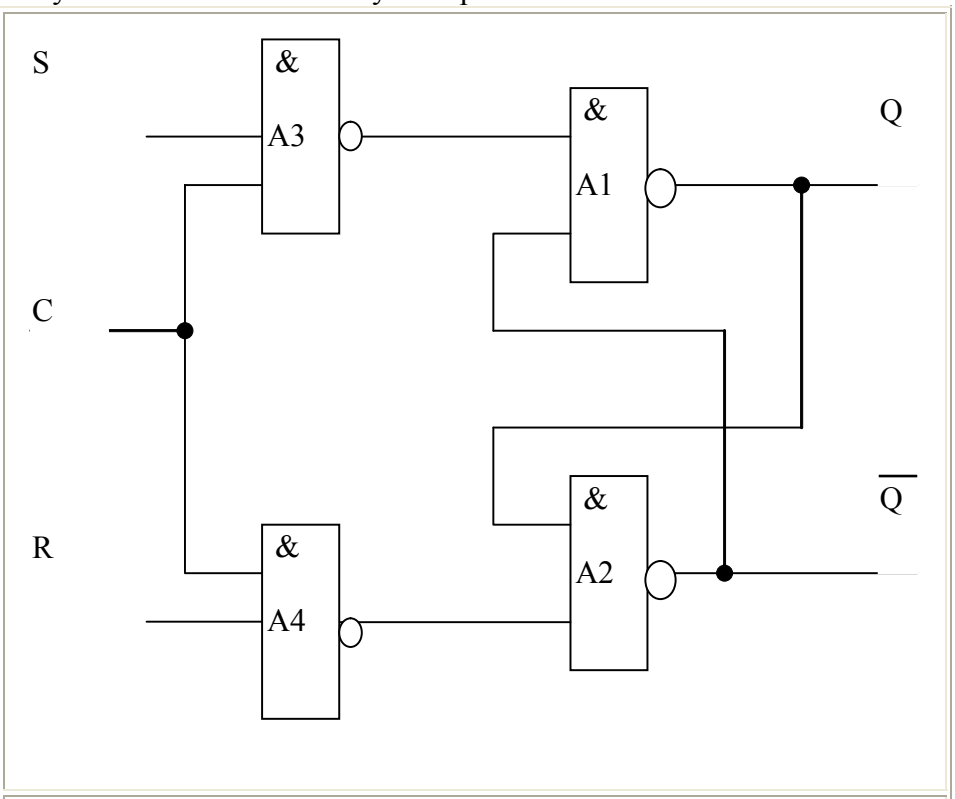
Zapojení RS klopného obvodu je uvedeno na následujícím obrázku.



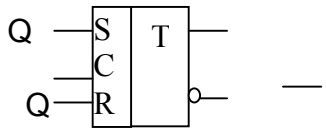
RS klopný obvod

V sekvenčním logickém systému je třeba, aby se nastavení nebo nulování klopných obvodů dále v synchronismu s hodinovými impulsy. To lze zabezpečit tak, že místo invertorů na předchozím obrázku použijeme dvoustupová hradla NAND, která otevíráme hodinovými impulsy. Je zřejmé,

že je-li úroveň na hodinovém vstupu log 0 (stav mezi dvěma impulsy), nezmění klopný obvod svůj stav ; pamatuje si jej po dobu mezi dvěma hodinovými impulsy. Jsou-li vstupy R a S uzemněny (je-li na nich logická nula), pak na výstupech řídících hradel je logická jednička nezávisle na úrovni hodinového vstupu. Výstupy Q a zůstávají proto na úrovni, do které se dostaly před tím, než jsme na oba vstupy R a S logickou nulu přivedli. Přivedením logické nuly na oba vstupy R i S zablokujeme tedy stav výstupů Q a , které jsou nyní nezávislé na hodinových impulsech.



RS klopný obvod s hodinovým vstupem



Předpokládejme nyní, že S = 1, R = 0. Po příchodu hodinového impulsu je pak na výstupu hradla A_3 logická 0 a na výstupu hradla A_4 logická jednička. Prostudujeme-li logické úrovně zapojení uvidíme, že obvod nastavíme do stavu, kdy Q = 1 a = 0. Tento stav setrvává do té doby, dokud v době příchodu hodinového impulsu je na vstupech S = 1 a R = 0, Ze symetrie obvodu okamžitě plyne, že pro S = 0 a R = 1, nastavíme obvod po příchodu hodinového impulsu do stavu Q = 0 a = 1.

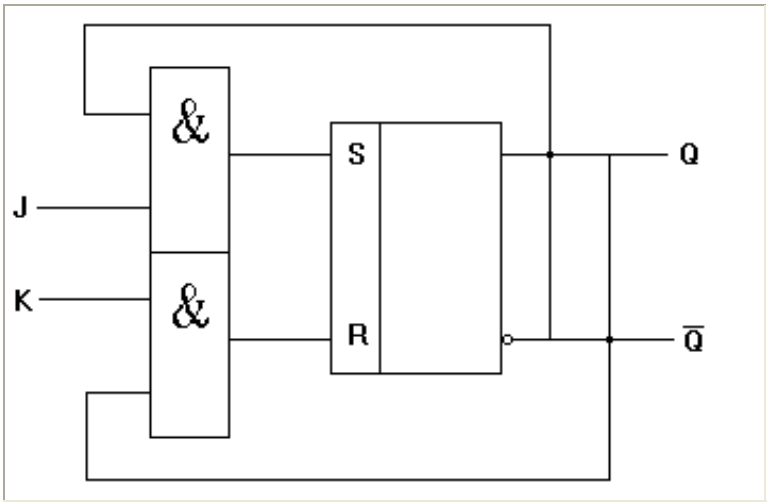
Je-li S = 1 = R, nastaví se výstupy hradel A_3 a A_4 po příchodu hodinového impulsu do stavu logické nuly. Tento stav by opět implikoval, že oba výstupy hradel A_1 a A_2 by měly být ve stavu logické jedničky, což je neslučitelné se zapojením klopného obvodu. Ve skutečnosti se stane to, že v závislosti na tom, který z výstupů hradel A_3 a A_4 stoupá po ukončení hodinového impulsu

rychleji, nastaví se na výstupech a Q buď stav Q = 1 nebo Q = 0. Stav výstupů proto nezáleží ani na stavech vstupů ani na úrovni hodinového impulsu, ale na vnitřních parametrech obvodu ; je to tedy neurčitý stav a v tabulce je znázorněn otazníkem. Proto se při buzení tohoto klopného obvodu musí pamatovat na to, aby stav R = S = 1 nemohl nastat.

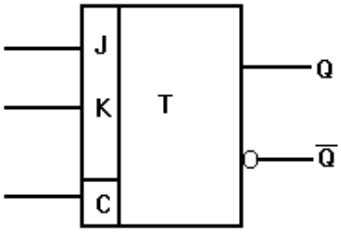
Kromě právě popsaného klopného obvodu R-S jsou používány tři další typy klopných obvodů : dvojčinné klopné obvody J-K, T a D. Dvojčinné klopné obvody J-K, D a T odstraňují neurčitý stav obvodu R-S. Obvod T pracuje jako binární obvod, který mění svůj stav po každém hodinovém impulsu.

5.2 Klopný obvod J-K

J	K	Q_i
0	1	0
1	0	1
0	0	$\overline{Q_{i-1}}$
1	1	Q_{i-1}



RealizaceJK klopného obvodu z RS klopného obvodu



Schematická značka JK klopného obvodu s hodinový vstupem

5.3 Klopný obvod typu T(RS-T)

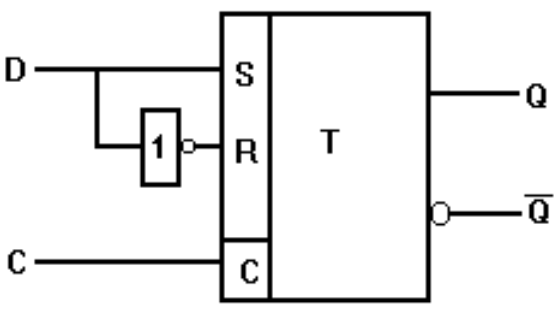
Mění svůj stav při každém hodinovém impulsu. Je tedy $Q_{n+1} = Q_n$. Z pravdivostní tabulky obvodu J-K můžeme vidět, že tuto funkci plní obvod J-K pro J = K = 1. Obvod typu T má tedy dva vstupy - vstup T (spojené vstupy J-K) a vstup pro hodinové impulsy. Je-li T = 1, obvod se překlápí, $Q_{n+1} = Q_n$, je-li T = 0, obvod zůstává překlopen do původního stavu; $Q_{n+1} = Q_n$. Tato funkce obvodu T je využita v synchronních čítačích - viz níže. Pokud nepotřebujeme obvod T elektricky ovládat, vystačíme s obvodem typu D, u něhož spojíme výstup se vstupem D. Snadno nahlédneme, že je pak $Q_{n+1} = Q_n$.

5.4 Klopný obvod typu D

Pravdivostní tabulka funkce D klopného obvodu je zřejmá z následujícího obrázku

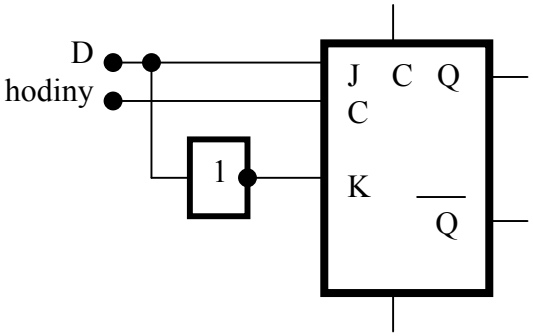
D	C	Q_i
1	⊓	1
0	⊓	0
?	_	Q_{i-1}

Realizace s využitím RS klopného obvodu je naznačena na následujícím obrázku



Realizace D klopného obvodu z RS klopného obvodu

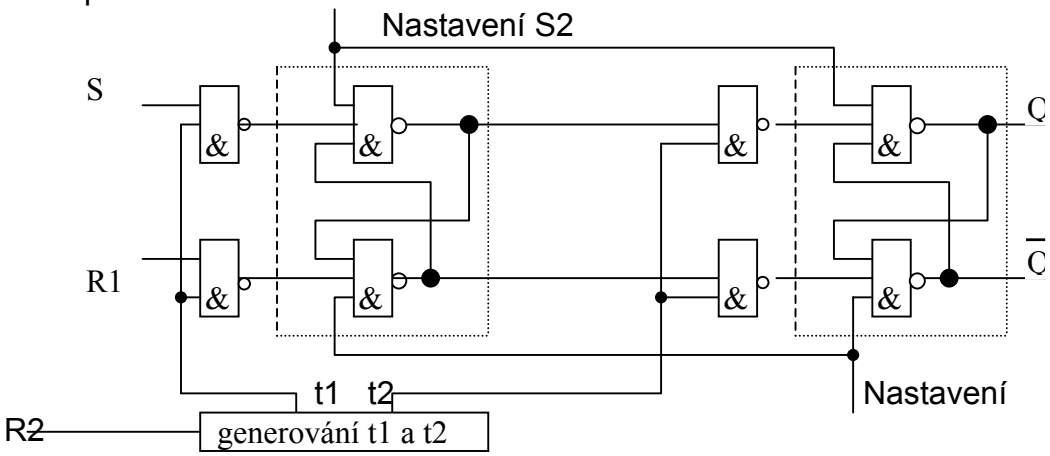
D klopný obvod také vznikne z obvodu typu J-K, vložíme-li invertor mezi vstupy J a K tak, že K je komplementem J (následující obr.). Z pravdivostní tabulky obvodu J - K plyne, že $Q_{n+1} = 1$ pro $D_n = J_n = {}_n = 1$ a $Q_{n+1} = 0$ pro $D_n = J_n = {}_n = 0$. Tedy $Q_{n+1} = D_n$. Vzhledem k tomu, že pravdivostní tabulka obvodu J-K pro J = se neliší od tabulky obvodu R-S pro R = , můžeme obvod tohoto typu rovněž sestavit z řízeného obvodu R-S (v tom případě hovoříme o jednoduchém nebo jednočinném klopném obvodu). Takový obvod mění svůj stav při náběžné hraně hodinového impulsu; v případě, že D obvod sestavíme z obvodu J-K typu master - slave, mění se stav s týlovou hranou hodinového impulsu. Klopné obvody typu D mohou sloužit jako paměti binární informace, která se vybaví hodinovým impulsem k dalšímu zpracování. Příkladem jednoduchého obvodu typu D je integrovaný obvod 7474 (dva jednoduché D-obvody v jednom pouzdru).



5.5 Dvojčinný klopný obvod J-K typu master-slave

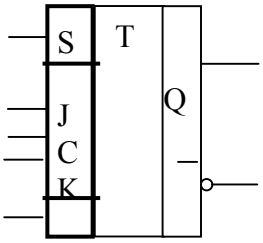
Pro odstranění neurčitého stavu klopného obvodu R-S byl vyvinut tzv. dvojčinný klopný obvod J-K. Principiální zapojení je na následujícím obrázku.

Principiální schéma:



Hodinový signál

Obr. J-K KO M-S



Zapojení obsahuje dva řízené klopné obvody R-S, u nichž výstupy Q a prvního jsou navázány na vstupy S a R (po řadě) druhého. Druhý klopný obvod se řídí invertovanými hodinovými impulsy a zpětná vazba je vedena z výstupu druhého klopného obvodu na vstup prvního. První klopný obvod se nazývá řídicí (master), druhý klopný obvod je řízený (slave). S náběžnou hranou hodinového impulsu se nastavuje úroveň na výstupech řídícího obvodu ; řízený obvod je uzavřen, neboť úroveň na jeho hodinovém vstupu = 0. S úběžnou hranou hodinového impulsu se uzavírá vstup řídícího klopného obvodu a stav na jeho výstupu je kopírován řízeným klopným obvodem. Jeho výstupní úrovně jsou vedeny zpětnou vazbou na vstup řídícího obvodu, tam však nezpůsobí žádnou změnu, neboť tentokrát je řídící obvod uzavřen (C = 0). Asynchronní vstupy jsou zavedeny do řídicího klopného obvodu. Nastavíme-li asynchronními vstupy řídící klopný obvod, přesune se tato informace do řízeného klopného obvodu okamžitě (je-li C = 0), neboť mezi hodinovými impulsy je C = 0, tedy = 1; řízený klopný obvod, ovládaný signálem , je tedy otevřen.

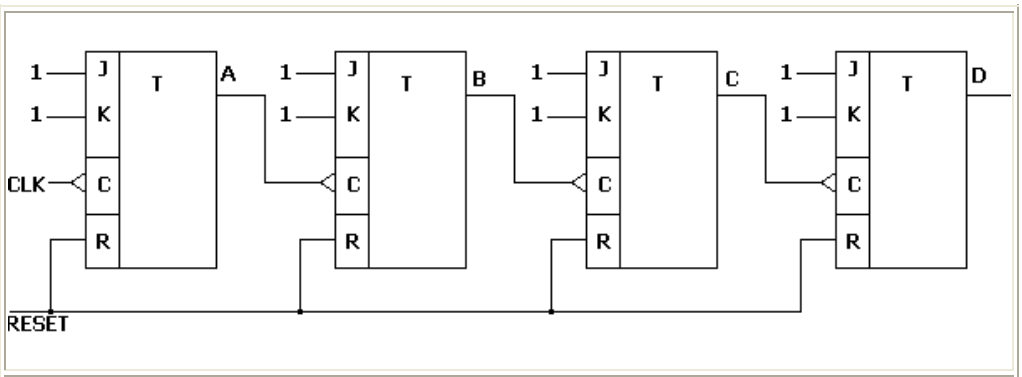
5.6 Funkce posouvání a čítání

5.6.1 Čítače kmitočtu

Kombinaci klopných obvodů schopnou čítat počet vstupních impulsů a vyjádřit jejich počet pomocí buď binárního nebo jiného kódu, nazýváme čítače.

5.6.1.1 Asynchronní čítač vpřed

Asynchronní čítač vpřed je znázorněn na následujícím obrázku:

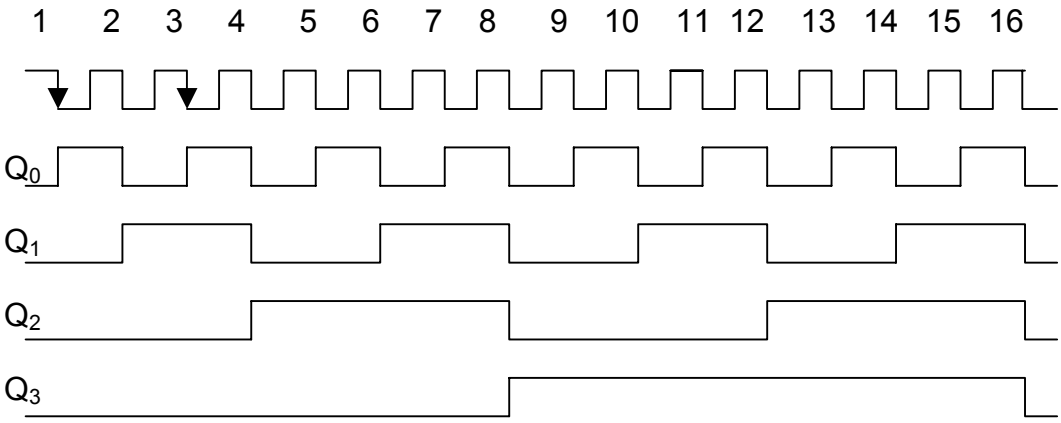


Realizace čítače řízeného sestupnou hranou impulsu

Sestává se z řetězce (v našem případě čtyř) klopných obvodů T. Klopné obvody byly vytvořeny pomocí obvodu J-K připojením obou vstupů na logickou 1. Jednotlivé klopné obvody mění stav výstupu při každé úběžné hraně na svém hodinovém vstupu. Překlápění obvodů se tedy řídí v podstatě dvěma pravidly:

1. Výstup Q_0 obvodu T_1 mění svůj stav při každé úběžné hraně vstupních impulsů,

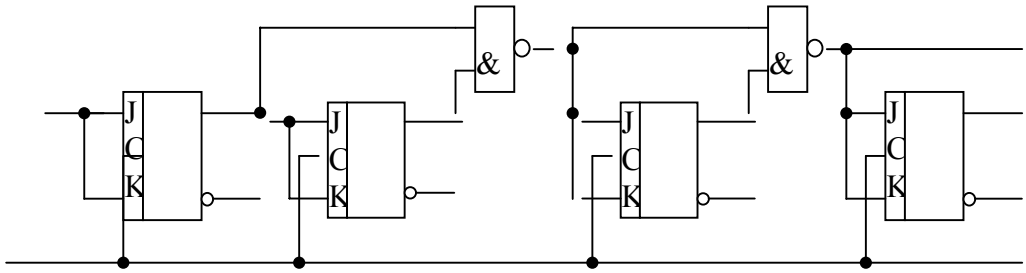
2. Všechny ostatní výstupy mění svůj stav právě když předcházející klopný obvod mění stav výstupu Q z 1 do 0.
Aplikací těchto pravidel dostáváme tvar signálu na výstupech Q_0 - Q_3 tak, jak je uvádí následující obrázek. Vidíme, že stav výstupů Q_0 - Q_3 je přesně binární reprezentace čísla, udávajícího pořadí vstupního hodinového impulsu.
Takovýto řetězec klopných obvodů čítá tedy v binární soustavě. Pro názorné zobrazení příslušného čísla je však třeba užít dekodéru, tj. logické sítě s 16 výstupy tak, aby při každé kombinaci jednotlivých bitů byl na logické úrovni 1 právě jeden z výstupů.



číslo výstupního	*výstupy klopných obvodů*			
impulsu	Q_3	Q_2	Q_1	Q_0
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
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0

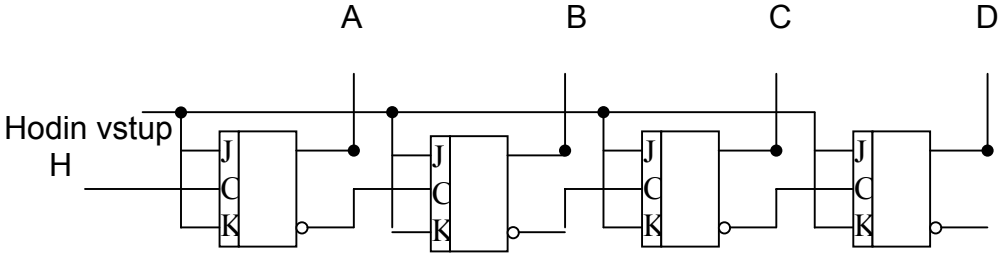
5.6.1.2 Synchronní čítače

Z rozboru funkce asynchronních čítačů je zřejmé, že změna stavu z 1 do 0 předcházejícího obvodu teprve působí změnu stavu následujícího obvodu. V případě, že všechny obvody jsou na logické úrovni 1, vybuzují postupně jeden druhý a doba potřebná k tomu, aby celý čítač vykonal odezvu na vstupní impuls, může být srovnatelná s dobou mezi jednotlivými impulsy. To je nebezpečné zvláště tehdy, je-li řada obvodů dlouhá, neboť dokud celý řetězec nedosáhne ustáleného stavu, nelze jeho výstupy synchronně (v jednom okamžiku) odečíst a zařízení tak ztrácí smysl.
Aby se doba odezvy čítače na vstupní impuls snížila, byla opět poněkud modifikována topologie zapojení čítače tak, aby na klopné obvody byl přiveden vstupní impuls synchronně. V tom případě je však třeba zabezpečit, aby měnily stav jen ty klopné obvody, u kterých je to požadováno. Zde se plně využívá funkce obvodu typu T. Pomocí řídící logiky se ze stavu výstupů předcházejících obvodů určuje logická úroveň vstupu T a tedy skutečnost, zda klopný obvod změní nebo nezmění stav při aplikaci následného hodinového impulsu. Čítač se tak v době mezi impulsy "připravuje" na zpracování následného hodinového impulsu. Příklad binárního synchronního čítače je na následujícím obr. Použitím tohoto zapojení lze zhruba zdvojnásobit pracovní frekvenci čítače ve srovnání s asynchronním. Využitím asynchronních vstupů klopných obvodů lze před započetím čítání nastavit počáteční stav čítače, tj. provést předvolbu.



5.6.1.3 Vratný čítač

Zařízení, které jsme právě rozebrali zobrazují počet vstupních impulsů v binárním tvaru, tj. každý další impuls způsobí zvýšení stavu čítače o 1. Často je třeba, aby čítač počet impulsů odečítal. Čítač, který toto provádí, se nazývá čítačem vzad. Asynchronní čítač vzad realizujeme tak, že místo výstupu předchozího klopného obvodu (Q_n) připojíme na hodinový vstup následujícího obvodu (Q_{n+1}) předchozí negovaný výstup ().
Čítače, které umožňují podle řídícího povelu čítání buď vpřed nebo vzad nazýváme vratnými. Asynchronní čítač vzad realizujeme tak, že místo výstupů $_i$ propojíme s hodinovými vstupy následujících obvodů výstupy. Chceme-li tedy realizovat vratný asynchronní čítač, musíme sestrojit přepínač, který bude přepínat do hodinových vstupů buď výstup nebo předchozího klopného obvodu. Příklad realizace takového přepínače je na následujícím obr.

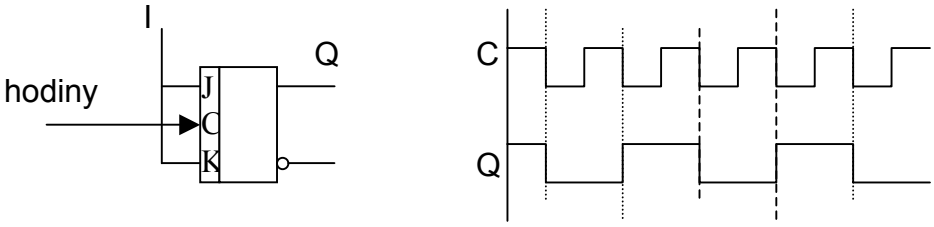


U tohoto obvodu se v prvním kroku KO nastaví do 1.

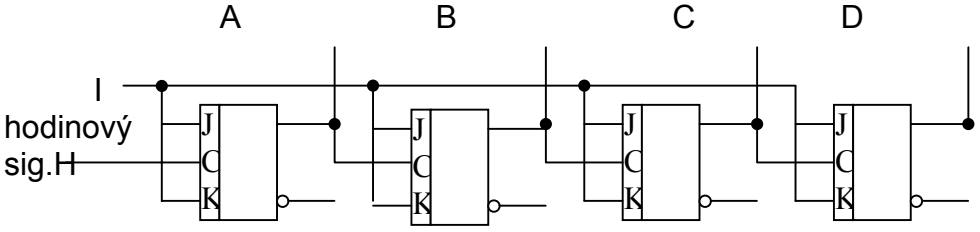
Čítač čítá	
P	
1	vpřed
0	vzad

Obdobně jako se konstruují synchronní čítače vpřed lze vytvořit i vratné synchronní čítače se zachováním jejich výhody oproti asynchronním - podstatně kratší doby odezvy na vstupní impuls. Jejich schéma je podstatně složitější, než u asynchronních vratných čítačů, neboť je nutné hradlovat hodinový signál v každém stupni; proto si je nebudeme uvádět. Vratné čítače se vyrábějí buď jako samostatné integrované obvody střední integrace, nebo jsou na jednom čipu integrovány s obvody plnícími další funkce (například programovatelný čítač/časovač 8253-4, využívaný v osobních počítačích ke generaci časových signálů, obsahuje tři 16bitové vratné čítače, obvody-registry umožňující nastavit a zapamatovat si jejich funkci, a některé další obvody).

5.6.1.4 Binární čítač v před (nahoru) - dělič



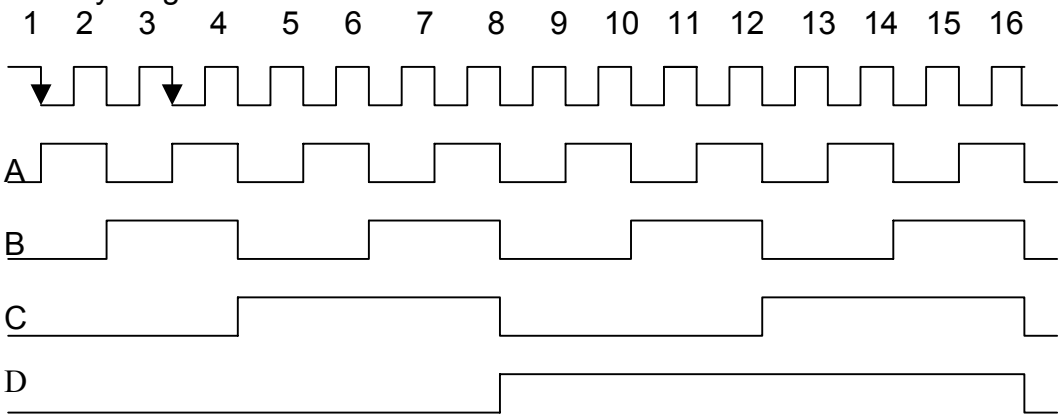
U KO s přivedením hodinového signálu KO překlopí, když hodinový signál přechází z 1 do 0. Z časového diagramu je názorné, že v poměru hodinového signálu
a signálu na výstupu Q je 2:1. Přechod z 1 do nuly nejdříve přivede KO do 1 a až po druhém signálu při přechodu z 1 do 0 dojde k překlopení výstupu do 0 odtud poměr 2:1
Této vlastnosti využijeme v kaskádním zapojení č KO.



Na vstupy JK přivedeme logickou úroveň 1. Přivedeme hodinový signál na první vstup. Výstup prvního KO se připojí na hodinový vstup druhého KO atd. až zapojíme takto čtyři KO.

Tím docílíme poměr mezi hodinovými signály a jednotlivými výstupy v poměru:
H/A 2:1, H/B 4:1, H/C 8:1, H/D 16:1

Časový diagram:



Pravdivostní tabulka vzestupného čtyřbitového binárního čítače je naznačena na dalším obrázku

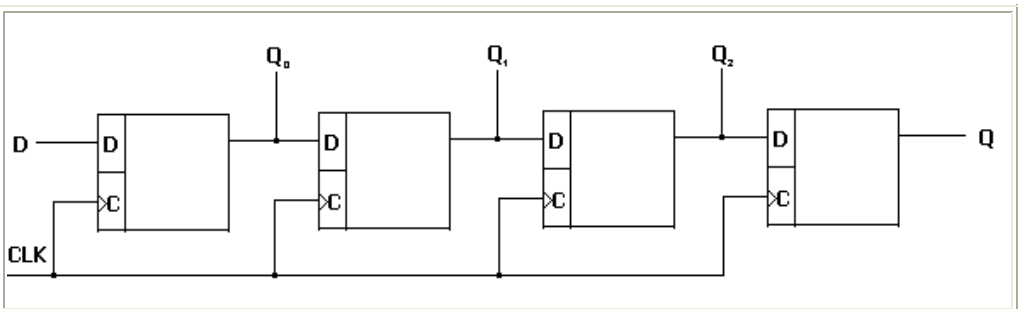
H		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Výstupy	A	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	B	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	C	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0
	D	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0

Po šestnáctém taktovacím pulsu se všechny KO vynulují.

Často tohoto zapojení se používá jako děliče kmitočtu:
Přivedeme-li na vstup frekvenci **f** dostaneme na výstupu binárního čítače frekvenci **f/2**, **f/4**,**f/8** atak dále.

5.6.2 Posuvné registry

Kombinací n klopných obvodů, schopnou zapamatovat si n-bitovou informaci, nazýváme registrem. Spojíme-li výstup klopného obvodu se vstupem následujícího klopného obvodu etc., dostáváme sestavu tzv. posuvného registru.



Sériový registr (posuvný registr)

Jedním taktem signálu CLK se informace posune o jeden D-KO

5.6.2.1 Sériový vstup dat

Předpokládejme,že máme realizovaný posuvný registr z 5 klopných obvodů a že chceme zapsat do registru binární číslo 01011. Číslo zapisujeme způsobem obvyklým u decimálních čísel tak, že nejméně významný bit je vpravo. Sledujme při tom následující tabulku. Nejprve je třeba vynulovat všechny klopné obvody aplikací logické úrovně 0 na mazací vstup registru.
Pak všechny výstupy Q_0 - Q_4 jsou 0.

Číslo hod. impulsu	Bit	Q_4	Q_3	Q_2	Q_1	Q_0
1	0 →	1	1	0	0	0
2	0 →	1	1	1	0	0
3	0 →	0	0	1	1	0
4	1 →	1	0	1	1	0
5	0 →	0	1	0	1	0

Nastavovací vstup registru je na úrovni logické 0, což má za následek, že všech pět nastavovacích vstupů jednotlivých klopných obvodů je na úrovni logické jedničky. Data na sériovém vstupu registru musí být synchronizována s hodinovými impulsy tak, že při aplikaci hodinového impulsu do registru je na sériovém vstupu nastaven jeden z datových bitů počínaje od nejméně významného (LSB). Při prvním hodinovém impulsu je tedy na vstupu registru (KO_4) nejméně významný bit, který se úběžnou hranou impulsu přesune na výstup Q_4, tedy na vstup S_3 a jeho komplement na R_3. Proto při aplikaci druhého hodinového impulsu se nejméně významný bit přesune na výstup Q_3 event. komplementovaný na , zatímco do $K0_4$ se načte další bit digitálního slova. Po pěti bitech se musí hodiny zastavit a podle tabulky vidíme, že v každém z klopných obvodů je po řadě informace o jednom z bitů zadaného dvojkového čísla. Tuto informaci můžete přečíst najednou z výstupů Q_4 - Q_0; v tom případě mluvíme o převodníku sériového na paralelní kód, nebo opět sériově z výstupu Q_0; například při značně větší rychlosti hodinových impulsů. To je tzv. metoda vyrovnávací paměti (registr FIFO, first in, first out). Např. při spojení terminálu s počítačem není možné, aby terminál byl k počítači připojen po celou dobu, kdy vkládáme klávesnicí data. Proto se data z klávesnice ukládají do vyrovnávací paměti (např. jeden řádek obsahující max. 80 znaků), která se rychle vyprázdní na povel k odeslání dat.

5.6.2.2 Paralelní vstup dat

Všimněme si nyní nastavovacích vstupů PS_0 - PS_4. Tyto vstupy jsou od vlastních nastavovacích vstupů klopných obvodů odděleny hradly NAND, které umožňují blokovat vnější vstupy a zapsat informaci jediným impulsem do vstupu nastavení registru. To je vhodné např. při přenosu dat od digitálních měřících přístrojů,kde je informace v paralelní formě, tj. všechny bity jsou dostupné najednou. K propojení měřícího přístroje s řídícím počítačem však někdy není možné použít velkého množství vodičů, proto je třeba převést tuto paralelní informaci na sériovou. Po ukončení měření, kdy je na vstupech PS_0 - PS_4 informace např. o jednom z decimálních čísel zobrazovací jednotky,

aplikujeme logickou úroveň 1 na vstup nastavení registru, čímž nastavíme výstupy jednotlivých klopných obvodů Q_0 - Q_4 do odpovídajících pozic (registr je třeba nejprve vynulovat). Proto se vstupu nastavení registru někdy říká zapisovací vstup (write). Informace shromážděná v registru se nyní může přečíst buď opět paralelně; v tom případě je třeba opatřit výstupy Q_0 - Q_4 obdobnými hradly NAND jako nastavovací vstupy. Druhý vstup hradel pak umožňuje oddělit výstupy registru od dalšího zařízení. Informaci však můžeme přečíst též sériově z výstupu Q_0 při synchronní aplikaci hodinových impulsů. V tomto případě hovoříme o převodníku paralelního na sériový kód.

5.6.3 Kruhový registr

Spojíme-li výstup Q_0 se sériovým vstupem registru, opakuje se na výstupech Q_0 - Q_4 táž informace vždy po pěti hodinových impulsech. Takovému uspořádání říkáme kruhový registr. Lze si např. představit desetibitový kruhový registr,do jehož MSB byla na počátku uložena jednička a jinak byl registr vynulován. Je zřejmé, že vždy po deseti hodinových impulsech bude na MSB opět jednička a podle toho, na kterém bitu je právě jednička poznáme, kolik impulsů (v rozmezí od 0 do 9) přijal hodinový vstup registru. Takový registr tedy tvoří dekadický čítač (dělič deseti), kde není nutný dekodér binární informace na dekadickou; komplikaci však tvoří skutečnost, že je třeba vždy po zapnutí přístroje vložit do registru jedničku, která pak v registru "obíhá". Pro dělení větším číslem potřebujeme rovněž značný počet klopných obvodů, takže cena zařízení neúměrně stoupá.

Kontrolní otázky:

9. Popište odlišnosti jednotlivých typů klopných obvodů.
10. Proč se využívají v realizaci prvků počítačů klopné obvody typu Master - Slave?
11. Proč musí být posuvný registr realizován dvojčinným klopným obvodem?

Úkoly k zamyšlení:

4. Zamyslete se nad obecnou architekturou počítače a určete, které sekvenční logické obvody se využívají v jednotlivých částech počítače?

Korespondenční úkol:

4. Navrhněte posuvný registr z klopných obvodů typu D a nakreslete schema zapojení.

Shrnutí obsahu kapitoly

V této kapitole jste se seznámili s realizací základních klopných obvodů, jako stavebních prvků složitějších sekvenčních obvodů. Důraz v této kapitole byl kladen na pochopení konstrukce čítačů a registrů. Velká pozornost byla věnována vysvětlení principů výstavby seriových a paralelních obvodů vstupu dat do posuvných registrů.

6 Prvky mikropočítače

6.1 Základní architektura počítače

Z koncepčního hlediska je mikropočítač takové uspořádání logických obvodů umožňující provádění logických i aritmetických operací podle posloupnosti povelů (programu) nad programem určenými vstupními veličinami za účelem získání výstupních hodnot na programem určených výstupech. Architektura většiny dnešních mikropočítačů se zakládá na koncepci stanovené ve 40. letech Johanem von Neumannem, kdy program i data jsou uloženy v jedné operační paměti.

Architektura mikropočítačů se sice v detailech od sebe značně odlišuje, můžeme však u každého vysledovat následující bloky (subsystémy):

Operační paměť
uchovává vstupní a výstupní data a program
(posloupnost povelů - instrukcí) ve formě binárních čísel

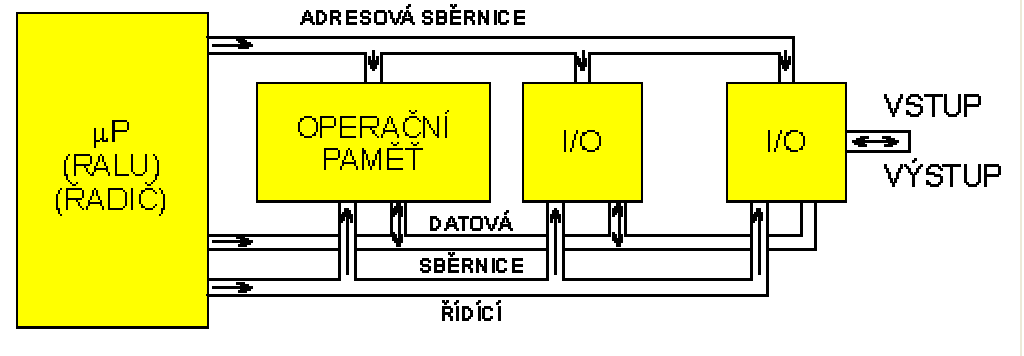
Řadič
dekóduje postupně instrukce programu uložené v operační paměti a generuje signály zajišťující činnost ostatních bloků mikropočítače

Aritmeticko-logická
jednotka provádí aritmetické a logické operace podle (ALU) signálů řadiče nad řadičem určenými veličinami. Někdy je doplněna registry pro uložení mezivýsledků operace (RALU)

Vstupní a výstupní jednotka
zabezpečuje komunikaci mikropočítače vnějším (I/0) okolím pomocí přídavných zařízen umožňujících komunikaci s mikropočítačem, např.klávesnice, tiskárna apod.

Blok registr
- aritmeticko-logická jednotka (RALU) spolu s řadičem tvoří tzv. procesor systému. Je-li obsažen v jednom nebo několika integrovaných obvodech velké integrace označujeme jej jako mikroprocesor (m P).

Propojení bloků je provedeno pomocí systému sběrnic. Principielní propojení sběrnicemi je uvedeno na následujícím obrázku.



Sběrnice mikropočítače

Adresová sběrnice je jednosměrná, slouží k adresování paměťového místa v operační paměti nebo vstupních a výstupních jednotek. Další datová sběrnice je obvykle obousměrná a slouží k přenosu dat mezi bloky. Třetí sběrnice je řídící, která slouží k přenosu povelů (signálů) z řadiče zajišťujících programem stanovenou činnost jednotlivých bloků. V následujících odstavcích probereme podrobněji principielní zapojení a činnost jednotlivých bloků.

6.2 Operační paměť

V současné době se převážně u mikropočítačových systémů používají polovodičové paměti. Polovodičové paměti můžeme zhruba rozdělit do dvou skupin:

a) Paměti, kde do libovolného místa určeného adresou můžeme buď zapsat data v binární formě nebo v paměti uložená data přečíst. Tyto paměti označujeme obvykle zkratkou RAM, což je zkratka anglického názvu "Random Access Memory".

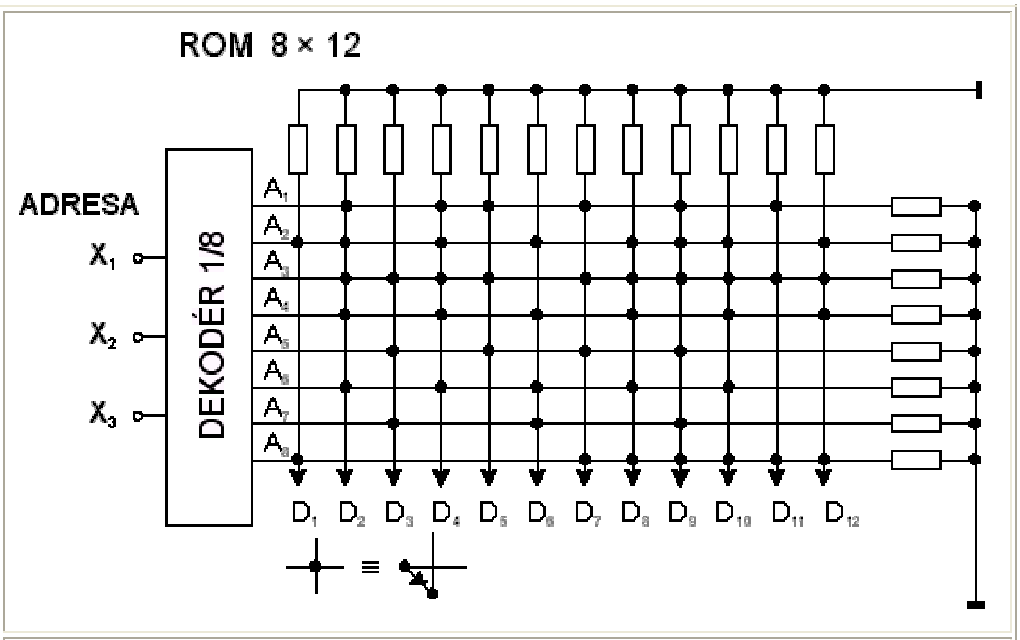
b) Paměť s neměnným zápisem dat neboli paměti konstant. Z těchto pamětí během činnosti mikropočítače je možno pouze číst binární data. Označujeme je obvykle zkratkou ROM, což je zkratka anglického názvu "Read Only Memory".
Z hlediska závislosti uchování informace na napájecím napětí se u paměti typu RAM informace odpojením napájecího napětí ztrácí, zatím co u pamětí typu ROM zůstane zachována.

6.2.1 Paměti typu ROM

Paměti typu ROM jsou paměti konstant u kterých jsou data trvale uložena. Data jsou v nich uložena buď přímo výrobcem nebo u paměti typu PROM (Programable Read Only Memory) je možné například pomocí tavných spojek podle požadavků uživatele do jednotlivých míst paměti data v binární formě jednou pro vždy zapsat. V současné době existují paměti označované zkráceně EPROM, u kterých je možno speciálním technologickým postupem zapsaná data vymazat a zapsat nová. Rychle se též začínají uplatňovat tzv. EEPROM (E^2PROM, Flash), tj. paměti, do nichž lze zapsat i vymazat data elektrickou cestou a přesto zůstanou zachována po vypnutí napájení. Známé aplikace jsou

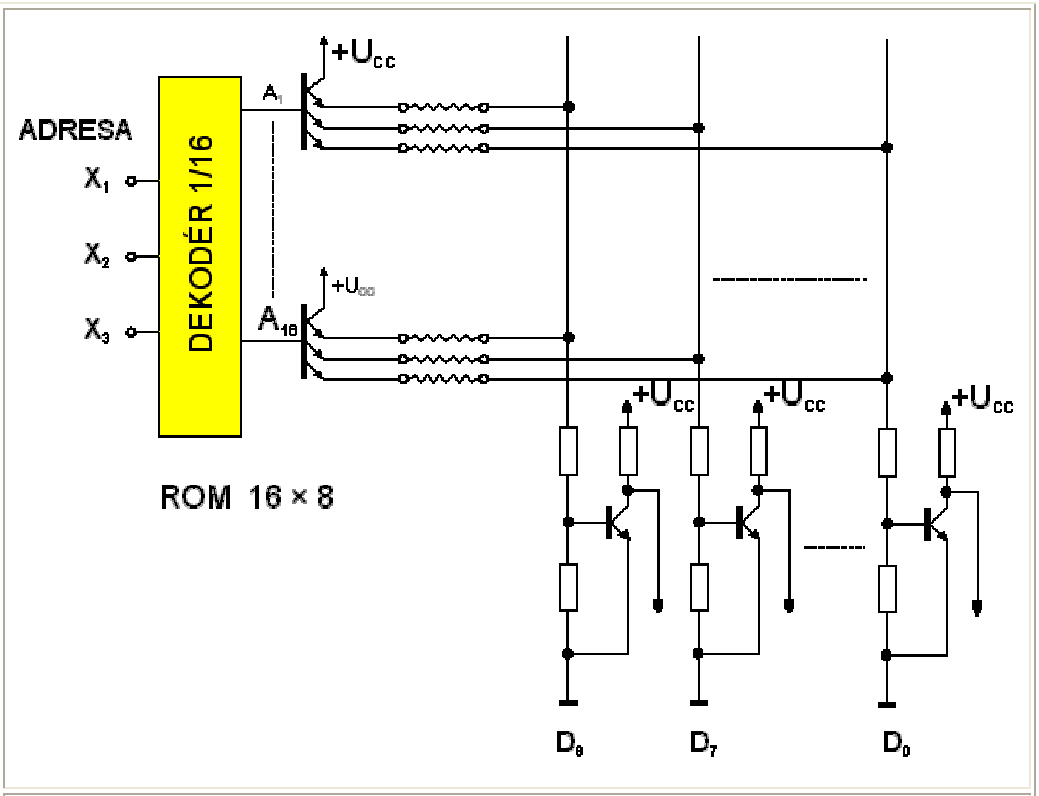
například Flash BIOS u základních desek osobních počítačů, rozšiřující karty do notebooků nahrazující diskovou paměť apod.

Paměti typu ROM jsou obvykle uspořádány v maticovém tvaru s organizací obvykle označovanou jako počet slov (bytů) x počet bitů ve slově. Principielní zapojení je na následujícím obrázku.



Principielní zapojení paměti ROM

Vstupní slovo dekodéru typu 1/n tvoří adresu, pomocí které se určuje aktivní adresový vodič na kterém je při čtení napěťová úroveň log 1 .Pomocí diod v místě křižování adresových vodičů s bitovými vodiči se úroveň log 1 přenese na bitové vodiče. Kde není v místě křižování vodičů dioda, je na příslušném bitovém vodiči napětí log 0. Na příklad při aktivaci adresového vodiče A je na bitových vodičích paměti uvedené na obrázku binární slovo (010110101010). Jako příklad programovatelné paměti PROM uvedeme paměť s víceemitorovými bipolárními tranzistory.

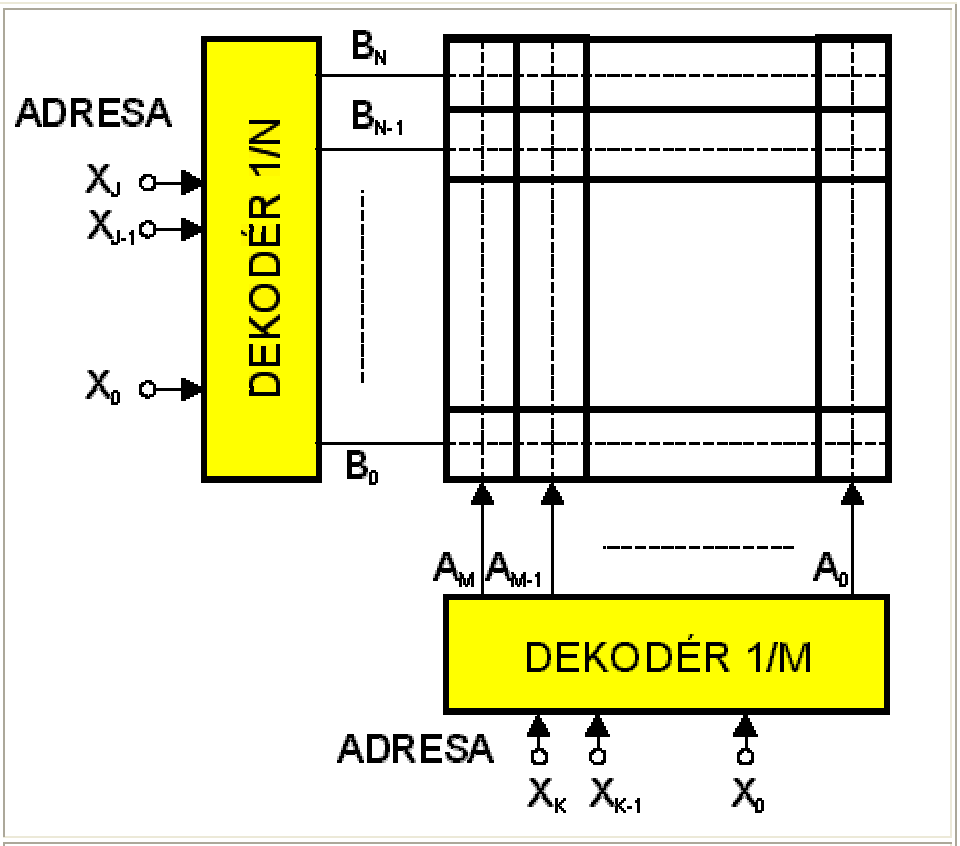


Příklad paměti ROM s víceemitorovými bipolárními tranzistory

Výrobce dodává uživateli paměť, kde výstupní slovo je vytvořeno ze samých logických jedniček. Přetavením tavné spojky pomocí proudového impulsu může uživatel dosáhnout na požadovaném bitu logickou nulu. Na výstupech paměti jsou zesilovače umožňující získat potřebný logický zisk.

6.2.2 Paměti typu RAM

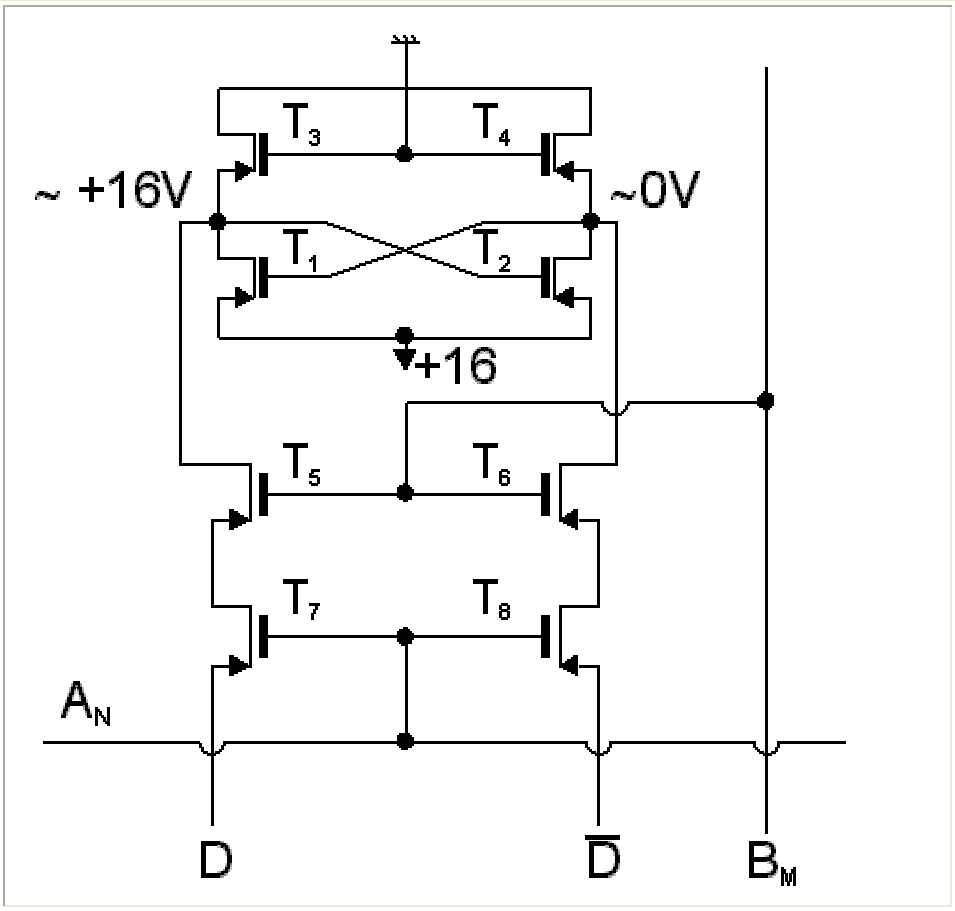
Paměti typu RAM jsou paměti, kdy do libovolného místa můžeme buď data v binární formě zapisovat nebo uložená data přečíst. Do paměti RAM můžeme zapisovat program,vstupní data, mezivýsledky apod. Příklad blokového schématu paměti typu RAM je na následujícím obrázku.



Blokové schéma paměti typu RAM

Jednotlivé paměťové buňky jsou obvykle uspořádány maticově, např. v organizaci N x M. Buňku, do které chceme provést zápis nebo čtení jejího obsahu, aktivujeme pomocí řádkových a sloupcových adresových vodičů buzených dekodéry 1/N a 1/M. Vstupní slova dekodérů pak tvoří adresu buňky. Vstupy a výstupy paměťových buněk jsou vedeny ke čtecím a zapisovacím zesilovačům. Pro zápis nebo čtení celého slova najednou se musí použít pro každý bit samostatné čtecí a zapisovací zesilovače. Čtení a zápis je řízen logickým obvodem ovládaným povely z řadiče.

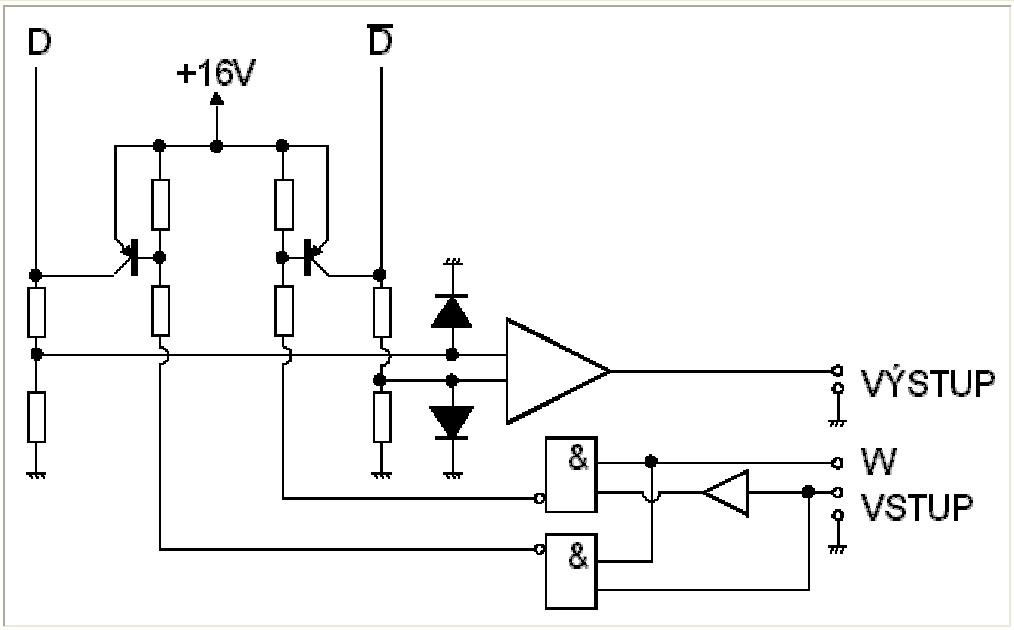
Jako paměťová buňka by mohl být použit klopný obvod (sekvenční) typu D .Z důvodů dosažení velké integrace obvodů na jednom polovodičovém čipu se zapojení v buňce zjednodušuje, přičemž se používají jak tranzistory unipolární tak bipolární. Pro paměťové buňky byla vyvinuta celá řada zapojení a pro detailní studium jejich zapojení odkazujeme čtenáře na doplňkovou literaturu. Zde si pouze uvedeme jako příklad zapojení 8tranzistorové paměťové buňky s tranzistory MOS s kanálem P uvedené na následujícím obrázku.



Příklad zapojení 8tranzistorové paměťové buňky s tranzistory MOS

Paměťovou buňku tvoří bistabilní klopný obvod tvořený čtyřmi tranzistory. Napěťovou úroveň log 1 pro zapojení na obrázku předpokládejme (~ +16 V) a pro log 0 (~ 0 V). Předpokládejme, že tranzistor T_1 je otevřen a tranzistor T_2 zavřen. Pak na kolektoru tranzistoru T_1 je napěťová úroveň log 1 (~ +16 V) a na kolektoru tranzistoru T_2 je úroveň log 0 (~ 0 V). Tyto napěťové úrovně se přenesou na hradla (řídící elektrody) obou tranzistorů a stav zůstane zachován pokud na kolektory tranzistorů T_1 a T_2 se nepřevede zvenčí opačné napětí. Buňka je přes adresovací tranzistory T_5 až T_8 připojena na datovou sběrnici (D a $\overline{D}$).
Adresovací tranzistory aktivujeme ustavením řádkového A_N a sloupcového B_M na napěťovou úroveň log 0 (~ 0 V). V případě, že adresovací tranzistory jsou otevřeny a na sběrnici D přivedeme úroveň log 0 a na $\overline{D}$ úroveň log 1, klopný obvod se překlopí.

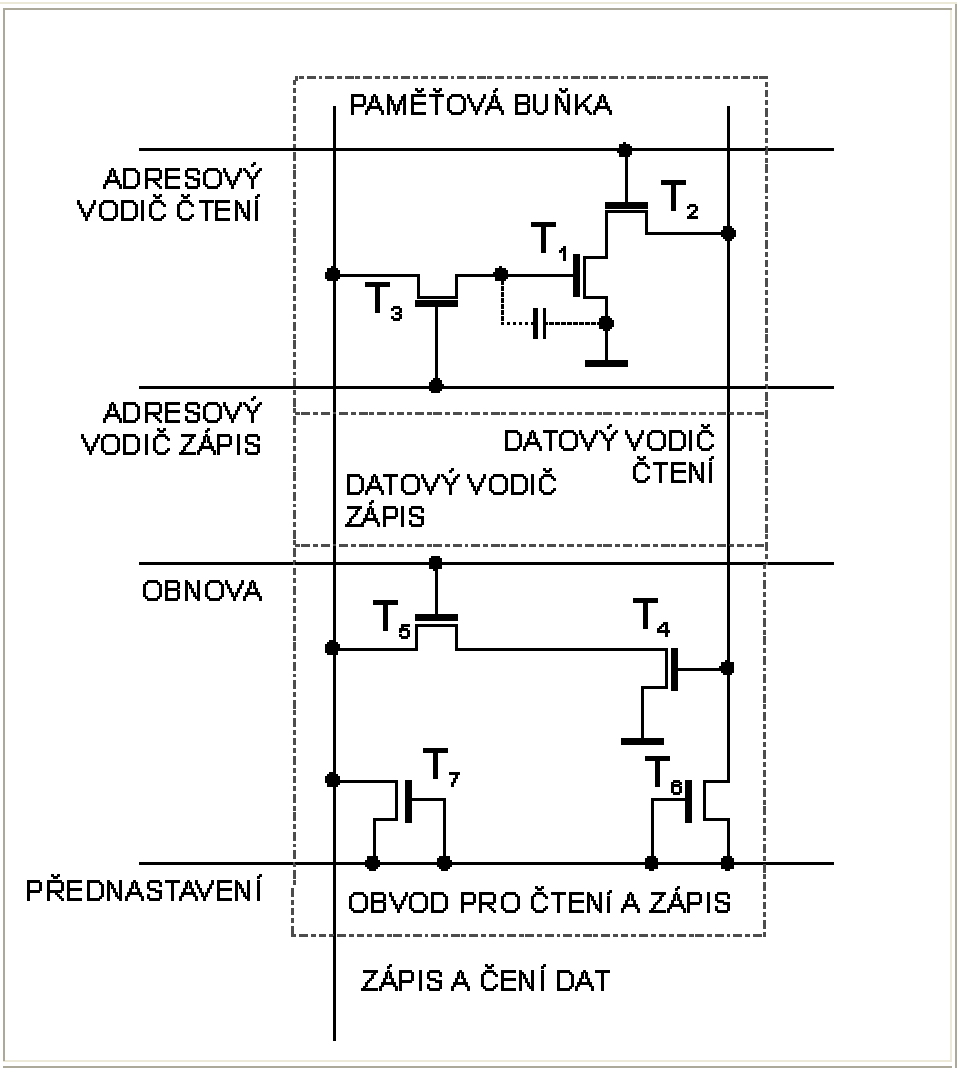
Principielní zapojení čtecího a zapisovacího obvodu pro uvedenou paměťovou buňku je uveden na následujícím obrázku.



Čtecí a zapisovací obvod paměťové buňky

Čtení se provádí pomocí diferenciálního zesilovače a zápis pomocí dvojice zesilovačů s bipolárními tranzistory, jejichž vstupy jsou ovládány povelem pro zápis (W). Je-li W na úrovni log 1 je vstup otevřen pro zápis dat. V opačném případě je vstup blokován a data uložená v buňce se mohou pomocí diferenciálního zesilovače pouze číst.

Snaha o zvýšení stupně integrace vedla vývojáře ke snaze o snížení počtu tranzistorů nutných k vytvoření jedné buňky a k její adresaci. Podstatného snížení počtu tranzistorů na buňku bylo dosaženo v tak zvaných dynamických pamětech, kde nositelem informace je parazitní kapacita mezi hradlem a emiterem unipolárního tranzistoru. Vlivem svodu se napětí na této kapacitě snižuje a je nutné je vždy po určité době (např. po 2 msec) obnovovat. Je však nutné mít na zřeteli, že rychlost dnešních mikropočítačů je taková, že mohou vykonat řádově desítky až stovky operací mezi dvěma obnovovacími cykly. Příklad paměťové buňky dynamické paměti je uveden na následujícím obrázku.



Příklad paměťové buňky dynamické paměti

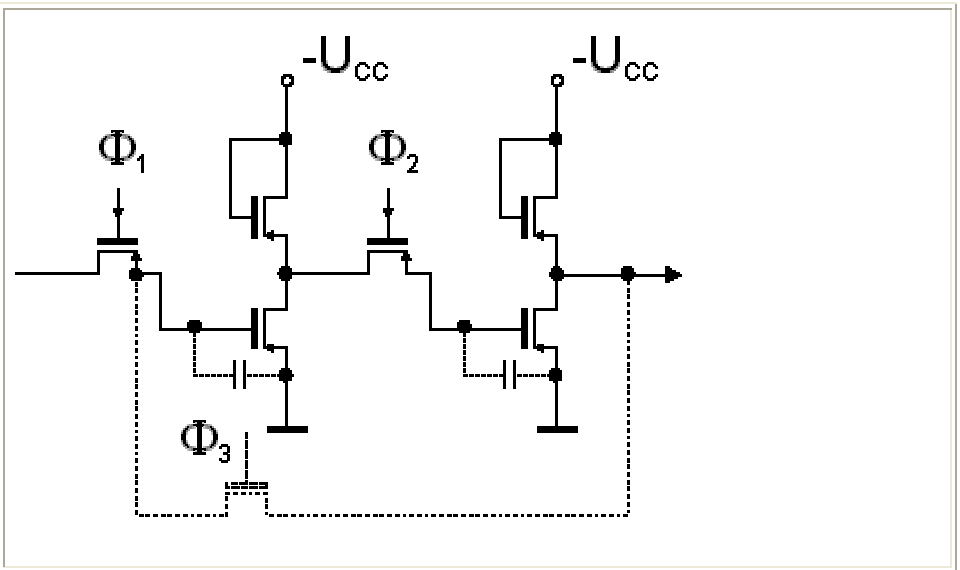
Zde je naznačen jeden sloupec paměti. Nositelem informace je kapacita mezi hradlem a emitorem tranzistoru T_1 .Tranzistory T_2 a T_3 slouží k připojení buňky ke čtecímu a zápisovému datovému vodiči. Postup obnovení informací v buňkách paměti si vysvětlíme pomocí čtecího a zapisovacího obvodu. Všechny datové vodiče paměťových buněk ve sloupci jsou propojeny paralelně a připojeny k jednomu obvodu.

Při čtení se nejdříve aktivují datové vodiče přes tranzistory T_6 a T_7 signálu log 1 na vodič přednastavení. Zároveň se aktivuje řádkový adresový vodič čtení, čímž se otevře tranzistor T_2 .Je-li v buňce zapsaná informace (bit) za úrovni log 1 je čtecí datový vodič přes tranzistory T_1 a T_2 uzemněn a je za něm úroveň log 0. V opačném případě, kdy informace je na úrovni log 0 je na čtecím datovém vodiči úroveň log 1. Informace na čtecím vodiči je tedy negována vzhledem k informaci zapsané v buňce.

Přivedeme-li na vodič obnovení log 1 otevře se tranzistor T_5 a informace se přepíše na zápisový datový vodič negovaně, tj. na zápisovém vodiči je informace stejná jako v buňce. Jestliže současně aktivujeme i řádkový zapisovací adresový vodič buňky a otevřeme tranzistor T_3, obnovíme informaci

zapsanou v buňce. V uvedeném zapojení se současně obnovuje informace zapsaná v buňkách v celé řádce. Je třeba podotknout, že čtení informace je nedestruktivní, tj. že se při čtení informace v buňce zachovává.

Vedle uvedených dynamických pamětí se používají paměti s postupným výběrem, které využívají dynamické posuvné registry. Nositelem informace u dynamického posuvného registru je opět parazitní kapacita mezi hradlem a emiterem unipolárního tranzistoru. Principielní schéma je na následujícím obrázku.



Principielní schéma

Posuv informace je řízen zpravidla dvoufázovými hodinovými impulsy Φ_1, Φ_2 a které se nepřekrývají. Tzv. statické posuvné registry mají v buňce ještě další tranzistor, který je řízen impulsy Φ_3. Tímto způsobem je zajištěna obnova informace i v případě, že neprobíhá posuv.

6.3 Jednočipové mikropočítače

Brzy po vyzkoušení mikroprocesorů ve výpočetních aplikacích se ukázalo, že se jedná o součástku mnohem universálnější, která se uplatní nejen ve výpočetních, ale i v řídicích aplikacích, kde významným způsobem zjednoduší a zlevní návrh řídicího hardware. Ukázalo se rovněž, že sestava řídicího systému s mikropočítačem může být optimalizována, tj. minimalizována z hlediska rychlosti, šířky slova zpracovatelného pomocí RALU, kapacity operační paměti, množství vstupně-výstupních obvodů i rozsahu řídicího programu. Postupně se tak vyvíjel minimalizovaný systém řídicího mikropočítače, který pak výrobci začali integrovat do jednoho čipu. Vznikl tak mikrokontrolér, mikrořadič, jednočipový mikropočítač (anglicky embedded microcontroller), který je svou architekturou přizpůsoben speciálně pro monitorování a řízení různých mechanismů a procesů. Kromě vlastního mikroprocesoru jsou na čipu integrovány ještě operační paměť (RAM), pevná paměť (ROM, PROM nebo EPROM), kde je uložen řídicí program, vstupní a výstupní obvody, případně přímo potřebná rozhraní pro řízení periferií,

například seriové rozhraní, časovač, A/D nebo D/A převodníky, výkonové budiče řídicích číslicových signálů apod. Jednočipové mikropočítače zpravidla obsahují oddělenou paměť programu (zpravidla užívaný název pro ROM, PROM nebo EPROM mikrořadiče) a paměť dat (zpravidla užívaný název pro RAM mikrořadiče) a časovače, které umožňují synchronizaci s vnějším okolím - s reálným světem. Proto se někdy jednočipovým mikropočítačům říká řídicí systémy v reálném čase. Aplikační rozsah těchto řídicích systémů je prakticky neomezený od běžných elektronických přístrojů používaných v domácnosti jako je videorekordér, automatická pračka, kuchyňský sporák, šicí stroj apod. až po nejnáročnější aplikace v automobilovém a leteckém průmyslu a ve vojenství. Vývoj a aplikace jednočipového mikropočítače je v číslicové technice jistou duální analogií k vývoji a aplikaci operačního zesilovače v technice analogové; i zde je obecný princip (v tomto případě univerzálního zesilovače) realizován v řadě modifikací, vhodných pro dané specifické aplikace.

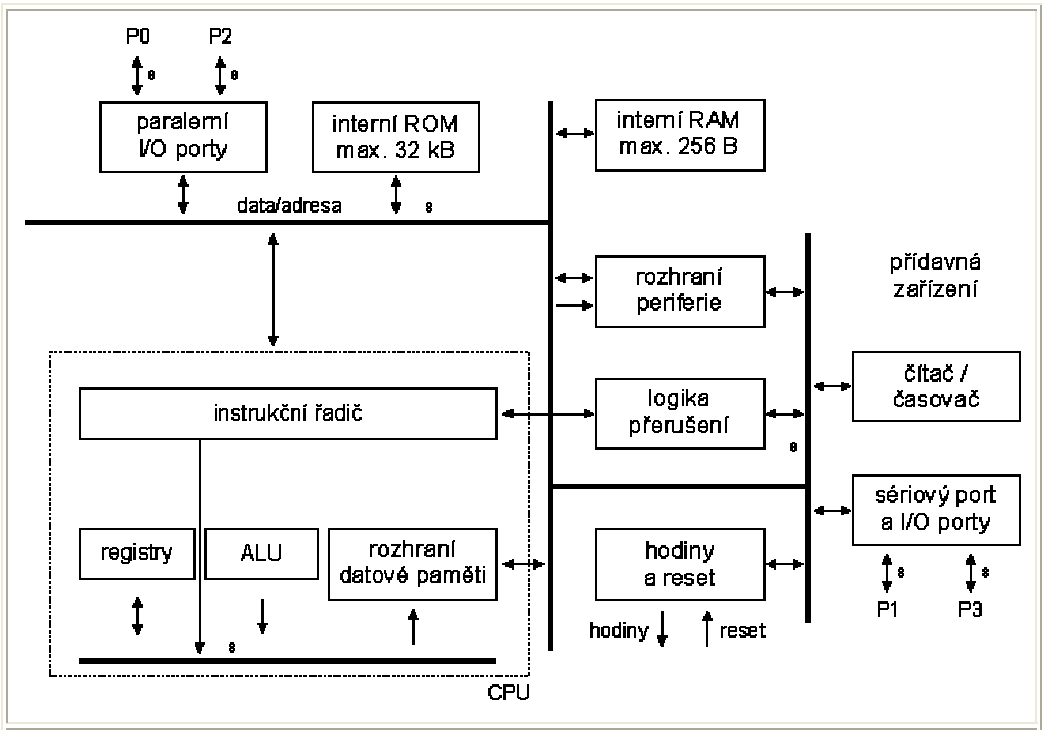
Největšího rozšíření z hlediska aplikací doznaly mikrokontroléry dvou významných výrobců mikroprocesorů - firem Motorola a Intel (podobně jako v technice stolních mikropočítačů). Spektrum výrobců jednočipových mikropočítačů je však velmi široké, mimo již jmenované firmy jsou to například americké Texas Instruments, National Semiconductors, evropské Philips, Siemens, japonské OKI, NEC, Toshiba, Hitachi apod. Charakteristické pro historický vývoj mikrokontroléru však je, že se zde neprojevuje snaha o drastické zvýšení rychlosti a kapacity paměti jako u osobního počítače. Je to dáno zejména faktem, že nasazením řídicího mikropočítače v dané aplikaci musí dojít k podstatnému zjednodušení výroby a tím ke snížení ceny výsledného výrobku. Stačí-li proto například k vykonání specifického řídicího programu operační paměť 64 bytů, není výrobce ničím motivován, aby použil mikrokontrolér s pamětí 128 bytů, protože by tím jen zdražil výsledný výrobek. Obecně platí, že se pro aplikaci použije ten nejjednodušší mikrokontrolér, který pro danou aplikaci vyhovuje. Tato motivace je proto také důvodem k tomu, že výrobci jednočipových mikropočítačů nabízejí velmi široké spektrum konfigurací mikrokontrolérů, lišící se i drobnými změnami v kapacitě pamětí (RAM i ROM), počtů vstupů/výstupů, použitelných periferií apod. Přito záměr výrobce aplikace je co nejjednodušší (a tím nejlevněji vyrobitelná a nejspolehlivější) sestava potřebného řídicího hardware, nejlépe složená pouze z vlastního mikrokontroléru. Historické řady mikrokontrolérů se proto spíše liší šířkou nabízených modifikací periferních obvodů, než vlastní sestavou řídicího mikropočítače. Z rozšiřování řídicího systému mikrokontroléru je patrná snaha o rozšíření kapacity pevné paměti PROM na čipu, z původních 1-4 kB u řady MCS-48 fy Intel na současných 32 kB u řady MCS-96 téže firmy, adresovatelného prostoru paměti, z původních 64 kB (řada Intel MCS-51) na 16 MB (řada Intel MCS-251), zrychlení provádění instrukcí (z původních max. 24 hodinových cyklů na 2 hodinové cykly u těchže řad) a o komfort programování (zdrojové programy pro řadu MCS-251 mohou být v jazyce C). Snaha je též o zaintegrování specializovaných obvodů pro řízení periferií, které pak minimálně zatěžují centrální jednotku mikrokontroléru. Mezi takové specializované obvody patří například zrychlení obsluhy přerušení, tzv. jednotka zpracování událostí apod. Jednotlivé generace jednočipových mikropočítačů se tak liší počtem časovačů, počtem a druhy vstupně výstupních

obvodů, počtem přerušovacích vstupů, přítomností A/D převodníku na čipu a jeho rozlišením, případně přítomností speciálních generátorů řídicích signálů (pulsní šířkově modulovaný výstup u řady MCS-96). K datu vydání skript byly autorům známy řady 8, 16 a 32bitových jednočipových mikropočítačů (poslední vyrábí například Motorola, Texas Instruments a Hitachi). Jednočipové mikropočítače jsou vyráběny, podobně jako mikroprocesory, buď technologií MOS (nebo jejími mutacemi HMOS, HMOS II apod.) nebo CMOS (nižší rychlost a spotřeba).

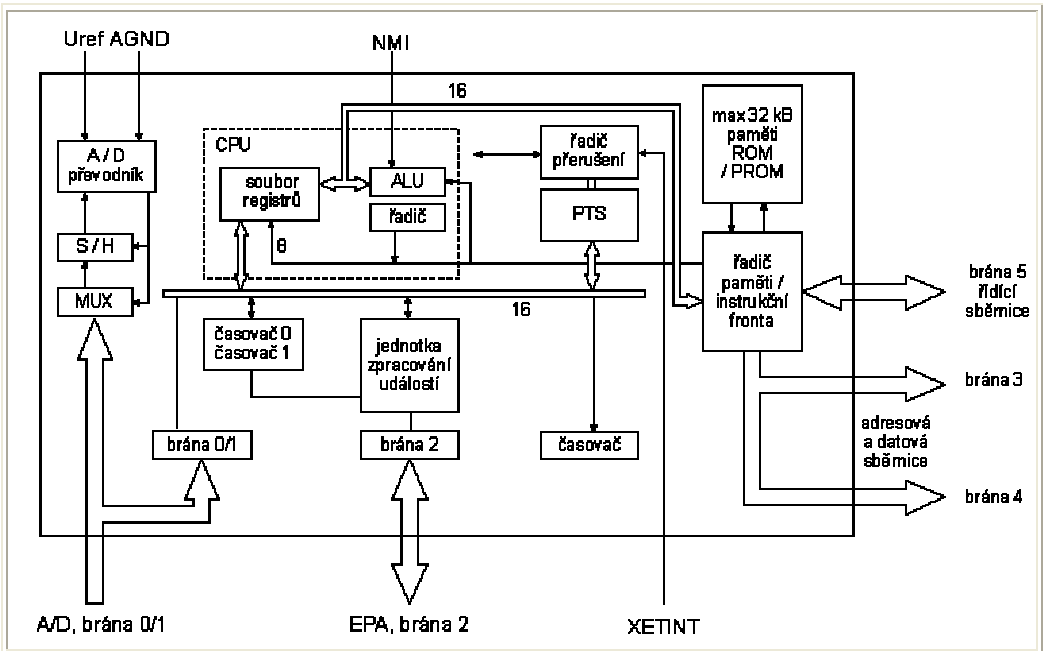
Pro ilustraci, jak se liší hardwarové sestavy jednotlivých historicky vzniklých řad jednočipových mikropočítačů uvedeme tři řady fy Intel: MCS-48 (vznikla v roce 1976), MCS-51 (vznikla v roce 1980) a MCS-96 (vznikla v roce 1983).

Řada Intel MCS-48	Řada Intel MCS-51	Řada Intel MCS-96
		(integrováno na čipu)
(integrováno na čipu)	(integrováno na čipu)	16bitová CPU
8bitová CPU	8bitová CPU	až 32kB ROM
1/2/4 kB ROM	4/8 kB ROM	až 744 bytů RAM
64/128/256 bytů RAM	128/256 bytů RAM	2 čítače/časovače a hlídací časovač (watchdog)
1 čítač/časovač	2 čítače/časovače	až 64 bitů číslicových vstupů/výstupů
paralelní vstup/výstup	paralelní vstup/výstup	až 5 seriových linek vstupu/výstupu
8bitový převodník A/D	seriový vstup/výstup	A/D převodník 8/10 bitů s multiplexerem
	až 6 zdrojů přerušení	jednotka zpracování událostí
		jednotka zrychlené obsluhy přerušení
		až 14 zdrojů přerušení

Srovnáme-li vnitřní struktury jednotlivých řad mikrokontrolérů najde v nich části, které jsou analogické vzájemně i ve srovnání s obecnou strukturou mikropočítače a části specifické pro příslušnou skupinu jednočipových mikropočítačů. Jako příklad uvádíme na následujících obrázcích zjednodušenou vnitřní strukturu 8bitového a 16bitového mikrořadiče.



Zjednodušená vnitřní strukturu 8bitového mikrořadiče



Zjednodušená vnitřní strukturu 16bitového mikrořadiče

I když jsou uvedená schemata základem určitých konkrétních jednočipových mikropočítačů, budeme je vykládat jako příklady možné architektury mikrořadičů.

Všimněme si nejprve 8bitového mikrořadiče. Základním prvkem jednočipového, tak jako i běžného, mikropočítače je centrální procesorová

jednotka CPU, tvořená řadičem instrukcí, ALU, registry pro zápis dat, instrukcí a stavu, a paměťovým rozhraním. Vnitřní adresová sběrnice je 8vodičová, multiplexovaná a slouží též jako (obousměrná) datová sběrnice. Řídicí sběrnice není v obrázcích zakreslena. Paměť programu (ROM) může mít kapacitu 64 kB právě tak jako paměť dat (RAM). To je kapacita, kterou je tento jednočipový mikropočítač schopen adresovat ať se jedná o paměť na čipu nebo externí paměť. Na čipu je však umístěno jen max. 256 bytů paměti dat a max. 32 kB paměti programu. Tento vzájemný poměr kapacity paměti programu a operační paměti je charakteristický pro řídicí softwarové aplikace, neboť ty neobsahují programové operace náročné na operační paměť. Mikrořadič má čtyři 8bitové obousměrné paralelní porty pro vstup/výstup číslicových logických signálů a jeden sériový port pro asynchronní a synchronní sériovou komunikaci a pro komunikaci v multiprocesorových systémech. Je schopen zpracovat logický signál přerušení z celkem 6 zdrojů. Může mít až tři programovatelné 16bitové čítače/časovače, použitelné buď odděleně pro komunikaci s vnějším prostředím, nebo lze jeden z nich využít pro generaci přenosové rychlosti sériového kanálu. Pracovní kmitočet hodinového signálu se může pohybovat v rozmezí od 0 do 16 MHz. Paměť programu může být v provedení ROM, PROM nebo EPROM. (V literatuře popisující mikrořadiče se vyskytuje často název OTPROM, což je zkratka pro One-Time-Programmable ROM; tento název je ekvivalentní označení PROM.) Mikrořadič má dva režimy snížené spotřeby, “idle”, kdy se spotřeba z původních 70 mA na napájecím napětí 5 V sníží na cca 30 mA, a “power-down”, kdy je jeho spotřeba pouze cca 50 μA.

Rychlejší 16bitový jednočipového mikropočítač má shodné součásti a proto je nebudeme opakovat. Adresová sběrnice je oddělena od datové a má 8 vodičů; řádková a sloupcová adresa jsou časově multiplexovány. Adresovatelný prostor je proto 64 kB, z otho na čipu může být až 32 kB. Datová sběrnice má plnou šířku 16 vodičů, přenos dat mezi CPU, pamětí a periferiemi je proto rychlejší než v předchozím případě. Aritmeticko-logická jednotka má zabudovanou hardwarovou násobičku (16x16 bitů za 1,75 μs při taktovací frekvenci 16 MHz) a děličku (32/16 bitů za 3 μs při taktovací frekvenci 16 MHz). Mikrořadič obsahuje systém pro zrychlení obsluhy přerušení, označený na obrázku jako PTS, a jednotku pro zpracování událostí, označenou jako EPA. Tato jednotka “předzpracovává” číslicové signály z periferních čidel z hlediska splnění potřebných podmínek (například komparace, časová následnost, apod.) a šetří tak procesoru čas, o který je pak odezva na konkrétní stav vnějších čidel rychlejší. Kromě toho obsahuje mikrořadič na čipu ještě převodník A/D s programovatelným rozlišením 10 nebo 12 bitů (nižší rozlišení znamená kratší dobu převodu), který je vybaven 8-14 kanálovým multiplexerem rovněž na čipu. Mikrokontrolér je tak schopen reagovat i na analogové signály z vnějších čidel bez dodatečných součástek. Číslicových logických vstupů může mikrokontrolér využít až 53 a obsahuje rovněž dva programovatelné čitače/časovače a hlídací časovač (watchdog). Sběrnice mikrořadiče jsou vyvedeny pro připojení vnější paměti nebo dalších vstupně-výstupních obvodů.

Z uvedených příkladů je zřejmé, že jednočipový mikropočítač je a zůstane široce používanou elektronickou součástkou v jednoúčelových aplikacích, kde je rozsah řídicích funkcí předem znám. Velký výběr mikrořadičů nabízených různými výrobci umožňuje nalézt typ, jaký je pro danou aplikaci nejvhodnější.

Vlastní nasazení pak vyžaduje vyvinout řídicí program a vyzkoušet jej buď na zařízení simulujícím funkci jednočipového mikropočítače (emulátoru) nebo na mikrořadiči osazeném pamětí EPROM, kterou je možné v případě chyby v programu přeprogramovat. Tyto mikrokontroléry jsou opatřeny, podobně jako paměti EPROM, okénkem, kterým je možné obsah celé paměti programu smazat. V současné době nabízí celá řada prodejců vývojové prostředky pro programování jednočipových mikropočítačů, jejich popis a použití však již přesahuje rámec těchto skript.

Literatura:

1) Mueller Scott, Osobní počítač, Computer Press Praha 2001
2) Pelikán, J.: Architektura počítačů PC. Texty v elektronické podobě. Masarykova univerzita v Brně 1998
3) Brandejs, M.: Architektura počítačů. Texty v elektronické podobě. Masarykova univerzita v Brně 1998
4) Klimeš, C.: Prostředky automatického řízení II. Počítačové systémy Skripta VŠB Ostrava
5) Lysenko, V.: Základy elektrotechniky. Učební texty v elektronické podobě. Ostravská univerzita, 2001