Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta

VYBRANÉ PARTIE Z ALGEBRY A JEJICH APLIKACE

BAKALÁŘSKÁ PRÁCE

Olga Jakešová
Český Krumlov 2010

Prohlašuji, že svoji bakalářskou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě – (nebo v úpravě vzniklé vypuštěním vyznačených částí) archivovaných pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českém Krumlově 14. března 2010

Olga Jakešová

……

Anotace

Cílem bakalářské práce bylo vytvořit rozšiřující učební pomůcku použitelnou pro studenty předmětu „Algebra III.“ na pedagogické fakultě Jihočeské univerzity v Českých Budějovicích. Text je koncipován tak, aby byl vhodný jak pro studenty učitelství matematiky na ZŠ i SŠ, tak i pro studenty oboru finanční matematika. Ukazuje zejména na praktické aplikace části algebraických metod v reálném životě.

Poděkování

Na tomto místě bych ráda poděkovala prof. RNDr. Pavlu Tlustému, CSc., vedoucímu bakalářské práce, za vedení a odborné připomínky. Dále bych ráda poděkovala Bc. Pavlu Čurdovi za odborné rady a spolupráci na té části této bakalářské práce, která se týká praktické realizace převodníku kódu BCD na sedmisegmentový zobrazovač.

OBSAH:

1 ÚVOD

Číslo je abstraktní entita užívaná pro vyjádření množství nebo pořadí. Čísla se zapisují pomocí číslic, a to v různých číselných soustavách, a pomocných znaků, zejména desetinné čárky a znamének plus a mínus.

Kolem roku 3000 př. n. l. byly v Sumerské říši k vyjadřování množství obchodovaného zboží používány různě vytvarované centimetrové hliněné žetony, představující například jednu ovci, jednu domluvenou míru obilí nebo oleje apod. Počet žetonů pak vyjadřoval celkové domluvené množství zboží. Žetony byly po obchodním jednání uzavřeny do duté hliněné koule opatřené pečetěmi smluvních stran. Pro ověření množství bylo ovšem nutno pečetě porušit. Proto bylo kromě pečetí na kouli napsáno, jaké žetony v jakém množství se v ní nacházejí. Obchodníci si po čase všimli, že tak vlastně žetony nepotřebují.

[14]

2 ČÍSELNÉ OBORY

Historický vývoj pojmu číslo byl stimulován potřebou provádět operace s čísly. Nemožnost číselně vyjádřit určité skutečnosti z reálného života vedla k postupnému rozšiřování oboru přirozených čísel, tj. k definici nadmnožiny k dosavadnímu číselnému oboru. Zavedené číselné množiny jsou ve vztahu inkluze: $N \subset Z \subset Q \subset R$.

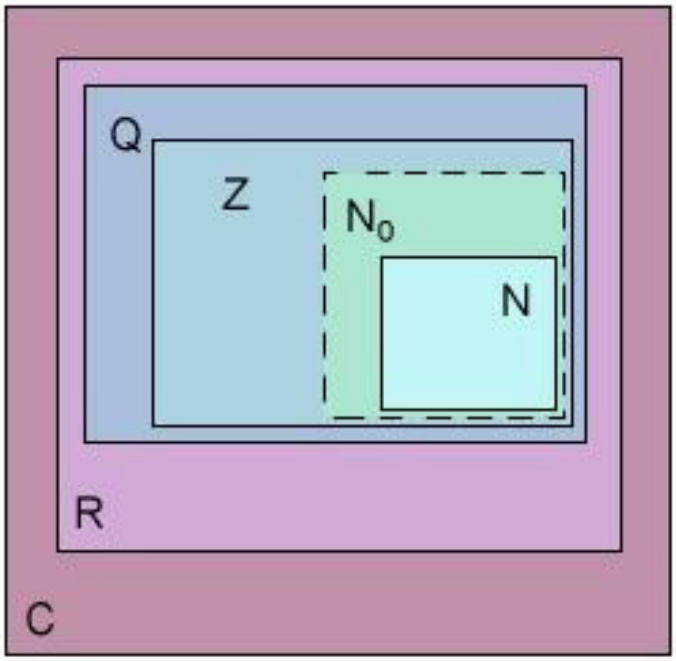
N – množina přirozených čísel, která slouží k vyjádření počtu prvků neprázdných množin (1, 2, 3, 4, 5, …). Někdy potřebujeme přidat i číslo 0, potom se používá symbol N_0 (0, 1, 2, 3, 4, 5, …).

Z – množina celých čísel, která umožňují vyjadřovat i změny v počtu prvků. Ke každému prvku existuje prvek opačný, což umožnilo vyjádřit rozdíl mezi „má dáti a dal“ (…, -3, -2, -1, 0, 1, 2, 3, …).

Q – množina racionálních čísel, která umožňují vyjádřit i počty dílů určitého celku, změny počtů apod. Obor celých čísel bylo potřeba rozšířit o čísla vyjadřující část celku.

R – množina všech reálných čísel, kterými lze vyjádřit i libovolné délky, obsahy, objemy apod. a jejich změny.

[1]



Obr. 1: Číselné obory

[14]

2.1 OBOR PŘIROZENÝCH ČÍSEL

$N = \{1, 2, 3, 4, \ldots\}$

2.1.1 Základní operace

Sčítání a násobení – pro každá tři přirozená čísla a, b, c platí:

a) $(a+b) \in N, a \cdot b \in N$ … uzavřenost na operace sčítání a násobení.

b) $a+b = b+a, a \cdot b = b \cdot a$ … komutativnost operací.

c) $a+(b+c) = (a+b)+c, a \cdot (b \cdot c) = (a \cdot b) \cdot c$ … asociativnost operací.

d) $a \cdot (b+c) = a \cdot b + a \cdot c$ … distributivnost násobení vzhledem ke sčítání.

Další operace

a) odčítání $(a-b)$, kde $a>b$: rozdílem dvou přirozených čísel a a b, kde platí $a>b$ je přirozené číslo x, pro které platí $a=b+x$.
b) mocnina (a^b): b-tou mocninou přirozeného čísla a nazýváme součin b stejných činitelů a ($a^b = a \cdot a \cdot a \cdot ... \cdot a$)

Soudělná čísla - přirozená čísla a,b jsou navzájem soudělná, mají-li společného dělitele většího než 1.

Prvočíslo - přirozené číslo nazýváme prvočíslem, má-li právě dva dělitele: číslo 1 a samo sebe.

Číslo složené - takové přirozené číslo, které má více než dva přirozené dělitele (1 není ani prvočíslo ani číslo složené).
Množina dělitelů čísla *n* - je množina všech přirozených čísel, jimiž je n dělitelné.

Největší společný dělitel čísel *a, b* – je největší z čísel, kterými je dělitelné jak číslo a, tak i číslo b. Značíme jej $D(a,b)$.

Množina násobků čísla *n* – je množina $\{n,2n,3n,4n,...,kn,...\}$, pro všechna přirozená čísla k.

Nejmenší společný násobek čísel *a, b* – je číslo $n(a,b)$, které je nejmenší ze všech čísel, které jsou násobky jak čísla a, tak i čísla b.

[1]

2.1.2 Kritéria dělitelnosti přirozených čísel

$n \in N$ je dělitelné

- dvěma: je sudé (právě tehdy, končí-li na některou z číslic 0, 2, 4, 6, 8).
- třemi: právě tehdy, je-li jeho ciferný součet dělitelný třemi.
- čtyřmi: právě tehdy, je-li jeho poslední dvojčíslí dělitelné čtyřmi.
- pěti: právě tehdy, je-li jeho poslední cifra dělitelná 5.
- šesti: právě tehdy, je-li dělitelné zároveň dvěma a třemi.
- sedmi: právě tehdy, když číslo, které vznikne z čísla n odtržením cifry na místě jednotek a odečtením dvojnásobku jednotek od takto „zkráceného čísla“, je dělitelné sedmi.
- osmi: právě tehdy, je-li poslední trojčíslí dělitelné osmi.
- devíti: právě tehdy, je-li jeho ciferný součet dělitelný devíti.
- deseti: právě tehdy, je-li jeho poslední cifra číslo 0.
- jedenácti: právě tehdy, je-li součet číslic na lichých místech zmenšený o součet číslic na sudých místech dělitelný jedenácti.
- další čísla: právě tehdy, je-li přirozené číslo n dělitelné zároveň dvěma nesoudělnými čísly a a b, pak je dělitelné také jejich součinem.

[1][3]

2.2 OBOR CELÝCH ČÍSEL

Obor celých čísel vytvoříme tak, že množinu čísel přirozených doplníme o číslo 0 a všechny rozdíly $0 - n$ pro všechna přirozená čísla n. Číslo $0 - n$ nazýváme číslem opačným k číslu n.

2.2.1 Dělitelnost v oboru celých čísel

Kritéria dělitelnosti v oboru přirozených čísel jsou použitelná i v oboru čísel celých, neboť dvojice navzájem opačných čísel *a, -a* mají společné celočíselné dělitele i společné celočíselné násobky. Za **nesoudělná** považujeme celá čísla a,b, jejichž společnými celočíselnými děliteli jsou pouze čísla -1 a 1.

[1]

2.3 OBOR RACIONÁLNÍCH ČÍSEL

Každé racionální číslo lze vyjádřit jako zlomek $\frac{p}{q}$, kde $p \in Z$ se nazývá čitatel a $q \in N$ se nazývá jmenovatel. Jmenovatel určuje jméno zlomku, např. $\frac{1}{2}$ je jedna polovina, $\frac{1}{3}$ je jedna třetina, $\frac{1}{4}$ je jedna čtvrtina. Zápis zlomků není jednoznačný, protože zlomek je možno zapsat nekonečně mnoho způsoby (krácení, rozšíření). Vydělením lze zlomek převést desetinné číslo.

Zápis racionálních čísel:

- pomocí zlomku v základním tvaru, kde čitatel a jmenovatel jsou nesoudělná čísla, např. $\frac{2}{5}$, $\frac{5}{3}$
- desetinným číslem s ukončeným rozvojem: 0,4
- symbolickým zápisem čísla s nekonečným, ale periodickým desetinným rozvojem: 1,66666… $\Rightarrow 1,\bar{6}$

Porovnávání racionálních čísel

- v desetinném zápisu: podle první rozdílné číslice zprava
- ve tvaru zlomku: po převodu na společného jmenovatele porovnáváme čitatele

Operace na množině racionálních čísel

Pro libovolná celá čísla a, c a celá nenulová čísla b, d platí:

$$\frac{a}{b}+\frac{c}{d}=\frac{ad+bc}{bd}, \frac{a}{b}-\frac{c}{d}=\frac{ad-bc}{bd}$$

$$\frac{a}{b}\cdot\frac{c}{d}=\frac{ac}{bd}, \frac{a}{b}:\frac{c}{d}=\frac{a}{b}\cdot\frac{d}{c}=\frac{ad}{bc}$$

[1]

2.4 OBOR REÁLNÝCH ČÍSEL

Reálná čísla jsou taková čísla, kterým můžeme jednoznačně přiřadit body nekonečné přímky (číselné osy) tak, aby tato čísla popisovala vzdálenost od nějakého vybraného bodu (nuly) na takové přímce. Tato nula pak přirozeně rozděluje reálná čísla na kladná a záporná. Do reálných čísel patří každý bod této nekonečné přímky.

2.4.1 Absolutní hodnota reálného čísla

Absolutní hodnota reálného čísla $|a|$ je velikost vzdálenosti obrazu tohoto čísla od obrazu 0 na číselné ose.

Je-li $a \geq 0 \Rightarrow |a| = a$

$a < 0 \Rightarrow |a| = -a$

2.4.2 Operace s reálnými čísly

Platí stejné věty jako pro operace s racionálními čísly. Pro nezáporný argument *a* a sudé *n* je navíc $\sqrt[n]{a} \in R_0^+$ (uzavřenost R_0^+ na odmocňování). Pokud je *n* liché, potom platí $\sqrt[n]{-a} = -\sqrt[n]{a}$.

[1]

2.5 OBOR KOMPLEXNÍCH ČÍSEL

Komplexní čísla vznikají rozšířením oboru reálných čísel tak, aby v něm každá algebraická rovnice měla řešení. Např. kvadratická rovnice $x^2 + 1 = 0$ nemá v oboru reálných čísel řešení, ale má řešení v oboru čísel komplexních.

Komplexní číslo má dvě složky **reálnou a imaginární**. Zapisuje se nejčastěji jako ***a+bi,*** přičemž ***i*** znamená **imaginární jednotku**, definovanou vztahem $i^2 = -1$. Zmíněná rovnice pak má dvě řešení $\pm i$. Pro operace s komplexními čísly platí pravidla jako pro počítání s dvojčleny.

2.5.1 Důvody pro zavedení komplexních čísel

Už perský matematik Al-Khwarizmi (asi 820) si všiml, že některé kvadratické rovnice nemají v množině reálných čísel řešení. Italský matematik Girolamo Cardano (1501-1576) ukázal, že by stačilo vhodně definovat odmocninu záporného čísla, a René Descartes zavedl v roce 1637 označení reálné a imaginární číslo. Zajímavé výsledky zkoumání těchto „neskutečných“ čísel ukázal Leonhard Euler a komplexní čísla přesně zavedl francouzský matematik Augustin Louis Cauchy (1821) a nezávisle na něm Carl Friedrich Gauss (1831).

Obor reálných čísel, který vyjadřuje dostatečně dobře jakoukoliv kvantitu (množství), se tedy rozšiřuje do oboru komplexních čísel, jejichž význam není intuitivně příliš zřejmý.

2.5.2 OPERACE S KOMPLEXNÍMI ČÍSLY

Algebraický tvar komplexních čísel – pro čísla v algebraickém tvaru lze jednoduchými algebraickými úpravami definovat vztahy pro součet, rozdíl a součin dvou komplexních čísel:

- $(a+bi)+(c+di)=(a+c)+i(b+d)$
- $(a+bi)-(c+di)=(a-c)+i(b-d)$
- $(a+bi)\cdot(c+di)=(ac-bd)+i(ad+bc)$

Podíl dvou komplexních čísel lze vyjádřit takto:

- $\frac{a+bi}{c+di}=\frac{(a+bi)\cdot(c-di)}{(c+di)\cdot(c-di)}=\frac{(ac+bd)+i(bc-ad)}{c^2+d^2}=\left(\frac{ac+bd}{c^2+d^2}\right)+i\left(\frac{bc-ad}{c^2+d^2}\right)$

Pro komplexní číslo $z=a+bi$ je definováno komplexně sdružené číslo ve tvaru $\bar{z}=a-bi$. Jejich součin $z\cdot\bar{z}=a^2+b^2$ je vždy reálný a nezáporný a je roven nule pouze když $z=0$.

[14]

3 ČÍSELNÉ SOUSTAVY

Číselná soustava je způsob reprezentace čísel. Podle způsobu určení hodnoty čísla z dané reprezentace rozlišujeme dva hlavní druhy číselných soustav: poziční číselné soustavy a nepoziční číselné soustavy. V praxi se však také používaly způsoby reprezentace používající postupy z obou těchto druhů. Dnes se obvykle používají soustavy poziční. Čísla dané soustavy se skládají z uspořádané množiny symbolů, které se nazývají číslice.

Poziční soustavy

Poziční soustavy jsou charakterizovány tzv. základem neboli bází (anglicky *radix,* značí se *r*), což je obvykle kladné celé číslo definující maximální počet číslic, které jsou v dané soustavě k dispozici. Mezi nejčastěji používané poziční soustavy patří dvojková, osmičková, desítková, dvanáctková, šedesátková.

Každé číslo vyjádřené v poziční soustavě může mít část celočíselnou a část desetinnou. Tyto části jsou odděleny desetinnou čárkou. V anglosaských zemích je místo desetinné čárky užívána desetinná tečka. Poziční soustavy se nazývají polyadické, což značí vlastnost, že číslo v nich zapsané lze vyjádřit součtem mocnin základu dané soustavy vynásobených příslušnými platnými číslicemi. Existují i soustavy, které využívají odečítání. Příkladem je trojková soustava, která obsahuje znaky s významem -1, 0, 1.

Nepoziční soustavy

Příkladem nepoziční soustavy jsou římské číslice. Dnes se prakticky nepoužívají. [14]

3.1 DESÍTKOVÁ SOUSTAVA

Desítková soustava je běžně používanou soustavou. Má pravděpodobně původ v počtu prstů na obou rukou.

Jednotlivé číslice udávají počty mocnin deseti:

$$(4385)_{10} = 4 \cdot 10^3 + 3 \cdot 10^2 + 8 \cdot 10^1 + 5 \cdot 10^0$$

Desítková soustava patří mezi polyadické soustavy, které se vyznačují tím, že číslo $\boldsymbol{a}$ je vyjádřeno jako mnohočlen, kde z je základ soustavy (v našem případě 10), součinitelé $\boldsymbol{a_0}$ až $\boldsymbol{a_n}$ mohou nabývat hodnot 0, 1, …, z-1 (v našem případě 0 až 9), obdobně $\boldsymbol{a_{-1}}$ až $\boldsymbol{a_{-n}}$:

$$a = a_n z^n + a_{n-1} z^{n-1} + ... + a_1 z^1 + a_0 z^0 + a_{-1} z^{-1} + a_{-2} z^{-2} + ... + a_{-k} z^{-k}$$

3.2 DVOJKOVÁ SOUSTAVA

Desítková soustava není však vhodná pro počítače a číslicové systémy, protože číslicové zařízení by muselo rozlišovat deset různých stavů (např. napěťových stupňů). To by kladlo vysoké nároky na jeho přesnost a kvalitu. Proto je výhodnější zobrazovat číslo ve dvojkové (binární) soustavě, kde mají prvky dva stavy. Čísla jsou zde vyjádřena pomocí číslic 0 a 1 jako součet mocnin dvou. Tak jako v desítkové soustavě má i zde každá číslice význam odpovídající jejímu umístění ve dvojkovém čísle.

Převod dvojkového čísla na desítkové provedeme tak, že řády, v nichž je ve dvojkovém čísle 1, vyjádříme v desítkové soustavě pomocí mocnin dvou:

$$(101000101)_2 = 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (325)_{10}$$

Číslo zapsané ve dvojkové soustavě má průměrně 3,3krát více míst než stejné číslo vyjádřené v desítkové soustavě. Proto se dvojková soustava nehodí pro ruční výpočty a je určena výhradně pro použití v počítačích.

[2][4][14]

3.2.1 Převod čísel z desítkové do dvojkové soustavy

Jednoduchý způsob převodu celých desítkových čísel je založen na postupném dělení dvěma a zbytky po dělení (0 nebo 1) tvoří obraz čísla ve dvojkové soustavě:

$(49)_{10} : 2 = 24$	zbytek: 1
$24 : 2 = 12$	zbytek: 0
$12 : 2 = 6$	zbytek: 0
$6 : 2 = 3$	zbytek: 0
$3 : 2 = 1$	zbytek: 1
$1 : 2 = 0$	zbytek: 1

Výsledné dvojkové číslo je $(110001)_2$

Tento způsob převodu vyžaduje operace s desítkovými čísly, proto se hodí pro ruční převod čísel z jedné soustavy do druhé. Pro počítače pracující ve dvojkové soustavě se používají převody jiné.

Desetinná čísla se převádí postupným násobením základem. Číslice, které jsou v dílčích výsledcích před řádovou čárkou, jsou číslicemi čísla v nové soustavě. Část za řádovou čárkou se dále násobí základem tak dlouho, dokud za řádovou čárkou nebude ve výsledku nula nebo podle toho, kolik míst chceme mít výsledek. Číslice musí být seřazeny podle sledu násobení za řádovou čárkou. Převod čísla $(0,78)_{10}$ do dvojkové soustavy:

$0{,}78 \cdot 2 = 1{,}56$	$0{,}56 \cdot 2 = 1{,}12$	$0{,}12 \cdot 2 = 0{,}24$	$0{,}24 \cdot 2 = 0{,}48$
1	1	0	0
$0{,}48 \cdot 2 = 0{,}96$	$0{,}96 \cdot 2 = 1{,}92$	$0{,}92 \cdot 2 = 1{,}84$	$0{,}84 \cdot 2 = 1{,}68$
0	1	1	1
$0{,}68 \cdot 2 = 1{,}36$	$0{,}36 \cdot 2 = 0{,}72$ … atd.		
1	0		

Výsledné dvojkové číslo po převodu je $(1100011110\ldots)_2$

[2]

3.2.2 Matematické operace ve dvojkové soustavě

Ve dvojkové soustavě je možné provádět všechny základní matematické operace obdobně jako v soustavě desítkové.

Sčítání dvojkových číslic

Při sčítání dvojkových číslic platí: **0 + 0 = 0**

0 + 1 = 1

1 + 1 = 10

Protože číslo dvě již nelze vyjádřit číslicí dvojkové soustavy, je potřeba použít dvou číslic a to v nultém a v prvním řádu, tedy $(10)_2$. Je-li součet větší než 1, dochází k přenosu řádu, který v desítkové soustavě nastane teprve při součtu větším než 9. Při sčítání čísel postupujeme analogicky jako u desítkové soustavy:

10111		sčítanec	23
11100		sčítanec	28
11100		přenos	1
110011	2	součet	51 10

Dvojkové odečítání

Máme-li od čísla *A* odečíst číslo *B*, tj. (*A*-*B*), můžeme postupovat tak, že číslo *B* učiníme záporným a přičteme k číslu *A*. Platí tedy (*A*-*B*) = *A*+(-*B*). Na tomto principu lze čísla odečíst s použitím sčítání. Musíme však nejdřív vytvořit zápornou hodnotu dvojkového čísla *B*. Jedna z možných metod používá dvojkově komplementární aritmetiku, tj. pomocí doplňkového kódu (také dvojkový doplněk), který se vytvoří tak, že u čísla zaměníme jedničky za nuly a nuly za jedničky (tím vytvoříme negaci neboli jedničkový doplněk) a k takto vzniklému číslu přičteme jedničku. Číslo se tedy neguje obrácením hodnoty všech bitů čísla a přičtením jedničky k výsledku negace. Pracujeme-li s dvojkovými čísly ve dvojkově komplementárním zápisu, udává vždy bit čísla, který je nejvíce vlevo znaménko čísla. Je to tzv. znaménkový bit. Je-li tento bit 1, je číslo záporné, je-li tento bit 0, je číslo kladné.

Např. dvojkově komplementární odečtení čísel 3 a 2:

$(010)_2$	číslo 2
$(011)_2$	číslo 3
$(101)_2$	negovaná hodnota čísla 2 (-2)
$(001)_2$	k negovanému číslu 2 přičteme číslo 1
$(110)_2$	negace po přičtení čísla 1
(011) + (110) = (001)	výsledek dvojkově komplementárního odečtení

Násobení dvojkových čísel

Při násobení dvojkových čísel platí tato pravidla: $0 \cdot 0 = 0$

$0 \cdot 1 = 0$

$1 \cdot 1 = 1$

Při násobení vícemístných čísel, obdobně jako v desítkové soustavě, násobíme postupně prvního činitele jednotlivými číslicemi druhého činitele a výsledky, vždy posunuté o řád, sečteme:

$$\begin{array}{r} 0011 \\ \cdot\,0101 \\ \hline 0011 \\ 0000 \\ 0011 \\ \hline 1111 \end{array}_{2} \qquad \begin{array}{r} 12 \\ \cdot\,13 \\ \hline 36 \\ 12 \\ \hline 156 \end{array}_{10}$$

Dělení ve dvojkové soustavě

Dělení ve dvojkové soustavě je poměrně složitá operace. U některých starších počítačů nepatřilo dělení k hlavním operacím a provádělo se zvláštním podprogramem. [2]

3.3 OSMIČKOVÁ A ŠESTNÁCTKOVÁ SOUSTAVA

Osmičková a šestnáctková číselná soustava jsou v informatice často používány ke stručnějšímu a přehlednějšímu zápisu binárního kódu.

3.3.1 Osmičková (oktávová) soustava

Osmičková soustava je číselná soustava o základu 8. Je snadno převoditelná do binární soustavy (číslo $8 = 2^3$). Odzadu rozdělíme binární znaky na trojice, a pokud je potřeba, dopíšeme na začátek jednu nebo dvě nuly. Poté můžeme každou trojici nahradit znakem 0, 1, 2, 3, 4, 5, 6, 7 a získáme zápis čísla v osmičkové číselné soustavě. Jedna číslice oktávového zápisu čísla nahrazuje tři číslice binárního zápisu. Pro převod můžeme použít následující tabulku:

Desítková číslice	0	1	2	3	4	5	6	7
Binární zápis	000	001	010	011	100	101	110	111

Tab. 1: Ukázka binárního zápisu

Převod čísla $(1572)_8$ do dvojkové (binární) soustavy:

1 = 001
5 = 101
7 = 111
2 = 010 výsledek: $(1572)_8 = (001101111010)_2$

Převod čísla $(11111011000)_2$ do osmičkové soustavy: jelikož máme pouze 11 binárních znaků, doplníme na začátku číslici 0 a poté číslo $(011111011000)_2$ rozdělíme do trojic:

011 = 3
111 = 7
011 = 3
000 = 0 výsledek: $(11111011000)_2 = (3730)_2$

Osmičková soustava už je dnes na ústupu a více se využívá soustava šestnáctková. [4][14]

3.3.2 Šestnáctková (hexadecimální) soustava

Šestnáctková soustava je číselná soustava o základu 16. Slovo „hexadecimální“ pochází z řeckého slova hexi = šest a latinského slova decem = deset. Hexadecimální číslice se zapisují pomocí číslic 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 a písmen *A, B, C, D, E, F* (písmena *A* až *F* reprezentují cifry s hodnotou 10 až 15). Díky jednoduchému převodu mezi šestnáctkovou a dvojkovou soustavou se hexadecimální zápis čísel často používá v oblasti informatiky. Převod čísla z hexadecimální soustavy do soustavy binární je usnadněn díky tomu, že základ soustavy číslo $16 = 2^4$.

Převod celých desítkových čísel na šestnáctkové

Celá desítková čísla můžeme převádět na šestnáctková např. pomocí postupného dělení šestnácti a sepisování zbytku po dělení.

Příklad: převod desítkového čísla 15119 do hexadekadického zápisu:

$15119 : 16 = 944$	zbytek: $15 = (F)_{16}$	
$944 : 16 = 59$	zbytek: $0 = (0)_{16}$	
$59 : 16 = 3$	zbytek: $11 = (B)_{16}$	
$3 : 16 = 0$	zbytek: $3 = (3)_{16}$	výsledek: $(15119)_{10} = (F0B3)_{16}$

Převod celých šestnáctkových čísel na desítkové

Výpočet hodnoty hexadecimálního čísla, které se skládá z k číslic x_0, x_1, …, $x_{(k-1)}$, nabývající hodnoty 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A, B, C, D, E, F* se provádí podle následujícího vzorce:

$$x = \sum_{i=0}^{k-1} x_i \cdot 16^i$$

Použijeme číslo $(15119)_{10} = (3B0F)_{16} \Rightarrow x_3 = 3, x_2 = B = 11, x_1 = 0, x_0 = F = 15$

$$(3B0F)_{16} = \sum_{i=0}^{k-1} x_i \cdot 16^i = x_3 \cdot 16^3 + x_2 \cdot 16^2 + x_1 \cdot 16^1 + x_0 \cdot 16^0 =$$

$$= 3 \cdot 16^3 + 11 \cdot 16^2 + 0 \cdot 16^1 + 15 \cdot 16^0 = 12288 + 2816 + 0 + 15 = (15119)_{10}$$

Převody mezi šestnáctkovou a dvojkovou soustavou

Převod z hexadecimální soustavy do dvojkové je usnadněn díky tomu, že číslo $16 = 2^4$. Postupujeme obdobně jako u osmičkové soustavy:

$(n)_{16}$	0	1	2	3	4	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*
$(n)_{10}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$(n)_2$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Tab. 2: Převody mezi šestnáctkovou, desítkovou a dvojkovou soustavou

Převod čísla $(3F5A)_{16} = (16218)_{10}$ do dvojkové soustavy:

3 = 0011
F = 1111
5 = 0101
A = 1010 výsledek: $(3F5A)_{16} = (11111101011010)_2$

Převod čísla $(11001101)_2$ do šestnáctkové soustavy:

1100 = *C*
1101 = *D* výsledek: $(11001101)_2 = (CD)_{16}$

[4][14]

3.4 OSTATNÍ ČÍSELNÉ SOUSTAVY

3.4.1 Šedesátková soustava

Používá se k měření času. Číslice se obvykle zapisují desítkovou soustavou jako 00 až 59 a řády se oddělují dvojtečkou. Názvy prvních dvou řádů jsou **kopa a velekopa.**

3.4.2 Dvanáctková soustava

Dvanáctková soustava Sumerů je dávána do spojitosti s šestiprstou lidskou rasou, která se vyskytuje v mýtech různých národů. Druhým důvodem pro tuto soustavu může být snazší dělení na třetiny oproti desítkové soustavě nebo fakt, že šestým symbolem jedné ruky byla sevřená pěst nebo prázdné místo (rutinní používání nuly je poměrně nová záležitost). Dvanáctková soustava je málo používaná, ale dodnes z ní zbyly názvy prvních dvou řádů **tucet** a **veletucet.**

3.4.3 Trojková (terciární) soustava

Ve trojkové soustavě se zapisuje např. **Morseova abeceda**, která obsahuje tři prvky (·−/). Pomocí tří prvků (SP TAB LF) může být také zapsán i program v ezoterickém programovacím jazyku Whitespace.

3.4.4 Unární (jednotková) soustava

Existuje také unární (jednotková) soustava, ve které je číslo v podstatě vyjádřeno opakováním jediného symbolu.

[14]

3.5 ZÁPIS ČÍSEL VE VYBRANÝCH ČÍSELNÝCH SOUSTAVÁCH

Desítkový zápis 10^1 10^0	**Dvojkový zápis** 2^3 2^2 2^1 2^0	**Oktálový zápis** 8^1 8^0	**Hexadec. zápis** 16^1 16^0
0	**0 0 0 0**	**0**	**0**
1	**0 0 0 1**	**1**	**1**
2	**0 0 1 0**	**2**	**2**
3	**0 0 1 1**	**3**	**3**
4	**0 1 0 0**	**4**	**4**
5	**0 1 0 1**	**5**	**5**
6	**0 1 1 0**	**6**	**6**
7	**0 1 1 1**	**7**	**7**
8	**1 0 0 0**	**10**	**8**
9	**1 0 0 1**	**11**	**9**
10	**1 0 1 0**	**12**	***A***
11	**1 0 1 1**	**13**	***B***
12	**1 1 0 0**	**14**	***C***
13	**1 1 0 1**	**15**	***D***
14	**1 1 1 0**	**16**	***E***
15	**1 1 1 1**	**17**	***F***

Tab. 3: Zápis čísel ve vybraných číselných soustavách

Příklad: $(12)_{10} = (1100)_2 = (14)_8 = (C)_{16}$

[4]

4 LOGICKÉ FUNKCE

Logický obvod je tvořen skupinou vzájemně spojených logických členů, o kterých lze říci, že jsou zařízení uskutečňující logickou funkci. S logickými funkcemi se můžeme setkat ve výrokové logice. Výroková logika jako obecný soubor pravidel správného usuzování vznikla z potřeb matematiky, přírodních věd i rétoriky již ve 4. století př. n. l. ve starověkém Řecku.

[1][2]

4.1 ZÁKLADNÍ POJMY

Výrok: každé tvrzení, kterému lze jednoznačně přiřadit pravdivostní hodnotu

pravdivostní hodnota výroku – pravda (1) $\Rightarrow$ výrok platí

nepravda (0) $\Rightarrow$ výrok neplatí

Př.: Dnes je středa.

$1+1=3$

Každé prvočíslo je liché.

Hypotéza: výrok, o jehož pravdivosti momentálně neumíme rozhodnout (domněnka)

Př.: Mimo Sluneční soustavu neexistuje život.

Výroková forma: je věta obsahující proměnnou (proměnné), stane se výrokem teprve při dosazení konkrétních hodnost za proměnné nebo kvantifikování

Př.: Číslo *n* je dělitelné šesti

$x+2 \geq 3$

Kvantifikovaný výrok: vymezuje rozsah (počet objektů, o nichž vypovídá)

Př.: Všichni to pochopili.

Nejméně dva žáci chyběli.

- **Obecné kvantifikované výroky –** přisuzují určitou vlastnost všem uvažovaným objektům bez výjimky. Obsahují slova: všichni, každý, žádný, nic apod.

  Symbolický zápis: $\forall x; T(x)$: tvrzení T platí pro každé x

  Př. Každý svého štěstí strůjcem.

  Nikdo není neomylný.

- **Existenční kvantifikované výroky –** vypovídají o tom, že existuje (aspoň jeden) objekt dané vlastnosti. Obsahují slova: existuje, najde se, některý, někdo, alespoň jeden, lze nalézt apod.

  Symbolický zápis: $\exists x; T(x)$: existuje x, pro které platí tvrzení T

  Př. Někteří to pochopili.

  Lze nalézt sudé prvočíslo.

Tautologie: je výrok, který je pravdivý, ať jsou dílčí výroky jakékoliv

Př.: Pachatelem je Petr nebo Pavel nebo nikdo z této dvojice.

Úsudek: je akt myšlení, který má tuto strukturu: známe pravdivostní hodnoty jednoho či více výroků (tzv. předpokladů) a přiřadíme pravdivostní hodnotu dalšímu výroku (výrokům) a provedeme závěr.

Př.: Když klesá tlak, zkazí se počasí.

Dnes klesá tlak. $\Rightarrow$ předpoklady

Dnes se zkazí počasí. $\Rightarrow$ závěr

Logická funkce: je předpis, který kombinaci, popř. i sledu hodnot jedné nebo více (nezávislých) logických proměnných jednoznačně přiřazuje hodnoty jedné (závislé) proměnné.

[1]

4.2 LOGICKÉ FUNKCE JEDNÉ PROMĚNNÉ

X – vstupní nezávisle proměnná

Y – výstupní závisle proměnná, $Y = f(X)$

- ***Y1* – FALSUM** hodnota *Y* je vždy 0
- ***Y2* – NEGACE** hodnota *Y* je vždy opačná než hodnota *X* ($Y = \overline{X}$)
- ***Y3* – ASERCE** hodnota *Y* opakuje hodnotu *X*
- ***Y4* – VERUM** hodnota *Y* je vždy 1

X	*Y1*	*Y2*	*Y3*	*Y4*
1	0	0	1	1
0	0	1	0	1

Tab. 4: Tabulka pravdivostních hodnot logických funkcí jedné proměnné

4.3 LOGICKÉ FUNKCE DVOU PROMĚNNÝCH

X – vstupní nezávisle proměnná

Y – výstupní závisle proměnná, $Y = f(X)$

Y1 – FALSUM

- Funkční hodnota vždy 0

X1	*X2*	*Y1*
1	1	0
1	0	0
0	1	0
0	0	0

Tab. 5: Funkce falsum

Y2 – KONJUNKCE (SOUČIN)

- Pokud jsou obě nezávisle proměnné 1, pak je výsledek 1
- Analogie s násobením $\Rightarrow$ lze použít spojku „i“

X1	*X2*	*Y2*
1	1	1
1	0	0
1	0	0
0	0	0

Tab. 6: Funkce konjunkce

Y3 – INHIBICE *X1, Y5* – INHIBICE *X2*

- Opakuje nezávisle proměnnou hodnotu 1, pokud je druhá proměnná opačné hodnoty

X1	*X2*	*Y3*	*Y5*
1	1	0	0
1	0	1	0
0	1	0	1
0	0	0	0

Tab. 7: Funkce inhibice

Y4 – ASERCE *X1, Y6* – ASERCE *X2*

- Opakuje hodnotu zvolené nezávisle proměnné

X1	*X2*	*Y4*	*Y6*
1	1	1	1
1	0	1	0
0	1	0	1
0	0	0	0

Tab. 8: Funkce aserce

Y7 – DILEMA (XOR)

- Taky někdy označovaná jako exklusive OR
- Výběr pravdivé hodnoty ze dvou různých

X1	*X2*	*Y7*
1	1	0
1	0	1
0	1	1
0	0	0

Tab. 9: Funkce dilema

Y8 – DISJUNKCE (SOUČET)

- Aspoň jedna proměnná nabývá hodnotu 1, potom výsledek je 1
- Analogie součtu $\Rightarrow$ lze použít spojku „nebo“

X1	*X2*	*Y8*
1	1	1
1	0	1
0	1	1
0	0	0

Tab. 10: Funkce disjunkce

Y9 – PIERCOVA FUNKCE NOR

- Negace součtu *Y8* – disjunkce
- Kombinací této funkce lze vyjádřit všechny ostatní logické funkce

X1	*X2*	*Y8*	*Y9*
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

Tab. 11: Piersova funkce NOR

Y10 – EKVIVALENCE

- Porovnání hodnot nezávisle proměnných
- Nabývá hodnoty 1, pokud mají obě proměnné stejnou hodnotu

X1	*X2*	*Y10*
1	1	1
1	0	0
0	1	0
0	0	1

Tab. 12: Funkce ekvivalence

Y11 – NEGACE *X2*, *Y13* – NEGACE *X1*

- Analogie funkce jedné proměnné
- Hodnoty *Y* nabývají opačných hodnot než *X*

X1	*X2*	*Y11*	*Y13*
1	1	0	0
1	0	1	0
0	1	0	1
0	0	1	1

Tab. 13: Funkce negace

Y12 – IMPLIKACE $X2 \Rightarrow X1$

- Hodnota *X1* vyplývá z *X2*
- Z nepravdy může plynout pravda, proto hodnota závisle proměnné je 1
- Z pravdy nemůže vyplynout nepravda, protože hodnota závisle proměnné je 0

X1	*X2*	*Y12*
1	1	1
1	0	1
0	1	0
0	0	1

Tab. 14: Funkce implikace $X2 \Rightarrow X1$

Y14 – FUNKCE IMPLIKACE $X1 \Rightarrow X2$

- Hodnota *X2* vyplývá z *X1*
- Z nepravdy může vyplynout pravda, proto hodnota závisle proměnné je 1
- Z pravdy nemůže vyplynout nepravda, proto hodnota závisle proměnné je 0

X1	*X2*	*Y14*
1	1	1
1	0	0
0	1	1
0	0	1

Tab. 15: Funkce implikace $X1 \Rightarrow X2$

Y15 – SHEFFEROVA FUNKCE NAND

- Negace součinu *Y2*
- Kombinací této funkce lze vyjádřit všechny ostatní logické funkce

X1	*X2*	*Y2*	*Y15*
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

Tab. 16: Shefferova funkce NAND

Y16 – VERUM

- Funkční hodnota vždy 1

X1	*X2*	*Y16*
1	1	1
1	0	1
0	1	1
0	0	1

Tab. 17: Funkce verum

SHRNUTÍ LOGICKÝCH FUNKCÍ DVOU PROMĚNNÝCH

Y1	**falsum**	***Y5***	**inhibice *X2***
Y2	**konjunkce (součin)**	***Y6***	**aserce *X2***
Y3	**inhibice *X1***	***Y7***	**dilema (XOR)**
Y4	**aserce *X1***	***Y8***	**disjunkce (součet)**
Y9	**Piercova funkce NOR**	***Y13***	**negace *X1***
Y10	**ekvivalence**	***Y14***	**implikace $X1 \Rightarrow X2$**
Y11	**negace *X2***	***Y15***	**Shefferova funkce NAND**
Y12	**implikace $X2 \Rightarrow X1$**	***Y16***	**verum**

X1	*X2*	*Y1*	*Y2*	*Y3*	*Y4*	*Y5*	*Y6*	*Y7*	*Y8*	*Y9*	*Y10*	*Y11*	*Y12*	*Y13*	*Y14*	*Y15*	*Y16*
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Tab. 18: Přehled všech logických funkcí dvou proměnných

[5][16]

5 BOOLEOVA ALGEBRA

Pro zjednodušování složených výroků byla v roce 1847 propracována anglickým matematikem Georgem Boolem (1815-1864) algebra logiky označovaná jako Booleova algebra. Na dlouhou dobu však upadla v zapomnění a její význam byl doceněn až v souvislosti s reléovými obvody a s rozvojem číslicové techniky.

Obr. 2: George Boole

Je třeba vzít na vědomí, že Booleova algebra není algebrou čísel, s jakou se setkáváme v matematice, ale je to algebra stavu. Vzhledem ke klasické algebře, je proto jinak definovaná, např. v ní vůbec neexistují operace odčítání a dělení.

Booleova algebra je zvláštní matematický způsob popisu chování i struktury logických obvodů. Tato algebra pracuje s logickými proměnnými, které mohou nabývat pouze dvou hodnot, logické hodnoty ‚0' a logické hodnoty ‚1'.

Pravidla Booleovy algebry můžeme rozdělit do několika skupin. Každému pravidlu pro logický součin odpovídá obdobné pravidlo pro logický součet. Říkáme, že funkce logického součtu a součinu jsou navzájem duální.

[15]

5.1 ZÁKLADNÍ OPERACE BOOLEOVY ALGEBRY

Logická negace NOT:

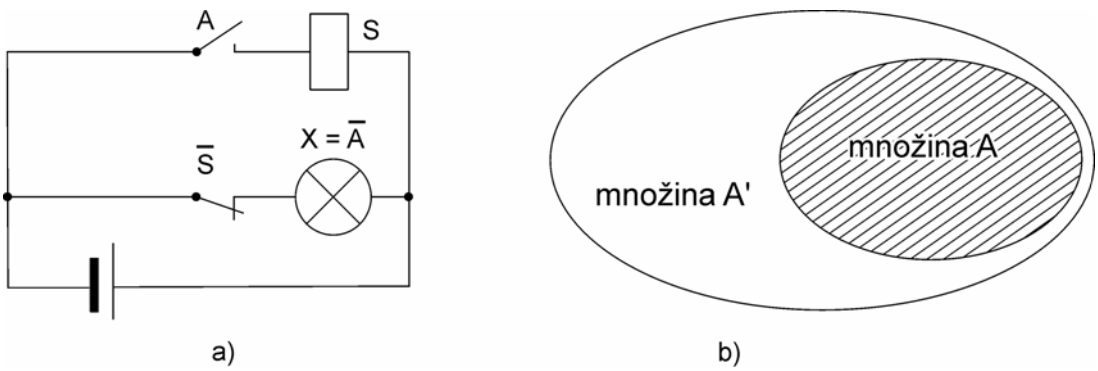
Tento logický operátor se nazývá také inverze, dává výsledek zvaný negace. Označujeme se přidáním „pruhu“ nad proměnnou $\overline{A}$, můžeme se ovšem setkat i s takovým zápisem, kdy bude negace vyznačena lomítkem před proměnnou $/A$.

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

Tab. 19: Pravdivostní tabulka logické negace

Znázornění logické negace (obr. 3):

a) rozpínacím kontaktem – při stisknutí tlačítka A relé rozpojí kontakt S a žárovka zhasne
b) množina A` je doplňkem (negací) množiny A



Obr. 3: Znázornění logické negace

Logický součin AND:

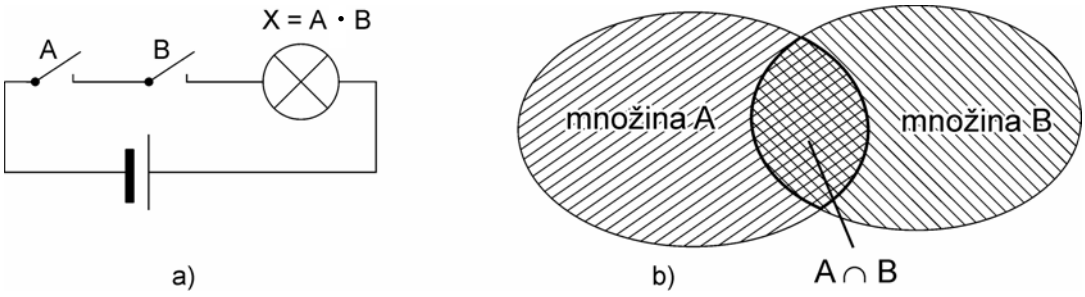
Logický součin se nazývá také průnik nebo konjunkce, vytváří „součin“ proměnných, obvykle se zapisuje $A \cdot B$.

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

Tab. 20: Pravdivostní tabulka logického součinu

Znázornění logického součinu (obr. 4):

a) sériovým zapojením kontaktů – žárovka se rozsvítí pouze tehdy, když jsou sepnuty oba spínače
b) průnikem množin



Obr. 4: Znázornění logického součinu

Logický součet OR:

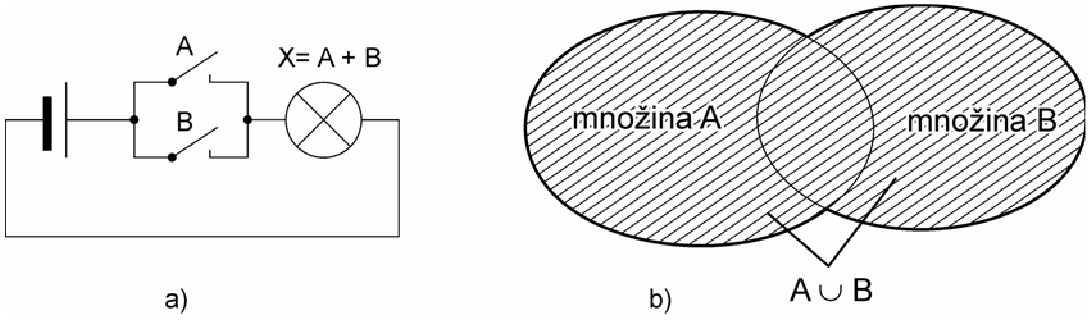
Logický součet se nazývá sjednocení nebo také disjunkce, výsledkem je „součet“ proměnných, obvykle se zapisuje jako $A + B$.

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 21: Pravdivostní tabulka logického součtu

Znázornění logického součtu (obr. 5):

a) paralelním zapojením kontaktů – žárovka se rozsvítí, pokud je sepnut alespoň jeden spínač
b) sjednocením množin



Obr. 5: Znázornění logického součtu

[15]

5.2 ZÁKLADNÍ PRAVIDLA

Booleova algebra je dvouhodnotová logická algebra, používající logického součtu, logického součinu a logické negace jako úplného systému základních logických funkcí. Používá se k úpravě a zjednodušování (minimalizaci) logických funkcí. Obsahuje následující zákony a pravidla:

- **Pravidlo agresívnosti a neutrálnosti hodnot 0 a 1:**

$A+1=1$ $\qquad$ $A\cdot 0=0$

$A+0=A$ $\qquad$ $A\cdot 1=A$

- **Pravidlo komutativní:** vyjadřuje, že nezáleží na pořadí proměnných

$A+B=B+A$ $\qquad$ $A\cdot B=B\cdot A$

- **Pravidlo asociativní:**

$A+(B+C)=(A+B)+C=A+B+C$

$A\cdot(B\cdot C)=(A\cdot B)\cdot C=A\cdot B\cdot C$

- **Pravidlo distributivní:**

$A+(B\cdot C)=(A+B)\cdot(A+C)$ $\qquad$ $A\cdot(B+C)=A\cdot B+A\cdot C$

- **Pravidlo absorpce:**

$A+A=A$ $\qquad$ $A\cdot A=A$

$A+A\cdot B=A$ $\qquad$ $A\cdot(A+B)=A$

- **Pravidlo absorpce negace:**

$$A + \overline{A} \cdot B = A + B \qquad A \cdot (\overline{A} + B) = A \cdot B$$

$$\overline{A} + A \cdot B = \overline{A} + B \qquad \overline{A} \cdot (A + B) = \overline{A} \cdot B$$

- **Pravidlo o vyloučeném třetím:**

$$A + \overline{A} = 1 \qquad A \cdot \overline{A} = 0$$

- **Pravidlo dvojité negace:**

$$\overline{\overline{A}} = A \qquad \overline{\overline{A + B}} = A + B \qquad \overline{\overline{A \cdot B}} = A \cdot B$$

- **Pravidla vytvoření negace – De Morganovy zákony:**

$$\overline{A + B} = \overline{A} \cdot \overline{B} \qquad \overline{A \cdot B} = \overline{A} + \overline{B}$$

$$A \cdot B = \overline{\overline{A} + \overline{B}} \qquad A + B = \overline{\overline{A} \cdot \overline{B}}$$

[2]

5.2.1 Důkaz De Morganova zákona

Všechna zde uvedená základní pravidla je možné dokázat tak, že ověříme jejich platnost pro všechny kombinace nezávisle proměnných. Postupujeme tak, že sestavíme pravdivostní tabulku pro levou a pravou stranu rovnice. Jsou-li obě výsledné strany obou tabulek shodné, pak je rovnost prokázána.

Příklad 1: Dokažte rovnost $A + B = \overline{\overline{A} \cdot \overline{B}}$

- vytvoříme tabulku pro přímé a negované vstupní proměnné
- provedeme matematické operace dle zadání
- porovnáme výsledky – jsou-li shodné, je rovnost dokázána

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$	$\overline{\overline{A} \cdot \overline{B}}$	$A + B$
0	**0**	**1**	**1**	**1**	**0**	**0**
0	**1**	**1**	**0**	**0**	**1**	**1**
1	**0**	**0**	**1**	**0**	**1**	**1**
1	**1**	**0**	**0**	**0**	**1**	**1**

Tab. 19: Důkaz De Morganova zákona (pro n proměnných má tabulka 2^n řádků)

Příklad 2: Dokažte platnost De Morganova zákona pro více proměnných:

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

Použijeme asociativní zákona, dostaneme tak součtový tvar pro který platí:

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

Obdobně můžeme rozšířit platnost De Morganova zákona na libovolný počet proměnných.

[4][15]

5.3 MINIMALIZACE

Při návrhu logického obvodu dbáme na to, aby jej bylo možno realizovat pokud možno co nejmenším počtem co nejjednodušších logických členů. To znamená, že algebraický výraz má být složen z co nejmenšího počtu členů, z nichž každý obsahuje nejmenší počet proměnných. Postup, pomocí kterého nahradíme složitější algebraický výraz jednodušším výrazem, nazýváme minimalizací.

Způsob úpravy algebraického výrazu, který využívá všech známých zákonů a pravidel Booleovy algebry se nazývá **přímá minimalizace.**

Příklady zjednodušování výrazů:

Příklad 1:

$$Z = AB\overline{C}D + \overline{A}B\overline{C}D + ABCD + \overline{A}BCD$$

$$Z = BD(A\overline{C} + \overline{AC} + AC + \overline{A}C)$$

$$Z = BD\left[\overline{C}(A + \overline{A}) + C(A + \overline{A})\right]$$

$$Z = BD(\overline{C} + C)$$

$$Z = BD$$

Příklad 2:

$$Z = \overline{A} + AB$$

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

$$Z = \overline{A}B + \overline{AB} + AB$$

$$Z = \overline{A}B + \overline{A}B + \overline{AB} + AB$$

$$Z = \overline{A}(B + \overline{B}) + B(A + \overline{A})$$

$$Z = \overline{A} + B$$

Příklad 3:

$$Z = (A + B)(A + C)$$

$$Z = A + AB + AC + BC$$

$$Z = A(1 + B + C) + BC$$

$$Z = A + BC$$

Příklad 4:

$$Z = \overline{A}BC + A\overline{B} + \overline{A} \cdot \overline{B} + ABC$$

$$Z = BC(A + \overline{A}) + \overline{B}(\overline{A} + A)$$

$$Z = BC + \overline{B} \qquad \text{(pravidlo absorpce negace)}$$

$$Z = \overline{B} + C$$

[4]

5.4 PRAVDIVOSTNÍ TABULKA

Pravdivostní tabulky jsou nástroj vyvinutý Charlesem Peirceem. Pravdivostní tabulka se používá v logice a pomáhá stanovit, zda je výraz pravdivý nebo zda je argument platný. Ukazují hodnoty, vztahy a výsledky logických operací na logických výrazech. Sloupcová záhlaví na pravdivostní tabulce ukazují vstupní proměnné a výstupní výrazy. Řady ukazují každou možnou kombinaci vstupů (jednu kombinaci na řádek) a výstupy, které vyplývají z každé kombinace vstupů.

[14]

5.4.1 Pravdivostní tabulka úplně zadané logické funkce

Logická funkce je úplně zadaná v případě, že je známá její hodnota (logická 0 nebo logická 1) pro všechny možné kombinace vstupních proměnných.

Př.: ***f (A, B, C)***

A	*B*	*C*	*f*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Tab. 20: Pravdivostní tabulka úplně zadané logické funkce

[15]

5.4.2 Pravdivostní tabulka neúplně zadané logické funkce

Logická funkce je neúplně zadaná tehdy, když její hodnota pro některé kombinace vstupních proměnných je libovolná, není určena nebo z fyzikálních důvodů nemůže nastat. Funkci pak označujeme obvykle znakem „*X*“.

Př.: ***f(A, B, C)***

A	*B*	*C*	*f*
0	**0**	**0**	**1**
0	**0**	**1**	**0**
0	**1**	**0**	**1**
0	**1**	**1**	***X***
1	**0**	**0**	***X***
1	**0**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**1**	***X***

Tab. 21: Pravdivostní tabulka neúplně zadané logické funkce

[15]

5.4.3 Řešení příkladu pomocí pravdivostní tabulky

Příklad: Sestavte logickou funkci pro ovládání žárovky ze dvou míst (schodišťový přepínač).

Řešení: Nejprve sestavíme pravdivostní tabulku pro dvě proměnné, kde spínače budou označeny *A, B* a žárovka *Z*.

A	*B*	*Z*
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Tab. 22: Pravdivostní tabulka

Z tabulky zapíšeme logickou funkci:

$Z = \overline{A} \cdot B + A \cdot \overline{B}$ (součet součinů)

$Z = (A + B) \cdot (\overline{A} + \overline{B})$ (součin součtů)

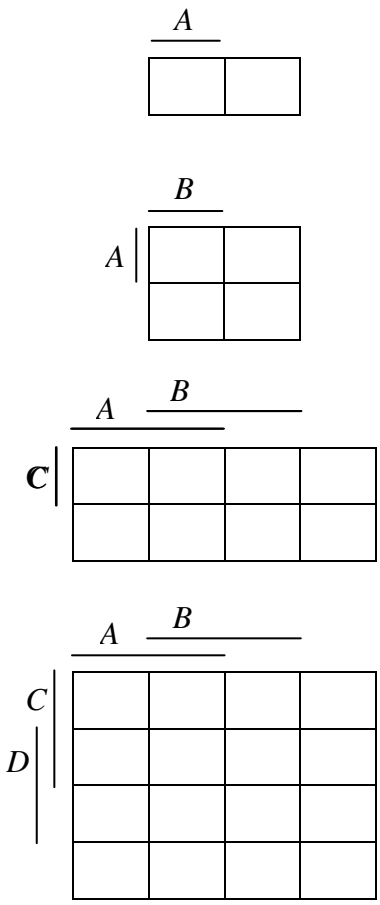
Algebraický výraz můžeme z tabulky dostat ve dvou tvarech:

- **Výraz ve tvaru součtu součinů:** dostaneme jej, napíšeme-li pro každé políčko, které obsahuje 1 pro výstupní proměnnou, kombinaci odpovídajících vstupních proměnných ve formě součinu a tyto jednotlivé součiny pak spojíme v součet.
- **Výraz ve tvaru součinu součtů:** dostaneme jej, napíšeme-li pro každé políčko, které obsahuje 0 pro výstupní proměnnou, kombinaci odpovídajících vstupních proměnných v podobě součtu a tyto součty pak spojíme v součin.

[2][15]

5.5 KARNAUGHOVY MAPY

Pro zobrazení logické funkce se kromě tabulky stavů používá také Karnaughova mapa. Slouží hlavně pro minimalizaci funkce. Její použití je výhodné maximálně pro čtyři proměnné. Použití pro více proměnných je již poměrně složité. Na rozdíl od tabulky stavů se hodnoty vstupních proměnných zapisují vedle mapy nebo nad ní a výstupní proměnná se zapisuje do políček mapy.



Obr. 6: Karnaughovy mapy pro jednu, dvě, tři a čtyři proměnné

[2]

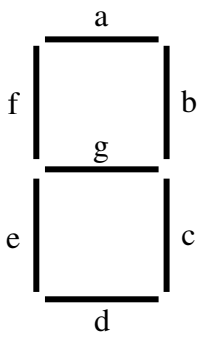
5.5.1 Minimalizace logických funkcí Karnaughovou mapou

Minimalizace logických funkcí pomocí Karnaughovy mapy je velmi rychlá a přehledná. Při této metodě vytváříme ze sousedních políček obsahujících logické 1 pravoúhlé plochy (smyčky). Počet polí ve smyčce může být pouze $\mathbf{2^n}$ (1, 2, 4, 8, atd.). Čím větší smyčka, tím dochází k větší minimalizaci a odpovídající výraz je jednodušší. Smyčky se mohou navzájem protínat. Za sousední políčka jsou považována také krajní a rohová políčka mapy, jako kdyby tvořila válcovou nebo kulovou plochu. Smyčky s logickými 1 odpovídají výrazu ve tvaru součtu součinů. Lze také vytvářet smyčky s logickými 0. V tomto případě však získáme výraz ve tvaru součinu součtů.

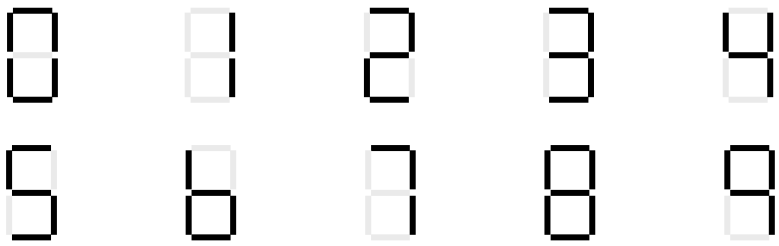
[2]

5.5.2 Praktický příklad využití Karnaughovy mapy

Nyní si celý postup minimalizace pomocí Karnaughovy mapy objasníme na příkladu převodníku kódu BCD na sedmisegmentový zobrazovač.



Obr. 7: Sedmisegmentový zobrazovač s popisem jednotlivých segmentů



Obr. 8: Číslice 0-9 zobrazené pomocí sedmisegmentového zobrazovače

n	Vstupní proměnné				Výstupní proměnné (segmenty)						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	0	0	0	0	0	0	0
11	1	0	1	1	0	0	0	0	0	0	0
12	1	1	0	0	0	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0	0	0
14	1	1	1	0	0	0	0	0	0	0	0
15	1	1	1	1	0	0	0	0	0	0	0

Tab. 23: Pravdivostní tabulka převodníku BCD na sedmisegmentový zobrazovač

Zápis segmentu a:

a) Logická funkce bez použití minimalizace:

$$a = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C \cdot D +$$
$$+ A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D$$

b) Logická funkce po použití minimalizace pomocí Karnaughovy mapy:

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

Tab. 24: Minimalizace segmentu **a** pomocí Karnaughovy mapy

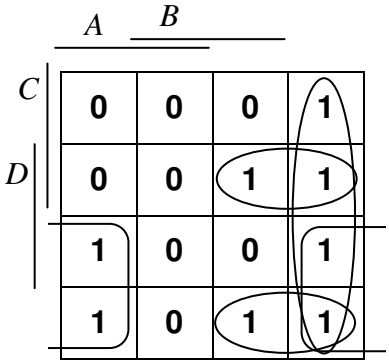
$$a = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot D + A \cdot \overline{B} \cdot \overline{C} + \overline{B} \cdot \overline{C} \cdot \overline{D}$$

Popis použití Karnaughovy mapy:

- sdružujeme sousední políčka, která obsahují logickou 1 $\Rightarrow$ vytváříme smyčky
- velikost smyčky vždy odpovídá mocnině 2^n (1, 2, 4, 8 políček)
- snažíme se o vytvoření co největších smyček $\Rightarrow$ čím vyšší počet políček, tím dojde k větší minimalizace (k popisu je potřeba méně proměnných)
- snažíme se o vytvoření co nejmenšího počtu smyček, musíme však pokrýt všechny logické 1
- smyčky se mohou překrývat $\Rightarrow$ jedna logická 1 může být použita současně pro více smyček

[2][15]

Zápis segmentu **b**:

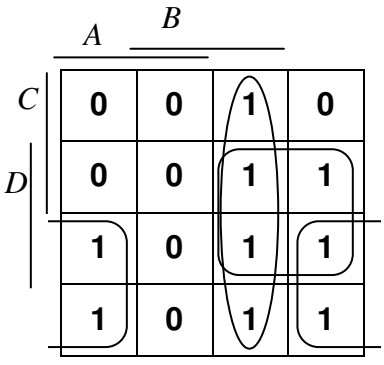


Tab. 25: Minimalizace segmentu **b** pomocí Karnaughovy mapy

Původní zápis: $b = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} +$
$+ \overline{A} \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D$

Zápis po minimalizaci: $b = \overline{A} \cdot \overline{B} + \overline{B} \cdot \overline{C} + \overline{A} \cdot C \cdot D + \overline{A} \cdot \overline{C} \cdot \overline{D}$

Zápis segmentu **c**:



Tab. 26: Minimalizace segmentu **c** pomocí Karnaughovy mapy

Původní zápis: $c = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot D +$
$+ \overline{A} \cdot B \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D$

Zápis po minimalizaci: $c = \overline{A} \cdot B + \overline{A} \cdot D + \overline{B} \cdot \overline{C}$

Zápis segmentu d:

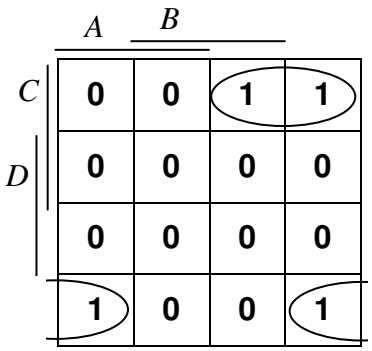
C \ D	A B = 00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	0	0	1	0
10	1	0	0	1

Tab. 27: Minimalizace segmentu **d** pomocí Karnaughovy mapy

Původní zápis: $d = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C \cdot \overline{D} +$
$+ A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$

Zápis po minimalizaci: $d = \overline{A} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C + \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot D$

Zápis segmentu e:

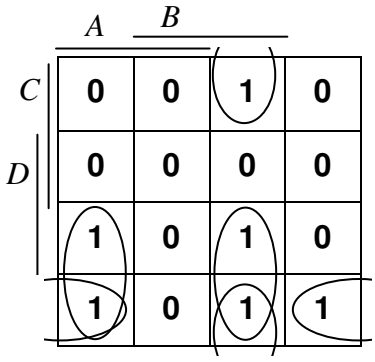


Tab. 28: Minimalizace segmentu **e** pomocí Karnaughovy mapy

Původní zápis: $e = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$

Zápis po minimalizaci: $e = \overline{A} \cdot C \cdot \overline{D} + \overline{B} \cdot \overline{C} \cdot \overline{D}$

Zápis segmentu f:



Tab. 29: Minimalizace segmentu **f** pomocí Karnaughovy mapy

Původní zápis: $f = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} +$

$+ A \cdot \overline{B} \cdot \overline{C} \cdot D$

Zápis po minimalizaci: $f = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{D} + \overline{B} \cdot \overline{C} \cdot \overline{D}$

Zápis segmentu g:

C \ A B	$\overline{A}\,\overline{B}$	$\overline{A}B$	$A\overline{B}$	AB
$\overline{C}\,\overline{D}$	0	0	1	1
$\overline{C}D$	0	0	0	1
$C\overline{D}$	1	0	1	0
CD	1	0	1	0

Tab. 30: Minimalizace segmentu **g** pomocí Karnaughovy mapy

Původní zápis: $g = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C \cdot \overline{D} +$

$+ A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D$ ¨

Zápis po minimalizaci: $g = \overline{A} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}$

6 PRAKTICKÁ REALIZACE PŘEVODNÍKU

V této kapitole bude nastíněna praktická realizace převodníku BCD na sedmisegmentový zobrazovač. Převodník bude realizován na základě teoretického rozboru z předcházející kapitoly. Vlastní realizace bude provedena třemi nezávislými způsoby:

- logickými hradly řady 74
- mikrokontrolérem s jádrem AVR®
- programovatelným hradlovým CPLD® polem

Tím bude názorně ukázána značná variabilita dostupných řešení tohoto i podobných úkolů, díky čemuž budou nastíněny moderní trendy v této oblasti. V závěru kapitoly budou shrnuty výhody a nevýhody jednotlivých řešení praktické realizace převodníku BCD na sedmisegmentový zobrazovač.

6.1 REALIZACE LOGICKÝMI FUNKCEMI

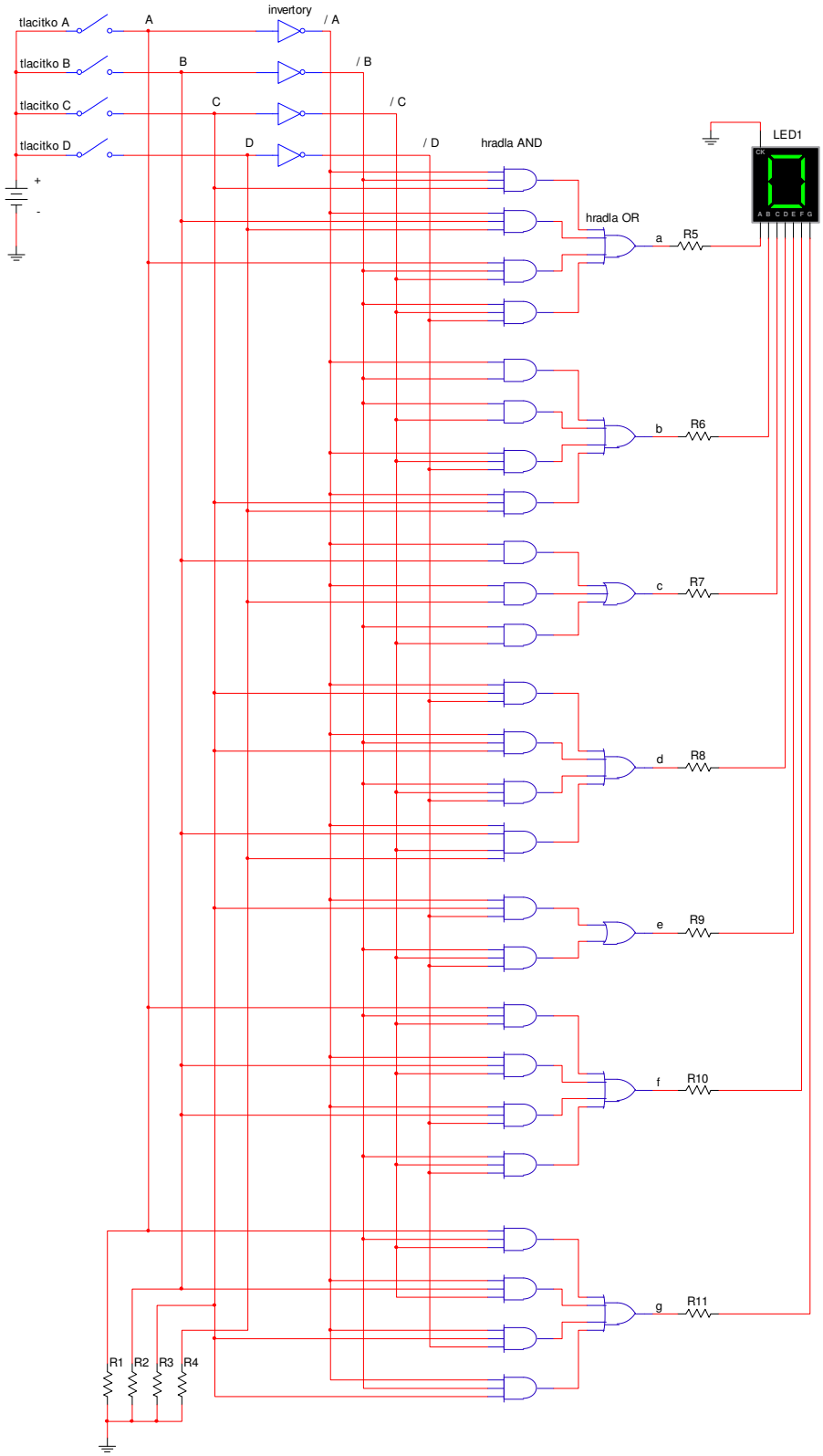
Prvním způsob realizace převodníku BCD na sedmisegmentový zobrazovač je uveden na obr. 9, kde byla zvolena hradla základních funkcí součinu AND, součtu OR a negace NOT. Schéma převodníku plnohodnotně odpovídá zápisu funkce po minimalizaci pro jednotlivé segmenty a až g sedmisegmentového zobrazovače. Jak je ze schématu zřejmé, byla použita hradla negace NOT, dále dvou, tří i čtyřvstupová hradla AND a dvou, tří i čtyřvstupová hradla OR. Jednotlivé vstupy A, B, C, D odpovídají vstupům jako v pravdivostní tabulce (tab. 23). Vstupní BCD kombinace převodníku se volí tlačítky A, B, C, D. V klidové poloze tlačítek (rozepnuto) je logická úroveň vstupů A, B, C, D definována pomocí rezistorů R1, R2, R3 a R4 na hodnotu log. 0 a současně vstupy $\overline{A}, \overline{B}, \overline{C}, \overline{D}$. Jsou nadefinovány na hodnotu log. 1. Tento stav je dál detekován jako 0 na sedmisegmentovém

zobrazovači. Pokud např. sepneme tlačítko A přivedeme na vstup A hodnotu log. 1, $\overline{A}$ se změní na hodnotu log. 0 a jako výsledek se na sedmisegmentovém zobrazovači ukáže číslo 8.
Nevýhodou tohoto zapojení je použití veliké rozmanitosti hradel (NOT, AND, OR) a také některé těžko dostupné funkce hradel, jako např. funkce čtyřvstupového hradla OR.

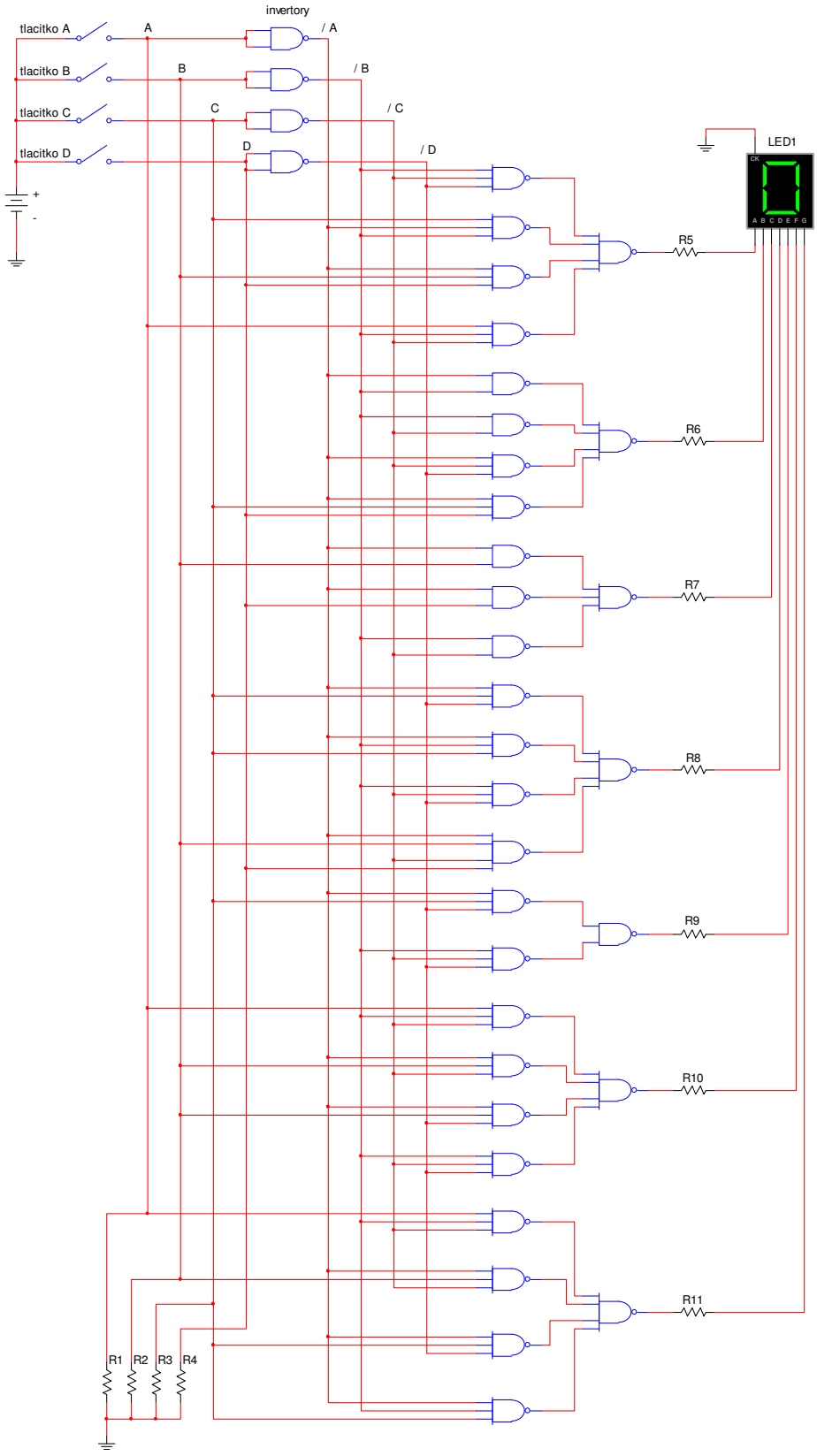
Za použití poznatků z Boolovy algebry byly jednotlivé funkce transformovány do formy, kde se vyskytuje jediný typ hradla a to typu NAND (obr. 10). Výhodou hradel NAND je podstatně větší využití (realizace všech funkcí NOT, AND, OR), dobrá dostupnost obvodů a jejich poměrně široká nabídka (existuje i 13ti vstupové hradlo NAND 74xx133).

Nevýhodou použití hradel obecně je často neúměrně vysoký počet integrovaných obvodů, zejména při návrhu složitějších funkcí, což s sebou přináší také zvýšení spotřeby výsledného zařízení. Další nevýhodou je poměrně komplikovaná změna funkce, která má za následek pracné přepojování částí obvodu.

[6]



Obr. 9: Schéma převodníku BCD na sedmisegmentový zobrazovač pomocí hradel AND a OR

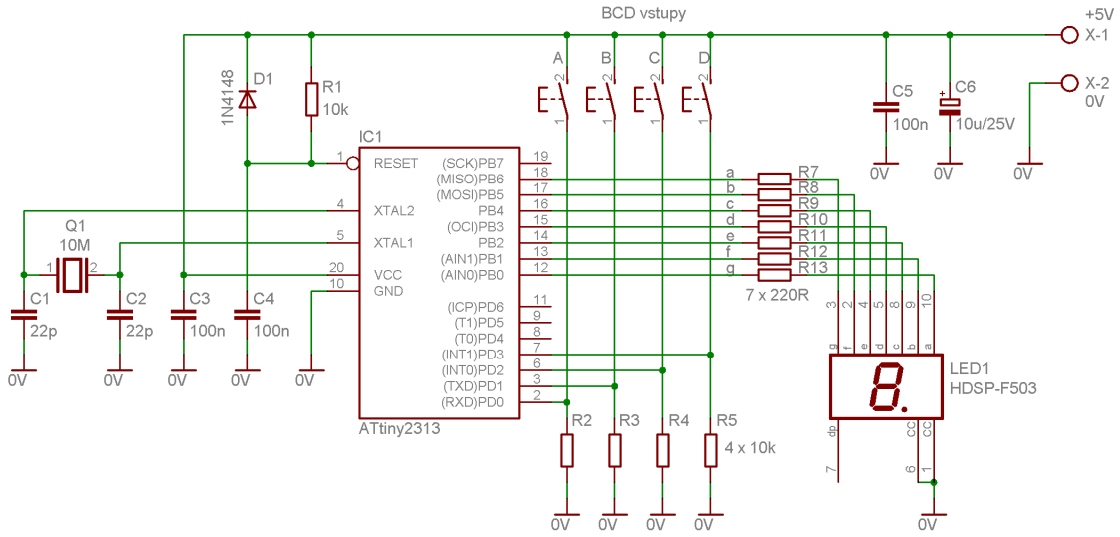


Obr. 10: Schéma převodníku BCD na sedmisegmentový zobrazovač pomocí hradel NAND

6.2 REALIZACE MIKROKONTOLÉREM

Pro ukázku realizace jednoduchého převodníku BCD na sedmisegmentový zobrazovač byl zvolen mikrokontrolér ATtiny 2313. Tento obvod je velice oblíbený zejména v amatérských konstrukcích z důvodu nízké ceny, velmi dobré vybavenosti interními periferiemi a slušným výpočetním výkonem (až 20 MIPS).

Mikrokontrolér je vlastně jednočipový mikropočítač, který na jednom čipu obsahuje veškeré komponenty nutné ke své autonomní činnosti. Dnešní mikrontroléry jsou běžně vybaveny kromě vlastního výpočetního jádra také datovou pamětí RAM, pamětí programu Flash, jednotkami styku s okolním světem (vstupně–výstupní brány). Dále jsou standardně k dispozici jednotky čítačů/časovačů, sériových kanálů a mnohé další integrované periferní jednotky.



Obr. 11: Schéma převodníku BCD na sedmisegmentový zobrazovač s mikrokontrolérem ATtiny 2313

Použitý mikrokontrolér ATtiny 2313 je základním členem rodiny s jádrem AVR firmy Atmel®. Toto jádro bylo navrženo pro vysoký výpočetní výkon s ohledem na programování vyššími programovacími jazyky jako je jazyk C. Tohoto úkolu bylo dosaženo především díky použití harvardské architektury, redukované instrukční sady (RISC) a rychle přístupného 32 bytového registrového pole.

Příklad obvodového řešení převodníku BCD na sedmisegmentový zobrazovač za použití mikrokontroléru ATtiny 2313 je uveden na následujícím obrázku. Mikrokontrolér IC1 je doplněn krystalovým rezonátorem Q1 ve funkci zdroje hodinového kmitočtu 10 MHz a dále jsou přidány blokovací kondenzátory C3, C5 a C6 pro vyhlazení napájecího napětí. Jako obvod resetu je aplikován integrační RC článek R1 (rezistor), C4 (kondenzátor) a vybíjecí D1 (dioda).

Vstupní BCD kombinace převodníku se volí tlačítky A, B, C a D. V klidové poloze tlačítek (rozepnuto) je logická úroveň vstupní brány PD obvodu IC1 definovaná pomocí rezistorů R2, R3, R4 a R5 na hodnotu log. 0. Vstupní BCD hodnota je dále načtena obslužným softwarem IC1 a dekódována v programové smyčce na příslušnou výstupní kombinaci, která je vystavena na výstupní bránu PB. Odtud jsou již přes předřadné rezistory R7 až R13 napájeny přímo segmenty displeje LED1.

Ukázka obslužného softwaru mikrokontroléru IC1 je realizována z ilustrativních důvodů dvojím způsobem. První program je napsán v jazyce symbolických adres assembleru. Druhé řešení je ukázáno ve vyšším programovacím jazyku C. Oba použité přístupy budou objektivně porovnány v závěru kapitoly.

[7][8][12][17]

6.2.1 Program napsaný v assembleru

Assembler, neboli jazyk symbolických adres, je nejnižším programovacím jazykem a procesor mu přímo rozumí. Assembler je vhodné použít při požadavku na optimalizaci rychlosti zpracování programu a nízkou velikost vlastního programového kódu. Tyto výhody jsou však vykoupeny větší náročností na tvorbu zdrojového kódu a programátor musí být dokonale seznámen s architekturou použitého procesoru.

Zdrojový kód lze psát v jakémkoli textovém editoru, který do textu nepřidává formátovací znaky (Microsoft Word®). Nejvýhodnější se však jeví použití AVR Studia®. AVR Studio® je freewareový program distribuovaný firmou ATMEL®, který je určený k tvorbě, simulaci a kompilaci zdrojových programů pro mikrokontroléry ATMEL.

Následující ukázka výpisu zdrojového kódu převodníku BCD na sedmisegmentový zobrazovač má pouze ilustrativní charakter pro vytvoření představy o způsobu práce v assembleru. Její detailní popis by silně přesahoval rámec této bakalářské práce, nicméně je dále uveden alespoň stručný popis uvedeného programu.

V úvodu programu je na řádku 1 definován použitý mikrokontrolér ATtiny 2313, na řádku 2 definována oblast paměti Flash a na řádku 3 vytvořeno symbolické jméno pro pracovní registr R16. Dále program pokračuje inicializací ukazatele zásobníku (řádek 6 + řádek 7) po restartu procesoru a definicí režimu vstupně–výstupních bran (řádek 9 až řádek 16). V další části běhu programu se dostáváme k hlavní programové smyčce “Main“.

```
1            .INCLUDE "tn2313def.inc" ; definice ATtiny 2313
2            .CSEG                    ; kodovy segment
3            .DEF reg = R16           ; pracovní registr
4
5    Reset:       ; reset MCU
6            ldi reg, RAMEND          ; ukazatel zasobniku
7            out SPL, reg
8                 ; PortB
9            clr reg                  ; PB0..7: log. 0
10           out PORTB, reg
11           ldi reg, 0xFF            ; PB0..7: vystupy
12           out DDRB, reg
13                ; PortD
14           out PORTD, reg           ; PD0..7: pull-up
15           clr reg                  ; PD0..7: vstupy
16           out DDRD, reg
17
18   Main:       ; hlavni smycka (1, 60us)
19               ; nacteni vstupniho BCD kodu
20           in reg, PIND             ; reg = PIND
21           andi reg, 0x0F           ; reg &= 0x0F
22               ; reg < 10 ???
23           cpi reg, 10              ; reg - 10
24           brlo Main1               ; reg <  10 ==> Main1
25           ldi reg, 10              ; reg >= 10 ==> reg = 10
26   Main1:      ; adresa prevodni tabulky
27           ldi ZL, LOW(Adr_Seg)     ; Z = Adr_Seg
28           ldi ZH, HIGH(Adr_Seg)
29               ; pricteni BCD kodu
30           add ZL, reg              ; Zl += reg
31           clr reg                  ; reg = 0
32           adc ZH, reg              ; Zh += C
33               ; nahrani segmentu z flash
34           lpm                      ; R0 = *Z
35           out PORTB, r0            ; vystup na port
36               ; zacykleni
37           rjmp Main
38
39               ; tabulka znaku
40   Seg:    .DB 0b01111110, 0b00110000    ; 0, 1
41           .DB 0b01101101, 0b01111001    ; 2, 3
42           .DB 0b00110011, 0b01011011    ; 4, 5
43           .DB 0b00011111, 0b01110000    ; 6, 7
44           .DB 0b01111111, 0b01110011    ; 8, 9
45           .DB 0b00000000                ; 10
46           .EQU Adr_Seg = 2 * Seg        ; adresa tabulky
47
```

Zde se v prvním kroku nahraje obsah vstupní brány PD (řádek 20) a vymaskuje dolní půlbyte operací logického součinu AND (řádek 21). Následuje podmínka přesahu vstupního rozsahu BCD čísla (řádek 23 a řádek 24). V případě, že je dekódované číslo v rozsahu 10 až 15, bude toto číslo bráno jako hodnota 10 (řádek 25). Tím máme v pracovním registru reg obsaženo vstupní BCD číslo včetně ošetření okrajové podmínky přetečení rozsahu. Nyní nahrajeme do interního 16bitového ukazatele Z počáteční adresu dekódovací tabulky (řádek 27 a řádek 28) pro budoucí nepřímou adresaci. Následuje přičtení pracovního registru reg k ukazateli Z (řádek 30 až řádek 32) včetně uvážení možnosti přenosu mezi byty. Touto operací jsme přesně určili paměťovou buňku dekódovací tabulky, která obsahuje námi hledaný byte odpovídající dekódované hodnotě. Nyní již stačí pouze vyčíst tento byte (řádek 34) a vystavit jej na výstupní bránu PB (řádek 35). Celý program je zacyklen skokem z řádku 37 zpět na návěští "Main" (řádek 18). Dekódovací tabulka je definována na konci programu (řádek 40 až řádek 46) v oblasti programové paměti Flash.

V rámci simulace uvedeného programu byla jednak ověřena jeho správná funkce a dále stanovena perioda zpracování programu na 1,6 μs při použitém taktovacím kmitočtu 10 MHz. Uvedený časový interval by se dal ještě zkrátit optimalizací uvedeného kódu, což by přineslo jisté znepřehlednění kódu, a tudíž bylo od tohoto záměru upuštěno.

[9][12][17]

6.2.2 Program napsaný v jazyku C

Jazyk C již patří do třídy vyšších programovacích jazyků. Tento jazyk již není pevně svázán s konkrétním hardwarem použitého mikrokontroléru a umožňuje vyšší míru přenositelnosti kódu. Z toho vyplývá, že programátor již nutně nemusí znát architekturu použitého procesoru do posledního detailu. To umožňuje poměrně jednoduchou realizaci cyklů, aritmetických operací i v plovoucí řádové čárce atd. Tím dochází k rapidnímu zjednodušení práce programátorů a i celkového času stráveného vývojem programu. Tyto nesporné výhody jsou však vykoupeny menší efektivitou výsledného kódu.

Zdrojový kód lze psát v jakémkoli textovém editoru, který do textu nepřidává formátovací znaky (Microsoft Word®). Nejvýhodnější je však také použití speciálního programu, který spojuje textový editor a kompilátor v jeden celek. Příkladem může být C kompilátor a editor IccAVR® od firmy Image Craft®, ve kterém byla vytvořena i následující ukázka zdrojového kódu převodníku BCD na sedmisegmentový zobrazovač.

Stejně jako v předcházející kapitole lze prohlásit, že ukázka zdrojového kódu dekodéru má opět pouze ilustrativní charakter. Její detailní popis silně přesahuje rámec této bakalářské práce, a proto je dále uveden pouze stručný popis uvedeného programu.

Na řádku 1 je překladač jazyka C seznámen s hardwarem použitého mikrokontroléru. Následuje definice dekódovací tabulky dekodéru (na řádku 3 až řádku 16). Stejně jako v minulé kapitole je tabulka umístěna v programové části paměti Flash. Vlastní program začíná funkcí “main“ (na řádku 19). V úvodu funkce jsou definovány režimy vstupně–výstupních bran (řádek 22 a řádek 23) a dále přechází program do nekonečné smyčky (řádek 25).

```
1   #include <io2313v.h>                // ATtiny 2313
2
3   unsigned char segment[] = {         // převodní tab. BCD -> 7seg.
4   //0bXabcdefg                        // segmenty (0 = off; 1 = on)
5     0b01111110, // 0
6     0b00110000, // 1
7     0b01101101, // 2
8     0b01111001, // 3
9     0b00110011, // 4
10    0b01011011, // 5
11    0b00011111, // 6
12    0b01110000, // 7
13    0b01111111, // 8
14    0b01110011, // 9
15    0b00000000  // 10
16  };
17
18
19  void main(void)
20  {
21    // inicializace PORT B a PORT D
22    PORTB = 0x00; DDRB = 0xFF;        // PB0..7: vystupy(log. 0)
23    PORTD = 0xFF; DDRD = 0x00;        // PD0..7: vstupy (pull-up)
24
25    for (;;)                          // nekonecna smycka (2,10 us)
26      PORTB = segment[((PIND & 0x0F) > 9) ? 10 : (PIND & 0x0F)];
27  }
```

V této smyčce dochází stejně jako v minulém příkladě k nahrání stavu vstupní brány PD a testu přetečení vstupního BCD čísla. V případě přetečení bude na výstupní bránu PB vystavena chybová hodnota 10 z dekódovací tabulky „segment“. V opačném případě bude z tabulky nahrána správná hodnota odpovídající dekódované hodnotě vstupního BCD čísla.

Simulací uvedeného programu byla stejně jako v předcházející kapitole ověřena správná funkce programu a dále stanovena perioda zpracování programu na 2,1 μs při použitém taktovacím kmitočtu 10 MHz. Delší změřená doba zpracování programu je právě ona daň za ulehčení, zjednodušení a zpřehlednění práce při vytváření nebo i pozdější editaci zdrojového kódu programu.

[10][18]

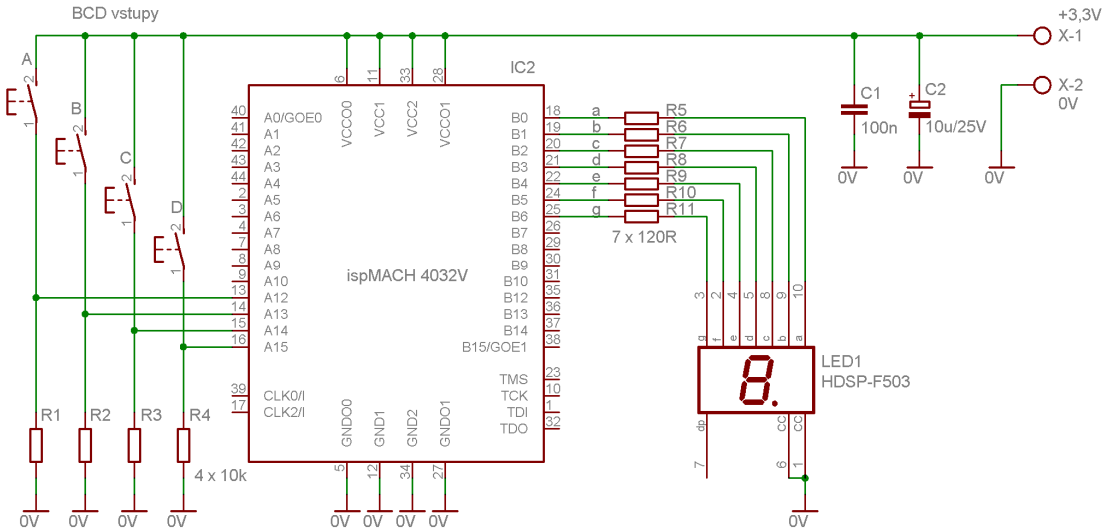
6.3 REALIZACE PROGRAMOVATELNÝM HRADLOVÝM POLEM

Programovatelná hradlová pole (PLD – Programmable Logic Devices) jsou elektronické obvodové prvky, které jsou určeny k realizaci uživatelských kombinačních a sekvenčních funkcí. Základním principem těchto obvodů je matice logických členů OR nebo AND jejichž propojením si uživatel volí požadovanou funkci. V současné době můžeme PLD obvody rozdělit do dvou základních skupin: CPLD (Complex PLX) a FPGA (Field Programmable Gate Arrays). CPLD obvody se skládají z několika vzájemně propojených logických polí tzv. makro-buněk a zpravidla se využívají k realizaci složitějších sekvenčních funkcí. Pro představu lze s těmito obvody realizovat logické obvody adekvátní tisícům logických hradel. Obvody FPGA jsou složeny z velkého množství malých buněk programovatelných polí. Touto strukturou je možno realizovat i nejsložitější sekvenční obvody jako je třeba jádro mikrokontoléru.

Pro praktickou ukázku realizace převodníku BCD na sedmisegmentový zobrazovač programovatelným hradlovým polem byl zvolen obvod ispMACH 4032V firmy Lattice Semiconuctor®. Tento obvod patří mezi základní představitele CPLD polí s vyváženým poměrem hustoty integrace, výkonu a ceny.

Možné schéma zapojení dekodéru je uvedeno na obr. 12. Z něj je jasně patrné, že veškerá logika dekodéru je zapouzdřena pouze v jediném integrovaném obvodu IC1, který je doplněn blokovacími kondenzátory C1 a C3 pro vyhlazení napájecího napětí +3,3 V. Vstupní BCD kombinace se zadávají tlačítky A, B, C a D stejným způsobem jako v předcházející kapitole. Jejich stav je dekódován logickými funkcemi převzatými

z kapitoly 6.1. a uloženými v obvodu IC1. Výstup dekodéru je pak zaveden přes předřadné rezistory R5 až R11 na příslušné segmenty displeje LED1.



Obr. 12: Schéma převodníku BCD na sedmisegmentový zobrazovač s PLD polem ispMACH 4032V

Vzhledem k již poměrně složitým vnitřním strukturám PLD obvodů poskytují sami výrobci specializovaná návrhová prostředí pro vytvoření požadovaných uživatelských funkcí. Výsledný návrh se následně nahraje do obvodu, který poté začne plnit svou funkci. Návrh funkcí je možné vytvořit dvojím způsobem. První metoda spočívá v nakreslení logického schématu v grafickém editoru, který je součástí vývojového prostředí. Tento přístup je poměrně dobře propracovaný a zpravidla nabízí rozsáhlé knihovny předpřipravených obvodů (logická hradla, čítače, násobičky, …). Díky tomu nemusí uživatel znát detailně vnitřní strukturu použitého obvodu a je schopen rychlého intuitivního návrhu. Tento přístup je však vhodný pouze do určité složitosti navrhovaného sytému a většinou se využívá pouze pro CPLD obvody. Druhý přístup je popis funkcí v HDL (Hardware Description Language) jazyku. Tímto postupem lze realizovat i velmi složité aplikace založené na FPGA polích. Nevýhodou však je větší složitost návrhu a dobrá znalost konkrétně použitého obvodu.

Návrh převodníku BCD na sedmisegmentový zobrazovač byl realizován schématickou formou ve vývojovém prostředí ispLEVER Classic® firmy Lattice. Logické schéma bylo převzato z kapitoly 6.1. a po překladu nahráno do obvodu ispMACH 4032V. Tím došlo k rapidnímu snížení počtu obvodů nutných k fyzické realizaci dekodéru. Z časových simulací bylo dále stanoveno maximální časové zpoždění dekodéru na cca 8 ns, což lze považovat za velice slušný výsledek.

[11][13][19]

6.4 POROVNÁNÍ JEDNOTLIVÝCH ŘEŠENÍ

V této kapitole byly nastíněny tři možné přístupy k realizaci převodníku BCD na sedmisegmentový zobrazovač, přičemž použití mikrokontroléru a logického programovatelného pole bylo pojato pouze informativní formou. Nyní se pokusíme objektivně porovnat jednotlivé výhody a nevýhody uvedených řešení.

Použití logických hradel

Toto řešení je koncepčně nejstarší metodou a dnes se ho využívá jen v omezené míře. Mezi výhody tohoto přístupu nesporně patří rychlá reakce jednotlivých hradel řádově v jednotkách ns v závislosti na použité technologii obvodu. Dále by se dala uvést dobrá dostupnost obvodů a jejich poměrně široká nabídka. Nevýhodou této metody je neúměrně vysoký počet integrovaných obvodů při návrhu složitějších funkcí, což s sebou přináší i zvětšení velikosti a spotřeby výsledného zařízení. Použitý počet hradel k realizaci dekodéru v kapitole 6.1. v porovnání s jediným obvodem v kapitole 6.3. hovoří jistě sám za sebe. Další nevýhodou je poměrně komplikovaná změna funkce, která s sebou přináší pracné přepojování částí obvodů.

Použití mikrokontroléru

Díky řízení činností mikrokontroléru programem, dosahují tyto prvky značné univerzality a jsou schopny řídit i velice složité procesy. Nespornou výhodou je velmi jednoduchá změna požadované funkce změnou ovládacího programu mikrokontroléru.

Nevýhodou je však konečná doba zpracování programu. V kapitole 6.2. bylo dosaženo zpoždění dekodéru 1,6 μs při programování assemblerem a 2,1 μs při použití jazyka C. V porovnání se zpožděním 8 ns za použití PLD obvodu v kapitole 6.3. jsou tyto časy neúměrně vysoké.

Použití programovatelných hradlových polí

Hradlová pole se snaží kompenzovat nevýhody obou předchozích řešení. Z logických hradel přejímají jejich rychlost reakce, která je v zásadě srovnatelná s rychlostí reakce logických obvodů. Současně odstraňují jejich nevýhodu v neúměrném narůstání jejich počtu v případě složitějších obvodů. Od mikrokontrolérů pochytily jednoduchou možnost změny požadované funkce změnou programu. Nevýhodou však zůstává, že PLD obvody nedosahují takové univerzality jako mikrokontroléry, která plyne zejména z jejich masového používání.

Závěr

Z uvedeného shrnutí by se pro skutečnou realizaci převodníku BCD na sedmisegmentový zobrazovač dala asi jednoznačně doporučit varianta s použitím PLD obvodu. Touto volbou minimalizujeme počet obvodových prvků v reálném zapojení při zachování výhody malého časového zpoždění převodníku. Navíc budeme mít výhodu snadné změny funkce obvodu v případě pozdějších úprav.

Při reálných návrzích logických obvodů není vždy volba řešení jednoznačně daná. Někdy je nutné dokonce jednotlivé varianty kombinovat a vytvořit tak celek odpovídající alespoň zdánlivě požadované funkci. Dále je nutné uvažovat ekonomický aspekt celého návrhu jak z pohledu ceny materiálu, tak i doby strávené samotným návrhem požadovaného obvodu. V neposlední řadě dochází občas také k poměrně bouřlivé obměně součástkové základny, zejména pak v procesorové technice. Při zohlednění všech těchto aspektů bývá většinou výsledný návrh kompromisem.

Seznam použité literatury

[1] PEŠKOVÁ, E., MULAČOVÁ, J. a kol.: Matematika – Přehled středoškolského učiva II. soubor, Praha, ORFEUS, 1992.

[2] KOVÁŘ, J., HANULÍK, S.: Číslicová technika. Učební texty SPŠ Zlín.

[3] TLUSTÝ, P.: Obecná algebra pro učitele, 2006

[4] PECINA, J., PECINA, P.: Základy číslicové techniky, 2007.

[5] PERRIN, J. P., DENOUETTE, M., DACLIN, E.: Logické systémy 1 a 2, Praha, SNTL, 1972.

[6] JEDLIČKA, P.: Přehled obvodů řady TTL 7400 (díl I. a díl II.), Praha, BEN, 2002.

[7] PINKER, J.: Mikroprocesory a mikropočítače. Praha, BEN, 2004.

[8] MATOUŠEK, D.: Práce s mikrokontroléry ATMEL AVR. Praha, BEN, 2003.

[9] BURKHARD, M.: C pro mikrokontroléry. Praha, BEN, 2003.

[10] HEROUT, P.: Učebnice jazyka C. České Budějovice, Kopp, 2001.

[11] KOLOUCH, J.: Programovatelné logické obvody. Elektronické skriptum. Brno: UREL FEKT VUT, 2007.

[12] Data sheet: ATtiny 2313

[13] Data sheet: ispMACH 4000V/B/C Family

[14] http://www.wikipedia.cz

[15] http://www.sosboh.cz/projekty/06_06_13/03/03.html

[16] http://sipal.fvtm.ujep.cz/Auto/Auto_02n.pdf

[17] http://www.atmel.com

[18] http://www.imagecraft.com

[19] http://www.latticesemi.com

(všechny odkazy platné k 12/2009)