Obsah

Kapitola 1
Úvod

Číslicová technika tvoří nezbytný základ pro automatizaci a procesorovou techniku. Jedná se o část elektroniky, která je tvořena logickými hradly, které jsou tvořeny polovodičovými součástkami a umoňují snadno skládat jednoduché logické součásti do komplexních celků, které provádějí nějakou předem definovanou činnost.

Číslicová technika se z velké části nezabívá elekterickými obovody, ale vytváří abstraktní vrstvu s metodamy návrhu složitých číslicových struktur. Tyto metody umožňují minimalizovat výsledný obvod a optimalizovat tak jeho spotřebu, spolehlivost, cenu, ...

Kapitola 2
Kódování dat

Kódování je proces při kterém se každému znaku nebo posloupnosti znaků (vzorů) jednoznačně přiřadí znak nebo posloupnost znaků (obrazů) z jiného souboru znaků → množinové zobrazení. Kódování je tedy transformace informace z jedné formy do jiné pomocí určitého postupu - algoritmu. Převodu informace do číselné podoby pomocí počítače se říká digitalizace.

Kombinaci bitů zobrazující znak se říká **kódové slovo**. Z ekonomických důvodů se znaky zobrazují pokud možno co nejmenším množstvím bitů. Jedna sekvence bitů může mít mnoho různých významů, například může reprezentovat znak, číslo, operaci, ...

Kódová slova pro různé kódy se řadí do **kódových tabulek**. Přiřazování kódových slov určitým znakům se říká **kódování**. Původní znak se získá z kódového slova **dekódováním**.

2.1 Kódy používané pro strojové operace

Binární data je nutné pro účelí zpracování v číslicových obvodech upravit, aby jejich zpracování co nejoptimálnější.

2.1.1 Přímí dvojkový kód

Je to kód, který jednoznačně vyjadřuje číslici pomocí dvou různých znaků (1, 0). Číslo v přímém dvojkovém kódu je vyjádřeno v polynomálním stavu způsobem:

$$N = \sum_{i=0}^{t=k-1} (n_i \cdot R^i)$$

Jedná se o nejpoužívanější způsob kódování celých binárních čísel.

2.2 Kódy pro zkrácení zápisu binárních čísel

K přehlednějšímu zobrazení dat ve dvojkové soustavě se používá zobrazení v osmičkové a šestnáctkové soustavě. Kromě toho se dále používají binární kódy, které umožňují zápis binárních hodnot výrazně zpřehlednit.

2.2.1 BCD kód

Jedná se o dvojkově desítkový kód (Binary Code Decimal). Tento kód se používá ke kódování desítkových číslic 0 až 9. V tomto kódu je každá desítková číslice D daného čísla vyjádřena kódovým slovem se čtyřmi bity ve dvojkové soustavě.

$$\begin{bmatrix} D & b_3 & b_2 & b_1 & b_0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 & 1 \\ 4 & 0 & 1 & 0 & 0 \\ 5 & 0 & 1 & 0 & 1 \\ 6 & 0 & 1 & 1 & 0 \\ 7 & 0 & 1 & 1 & 1 \\ 8 & 1 & 0 & 0 & 0 \\ 9 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Pro zápis větších čísel se používá kombinace jednotlivých BCD čísel.

$$579_{(10)} = 0101\ 0111\ 1001_{(BCD)}$$

$$5 \rightarrow 0101$$

$$7 \rightarrow 0111$$

$$9 \rightarrow 1001$$

Znaménko mínus se zobrazuje takovou kombinací bitů, která se v BCD kódu nevyskytuje. Znaménko plus se zobrazuje jako 1010 a znaménko mínus jako 1011.

2.2.2 Expres 3 kód (BCD + 3)

U kódu BCD vznikne problém v případě, kdy je třeba vyjádřit prázdnou informaci (null). V takovém případě ukazuje počítač hodnotu 0000, ale u BCD kódu je takto zaznamenána číslice nula. Kvůli tomu vznikl **kód BCD+3**. Tento kód byl využíván v některých počítačích pro snazší realizaci dekadických operací.

Kód BCD+3 vznikne tak, že se ke každé číslici v BCD přičte hodnota 3:

D	BCD	$BCD+3$
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

2.2.3 Grayův kód

Grayův kód má tu vlastnost, že při přechodu od jednoho kódového slova ke druhému se mění vždy jen hodnota jednoho bitu. To je patrné zejména při změně více řádů př: $(0111)_2 \rightarrow (1000)_2$. Jestliže se tedy v Grayově kódu zobrazí posloupnost čísel, potom při přechodu mezi dvěma kódovými slovy může nastat nepřesností chyba rovnající se nejvýše jedničce.

Při realizaci logických obvodů je pak možné se vyhnout chybám vznikajících při změně více bitů. Takovým nežádoucím stavům se říká **hazardy**.

Grayův kód se odvozuje z binárního čísla D jako součet nonekvivalence dvou vedle sebe ležících řádových míst.

D	BCD	$Gray$
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

$$g_n = b_{n+1} \oplus b_n$$

Příklad převodu binárního čísla do grayova kódu

$$26_{10} = b_4 b_3 b_2 b_1 b_0 = 11010_2 = g_4 g_3 g_2 g_1 g_0 = 10111_{gray}$$

$$g_4 = b_4 = 1$$

$$g_3 = b_4 \oplus b_3 = 1 \oplus 1 = 0$$

$$g_2 = b_3 \oplus b_2 = 1 \oplus 0 = 1$$

$$g_1 = b_2 \oplus b_1 = 0 \oplus 1 = 1$$

$$g_0 = b_1 \oplus b_0 = 1 \oplus 0 = 1;$$

Kapitola 3
Logická aritmetika

Mnoho logických obvodů je založených na principu vykonávání aritmetických operací s čísly. Typickým příkladem je aritmeticko-logická jednotka procesoru. Základní aritmetické operace jsou sčítání, odčítání, násobení a dělení. Obecně platí, že tyto aritmetické operace se vykonávají ve všech číselných soustavách stejně, pouze musí být vzat v úvahu základ dané soustavy a jeho přenos do vyšších řádů. V číslicových obvodech jsou ale nejpoužívanější číselné soustavy dvojková osmičková a šestnáctková.

3.1 Aritmetická operace součet

Sčítání v soustavách dvojkové, osmičkové a šestnáctkové probíhá podobně jako v soustavě desítkové. Jediný rozdíl je v přenosech do vyšších řádů.

Přenos do vyššího řádu nastane v případě, že je součet sčítaných číslic roven nebo větší než základ číselné soustavy, ve které se nacházejí daná čísla. Obecně lze součet čísel A a B zapsat jako:

$$S = A + B$$

Nebo také v souladu s polynomální rovnicí definující libovolné číslo v libovolné číselné soustavě:

$$\sum_{i=0}^{n} s_i \cdot R^i = \sum_{i=0}^{n-1} a_i \cdot R^i + \sum_{i=0}^{n-1} b_i \cdot R^i$$

kde n je počet číslic tvořící dané číslo.

V každém číselném řádu se provádějí následující o perace:

$$a_i \cdot R^i + b_i \cdot R^i + c_i \cdot R^i = s_i \cdot R^i + c_{i+1} \cdot R^{i+1}$$

Koeficient c_i představuje přenos z nižšího řádu do stávajícího řádu (pokud se jedná o první číslici v pořadí pak je $c_0 = 0$) a c_{i+1} ze stávajícího řádu do vyššího. Při sčítání v binární soustavě se využívají vlastnosti binárních čísel:

$$0 + 0 = 0 \rightarrow c = 0$$

$$0 + 1 = 1 \rightarrow c = 0$$

$$1 + 0 = 1 \rightarrow c = 0$$

$$0 + 0 = 0 \rightarrow c = 1$$

Pro součet v desítkové soustavě platí $1 + 1 = 2$, ale ve dvojkové soustavě není číslice 2 definována, proto dojde k přenosu do vyššího řádu a platí:

$$1 + 1 = 10$$

Příklad součtu dvou bnárních čísel:

$$11010 + 110 = 100000$$

3.2 Aritmetická operace rozdíl

Rozdíl dvou čísel je inverzní operace k operaci součet. Pro operaci rozdíl tedy platí podobná pravidla jako pro operaci součet a to ve všech číselných soustavách. Vztahy pro operaci rozdíl v jednom číselném řádu lze vyjádřit pomocí rovnice:

$$a_i \cdot R^i - b_i \cdot R^i - c_i \cdot R^i = r_i \cdot R^i - c_{i+1} \cdot R^{i+1}$$

kde a_i a b_i představují jednotlivé číslice odečítaných čísel na *i-tém* řádovém místě a r_i jejich rozdíl. Při odčítání v binární soustavě se využívá vlastnosti binárních čísel:

$$0 - 0 = 0 \rightarrow c = 0$$

$$0 - 1 = 1 \rightarrow c = -1$$

$$1 - 0 = 1 \rightarrow c = 0$$

$$1 - 1 = 0 \rightarrow c = 0$$

Příklad rozdílu dvou binárních čísel:

$$11010 - 110 = 10100$$

Tímto způsobem se ale většinou v číslicových obvodech neodečítá. Místo toho se odečítá číslo vyjádřené ve tvaru dvojkového doplňku, které reprezentuje zápornou hodnotu binárního čísla, následně se obě čísla jednoduše sečtou.

3.3 Vyjádření záporných čísel v binární soustavě

Pomocí N bitů lze vyjádřit 2^N jedinečných kladných binárních číselných hodnot. Při vyjadřování záporných čísel je nutné tento rozsah rozdělit mezi hodnoty kladné a hodnoty záporné. Proto pomocí N bitů lze vyjádřit 2^{N-1} jedinečných kladných a přibližně stejné množství záporných binárních hodnot (záporných hodnot je o jednu méně než kladných kvůli neexistenci záporné nuly).

Existují tři způsoby vyjádření záporného binárního čísla:

- Pomocí znaménka a absolutní hodnoty binárního čísla
- Pomocí jednotkového doplňku
- Pomocí dvojkového doplňku

Zobrazení kladných hodnot je u všech tří způsobu reprezentací záporných čísel vždy stejná.

3.3.1 Znaménko a absolutní hodnota binárního čísla

Při vyjádření binárního čísla se používá nejvyšší bit (MSB) jako znaménkový. Kladné číslo má znaménkový bit 0 a záporné 1. Za znaménkovým bitem následuje absolutní hodnota čísla.

Příkladem může být číslo $5_{(10)} = 101_{(2)}$. Pro výjádření čísla 5, jsou potřeba 3 bity. Přiřazením znaménkového bitu na nejvyšší řád čísla vznikne 4-bitové znaménkové číslo:

$$+5_{(10)} \rightarrow 0101_{(2)}$$
$$-5_{(10)} \rightarrow 1101_{(2)}$$

V tomto způsobu vyjádření záporných čísel je uvažuje

pouze kladná nula $\rightarrow 0_{(10)} = 0000_{(2)}$. Hodnota záporné nuly se zakazuje.

3.3.2 Jednotkový doplňek - F_{1K}

Jedničkový doplněk (též inverzní kód) binárního čísla je způsob reprezentace čísel se znaménkem, u něhož se záporná hodnota získá znegováním jednotlivých bitů v binární reprezentaci čísla (nahrazením nul jedničkami a naopak). Zobrazované číslo F lze vyjádřit (popsat) pomocí jednotkového doplňku pomocí vztahu:

$$F_{1K} = (2^N - 1) - F$$

kde F představuje zobrazované číslo o počtu bitů N a číslo F_{1K} je číslo F převedené do jednotkového doplňku.

Číslo vyjádřené výrazem $2^N - 1$ představuje jedničky ve všech řádech čísla. Vyjádření čísla 5_{10} v doplňkovém kódu:

$$F_{1K} = 1111_{(2)} - 0101_{(2)} = 1010_{(2)} = -5_{10}$$

Záporná nula je v tomto zobrazení opět zakázána - $0000_{(2)} \rightarrow 1111_{(2)}$

3.3.3 Dvojkový doplněk - F_{2K}

Dvojkový doplněk je nejčastěji používané zobrazení (kódování) záporných čísel v binární číselné soustavě. Jeho výhodou je efektivní provádění aritmetických operací. Záporné číslo ve dvojkovém doplňku je tvořeno jako doplněk čísla F do čísla 2^N, kde N je počet bitů binárního čísla F. Dvojkový doplněk čísla F lze matematicky vyjádřit pomocí vztahu:

$$F_{2K} = 2^N - F$$

Dvojkový doplněk se nejsnáze určí tak, že se všechny bit čísla F znegují a k výsledku se přičte hodnota 1.

$$F_{2K} = 0101_2 = 1010_2 + 1_2 = 1011_2 = -5$$

Dvojkový doplněk umožňuje zjednodušit konstrukci aritmeticko – logické jednotky uvnitř procesoru díky tomu, že sčítání a odečítání lze provádět pro čísla se znaménkem stejně jako pro čísla bez znaménka, odlišná je pouze interpretace přetečení (OV - overflow).

V případě dvojkového doplňku neexistuje záporná nula. Kvůli tomu je interval zobrazovaných kladných a záporných čísel nesymetrický:

$$(-128; 127)$$

Pro dvojkový doplněk platí, že pokud je funkce pro převod binárního čísla do dvojkového doplňku použita dvakrát po sobě, je opět získána původní hodnota $\rightarrow (-)(-) = +$.

3.4 Aritmetická operace součin

Násobení v soustavách o základu R je možné realizovat dvěma různými způsoby. Jedním z nich je převedení na opakované sčítání (sekvenční metoda) a druhý je klasické násobení analogické násobení v desítkové soustavě (kombinační metoda).

V případě převodu násobení na opakované sčítání je násobenec postupně sčítán tolikrát, kolikrát to předepisuje hodnota příslušného řádového koeficientu násobitele:

$$A_{(R)} \cdot B_{(R)} = \sum_{i=0}^{n=A} B_i = \underbrace{B + B + ... + B}_{A}$$

V případě postupného roznásobování každého číselného řádu jednoho čísla každým číselným řádem druhého čísla:

$$A_{(R)} \cdot B_{(R)} = (a_{n-1} \cdot R^{n-1} + ... + a_0 \cdot R^0) \cdot (b_{m-1} \cdot R^{m-1} + ... + b_0 \cdot R^0)$$

$= (a_{n-1} \cdot b_{m-1} \cdot R^{n-1+m-1}) + (a_{n-1} \cdot b_{m-2} \cdot R^{n-1+m-2}) + ... + (a_0 \cdot b_0 \cdot R^0)$

3.5 Aritmetická opeace podíl

3.5.1 Reprezentace desetinných čísel

Libovolné desetinné číslo lze vyjádřit v libovolné číselné soustavě o základu R zápisem:

$$A = \underbrace{a_{n-1} \cdot R^{n-1} + ... + a_0 \cdot R^0}_{Celá\ část}, \underbrace{a_{-1} \cdot R^{-1} + ... + a_{m-1} \cdot R^{m-1}}_{Desetinná\ část}$$

kde R je základ číselné soustavy, $a_{n-1}...a_{m-1} \in \{0, 1, ...R-1\}$, n je počet číslic celé části čísla a m je počet číslic desetinné části čísla.

Protože kladný exponent mocniny vyjadřuje číselnou hodnotu $x > 0$ a záporný exponent mocniny vyjadřuje číselnou hodnotu $x < 0$, polynomický zápis čísla se zápornými exponenty základu číselné soustavy tvoří desetinnou část čísla.

3.5.2 Převod desetinných čísel mezi číselnými soustavami

Jedno z možností převodu desetinných čísel z desítkové číselné soustavy do libovolné jiné číselné sustavy je pomocí postupného odčítání mocnin základu dané číselné soustavy.

Převod zlomku na desetinné číslo v dané číselné soustavě může zkončit třemi různými způsoby:

- desetinné číslo bude mít v dané číselné soustavě ukončený desetinný rozvoj
- desetinné číslo bude mít v dané číselné soustavě neukončený periodický rozvoj
- desetinné číslo bude mít v dané číselné soustavě neukončený neperiodický rozvoj

Důležitým faktem je, že desetinný rozvoj čísla vyjádřeného například v desítkové soustavě a desetinný rozvoj téhož čísla například ve dvojkové soustavě může být rozdílný. Některé zlomky mohou mít v některých číselných soustavách neukončený periodický desetinný rozvoj v jiných neukončený neperiodický desetinný rozvoj a v dalších zase mohou mít ukončený desetinný rozvoj. To závisí na tom, zda je hodnota desetinné části určitým násobkem základu dané soustavy. Hodnota desetinného čísla se v různých číselných vyjádřeních nezmění (v závislosti na přesnosti vyjádření, počtu platných desetinných míst

čísla), ale jeho vyjádření může mít různý tvar.

Příkladem může být číslo $0,35$, které má v trojkové soustavě neukončený periodický desetinný rozvoj:

$0,35_{(10)} = 0,\overline{1001}_{(3)}$

3.6 Shrnutí

Kapitola 4
Logická algebra

Logika je nauka o základech myšlení v procesu vytváření úsudku a důkazu. Logický obvod je takový obvod, u něhož může každá veličina na vstupu i výstupu v ustáleném stavu s určenou přesností nabývat jen jedné ze dvou možných hodnot a který obsahuje takové prvky, jejichž vstupní a výstupní veličiny mohou nabývat také jen jedné ze dvou možných hodnot.

Logický obvod je realizován skupinou logických členů vzájemně spojených tak, aby realizovaly požadované logické funkce. Podle druhu realizované logické funkce se logické obvody dělí na:

• **Kombinační logické obvody** - jedná se o systémy, jejichž odezva je v určitém časovém okamžiku podmíněna výhradně hodnotami, které panují na vstupech tohoto systému (podle toho jaká kombinace logických hodnot je na vstupu je určitá kombinace logických hodnot na výstupu). Kombinační logické obvody lze popsat konečným množstvím rovnic tvaru:

$$\left[\begin{array}{c} y_0 = f_0(x_1, x_2, ...x_n) \\ y_1 = f_1(x_1, x_2, ...x_n) \\ ... \\ y_m = f_m(x_1, x_2, ...x_n) \end{array}\right].$$

Kde y_1, y_2, ..., y_m značí výstupní bitové proměnné logického obvodu a x_1, x_2, ..., x_n značí výstupní bitové proměnné.

• **Sekvenční logické obvody** - jedná se o systémy, jejichž odezva je v určitém časovém okamžiku dána ne jen hodnotami bitových proměnných na vstupech tohoto systému, ale i posloupností (sekvencí) předcházejících vstupních hodnot. Sekvenční obvod je proto opatřen pamětí, která svým stavem definuje vnitřní stav tohoto. Formálně lze sekvenční logický obvod popsat dvěma soustavami rovnic, jednou pro vstup a jednou pro vnitřní stavy (u kombinačních logických obvodů jsou vstupní hodnoty závislé ne jen na kombinaci vstupních hodnot, ale také na vnitřním stavu obvodu).

U kombinačního i sekvenčního logického obvodu lze v postupu jejich návrhu rozlišovat dva základní kroky:

• Přesný popis chování systému logickými funkcemi případně jejich zjednodušení (minimalizace).

• Obvodová realizace logického systému, která splňuje

chování dané logickými funkcemi.

4.1 Logická proměnná

Logická proměnná je obecně proměnná, která může nabývat nějakého konečného počtu logických hodnot. V případě logických obvodů jsou to zpravidla hodnoty 1 a 0 definující stav zapnuto a vypnuto popřípadě pravda a nepravda. Obecně se ale může jednat o jakékoli jiné dva související stavy.

V logických systémech, kde se vyskytuje více logických proměnných vedle sebe je vhodné z důvodů přehlednosti jednotlivé logické proměnné pojmenovat. Logické proměnné se dělí na vstupní a výstupní. Vstupní proměnné jsou většinou značeny alfabetickými znaky a, b, c,..., nebo jedním písmenem s indexem, které identifikuje daný druh vstupních proměnných (adresové, datové, ...) x_0, x_2, ..., nebo kombinace obojí. Výstupní proměnné jsou zpravidla značeny písmenem y_n, kde n je index proměnné. Výstupní proměnné mohou být pojmenovány libovolným způsobem, ale jejich název nesmí korespondovat s názvem vstupních proměnných.

4.2 Logická funkce

Logická funkce je taková funkce, která pro konečný počet vstupních parametrů vrací logické hodnoty. Parame-

try logických funkcí jsou opět logické proměnné (hodnoty). Logická funkce je tvořena **logickým výrazem**, který udává matematicko-logické vztahy mezi jednotlivými logickými proměnnými.

Úkolem kombinačních logických obvodů je realizovat funkce f_i tak, že každé kombinaci hodnot vstupních proměnných přiřadí určitou hodnotu výstupní proměnné. Popisem funkcí logických obvodů se zabývá logická algebra. Jedná se o algebraickou strukturu zobecňující vlastnosti množinových a logických operací.

Na vstupu systému lze z vnějšku dosadit až 2^n různých binárních kombinací, kde n je počet vstupních bitových proměnných. Pro n vstupních bitových proměnných lze definovat až 2^{n^n} různých funkcí f_i. To znamená, že pro n vstupních proměnných je možné definovat 2^{n^n} různých logických funkcí aniž by se definiční obor dvou libovolných funkcí shodoval. Všem kombinacím vstupních proměnných se říká **definiční obor funkce** a výsledným hodnotám na základě kombinací vstupních proměnných se říká **obor hodnot funkce**. Vstup dané logické funkce tedy tvoří uspořádanou n-tici logických hodnot.

4.3 Logický signál

Logický, dvoustavový, nebo binární signál je fyzikální veličina, která slouží jako nosič logické proměnné. Typ-

ickým příkladem logického signálu je elektrický proud, který vodičem buď protéká, nebo jím neprotéká. Logický signál sám o sobě nemusí být pouze dvoustavový, ale binární data jsou do něj vhodným způsobem zakódována (modulace).

4.4 Logický vektor

Logický vektor označuje uspořádanou n-tici logických proměnných, kde n je počet proměnných. V zápisu tedy záleží na jejich pořadí. Nejčastěji se logický vektor využívá v souvislosti se **vstupním vektorem** - I = $\{x_0, x_1, ...x_n\}$ a **výstupním vektorem** O = $\{y_0, y_1, ...y_n\}$, které označují seznam vstupních a výstupních logických proměnných.

Hodnota logického vektoru vyjadřuje pravdivostní ohodnocení jednotlivých logických proměnných vektoru, které jsou převedeny na číselnou hodnotu. Platí, že hodnota binárního vektoru se může pohybovat pouze v intervalu $0 - 2^n$, kde n je počet logických proměnných vektoru a hodnota 2^n udává počet kombinací dvou stavů n logických proměnných.

$$V = \{x_0, x_1, x_2, x_3, x_4\} = 00110 = 6$$

4.4.1 Logické funkce jedné proměnné

Logickou funkci jedné nezávislé proměnné lze vyjádřit čtyřmi způsoby.

$$\begin{bmatrix} a & F_0 & F_1 & F_2 & F_3 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

- F_0 - **triviální identita**, nulová funkce
- F_1 - **identická funkce**, její hodnota se shoduje s hodnotou proměnné a
- F_2 - **funkce negace**, hodnota funkce je opačná oproti hodnotě proměnné a
- F_3 - **triviální identita**, jednotková funkce

Praktický význam mají pouze funkce **identická** a **negace**. Negace proměnné se označuje pruhem nad příslušnou veličinou: $\overline{a}$

4.5 Logické funkce dvou proměnných

Pro dvě vstupní proměnné lze nalézt šestnáct navzájem různých logických funkcí.

$$\begin{bmatrix} a & 0 & 0 & 1 & 1 \\ b & 0 & 1 & 0 & 1 \\ F_0 & 0 & 0 & 0 & 0 \\ F_1 & 0 & 0 & 0 & 1 \\ F_2 & 0 & 0 & 1 & 0 \\ F_3 & 0 & 0 & 1 & 1 \\ F_4 & 0 & 1 & 0 & 0 \\ F_5 & 0 & 1 & 0 & 1 \\ F_6 & 0 & 1 & 1 & 0 \\ F_7 & 0 & 1 & 1 & 1 \\ F_8 & 1 & 0 & 0 & 0 \\ F_9 & 1 & 0 & 0 & 1 \\ F_{10} & 1 & 0 & 1 & 0 \\ F_{11} & 1 & 0 & 1 & 1 \\ F_{12} & 1 & 1 & 0 & 0 \\ F_{13} & 1 & 1 & 0 & 1 \\ F_{14} & 1 & 1 & 1 & 0 \\ F_{15} & 1 & 1 & 1 & 1 \end{bmatrix}$$

- F_0 - triviální identita, $f = 0$
- F_1 - logický součin, AND, $f = a \cdot b$
- F_2 - přímá inhibice, $f = a \cdot \overline{b}$
- F_3 - funkce identická, $f = a$
- F_4 - zpětná inhibice, $f = \overline{a} \cdot b$

- F_5 - funkce identická, $f = b$
- F_6 - nonekvivalence, XOR, $f = a \oplus b$
- F_7 - logický součet, OR, $f = a + b$
- F_8 - negovaný logický součet, NOR, $f = \overline{a + b}$
- F_9 - ekvivalence, XNOR, $f = \overline{a \oplus b}$
- F_10 - negace, NOT, $f = \overline{b}$
- F_11 - zpětná implikace, $f = a + \overline{b}$
- F_12 - negace, NOT, $f = \overline{a}$
- F_13 - přímá implikace, $f = \overline{a} + b$
- F_14 - negovaný logický součin, NAND, $f = \overline{a \cdot b}$
- F_15 -triviální identita, $f = 1$

4.6 Logické členy

Logické členy jsou implementací vybraných logických funkcí dvou proměnných. V základu lze libovolně složitou logickou funkci (obvod) zredukovat na kombinaci základních logických členů - **úplný soubor logických funkcí**. Pomocí těchto základních funkcí lze realizovat libovolný logický obvod:

- Logický součin - AND - $f = a \cdot b$
- Logický součet - OR - $f = a + b$
- Logická negace - NOT - $f = \overline{a}$

Pomocí logické funkce AND, OR a NOT lze vyjádřit logické funkce NAND A NOR:

- Negovaný logický součin - NAND - $f = \overline{a \cdot b}$
- Negovaný logický součet - NOR - $f = \overline{a + b}$

Rovněž pomocí logických funkcí NAND a NOR lze realizovat jakýkoli číslicový obvod. Dalšími důležitými logickými funkcemi jsou:

- Nonekvivalence - XOR - $f = a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$
- Ekvivalence - XNOR - $f = \overline{a \oplus b} = a \cdot b + \overline{a} \cdot \overline{b}$

Obě funkce jsou důležité pro realizaci aritmetických obvodů a číslicových komparátorů.

4.7 Způsoby popisu logických funkcí

Při konstrukci logických obvodů je nutné definovat a

popsat chování jednotlivých logických funkcí. To znamená určit jejich výstupní hodnoty na základě vstupních hodnot. K tomuto účelu se využívá několik matematických nástrojů.

4.7.1 Pravdivostní tabulka

Pravdivostní tabulka je nejběžnějším způsobem popisu logických funkcí. Popisuje zcela přesně chování logického obvodu, ale neobsahuje žádný návod pro jeho realizaci. Jedná se tedy pouze o model chování logického systému. Obsahuje výčet všech kombinací vstupních proměnných a jim odpovídajících výstupů. Má-li logická funkce n nezávislých proměnných, bude mít pravdivostní tabulka 2^n řádků.

Pravdivostní tabulka se skládá z několika částí. První část se označuje písmenem *N*. Jedná se o **stavový index řádku**, který zároveň udává hodnotu vstupního vektoru (kombinace vstupních hodnot). Další část obsahuje seznam vstupních proměnných, které se většinou označují počátečními písmeny anglické abecedy. Definují obor hodnot funkce, tedy všechny kombinace logických hodnot proměnných, které mohou přijít na vstup funkce, Třetí část obsahuje tzv. **mintermy** a **maxtermy**. V poslední části se vyskytují výstupní hodnoty logických funkcí na určitém indexu - kombinaci vstupních proměnných. Sloupec vyjadřující danou logickou funkci tedy definuje

definiční obor dané logické funkce.

$$\begin{bmatrix} N & A & B & MIN & MAX & Y_0 & Y_1 \\ 0 & 0 & 0 & \overline{a} \cdot \overline{b} & a + b & 0 & 0 \\ 1 & 0 & 1 & \overline{a} \cdot b & a + \overline{b} & 1 & X \\ 2 & 1 & 0 & a \cdot \overline{b} & \overline{a} + b & X & 0 \\ 3 & 1 & 1 & \overline{a} \cdot \overline{b} & \overline{a} + \overline{b} & 1 & 1 \end{bmatrix}$$

Pravdivostní tabulka může vyjadřovat určitou, ale i neurčitou hodnotu funkce. Neurčitá hodnota logické funkce se značí písmenem X, které říká, že při těchto kombinacích vstupních proměnných je lhostejno, jestli logická funkce bude vracet hodnotu 1 nebo 0. Jedná se tedy o nespojitou funkci, která pro danou kombinaci logických hodnot není definovaná.

Z pravdivostní tabulky lze získat logické výrazy pro jednotlivé logické funkce. Logický výraz pro funkci F_i může být z pravdivostní tabulky získán dvěma způsoby:

- Součtovou formou
- Součinovou formou

Podle toho jestli jsou k popisu logické funkce použity řádky, v nichž je funkce jedničková nebo nulová.

Základní součinový člen je součin, který obsahuje všechny vstupní proměnné, tak aby byl jeho výsledek roven stavu logické jedničky. Je také označován jako **minterm**.

Základní součtový člen je součet, který obsahuje všechny vstupní proměnné, tak aby jeho výsledek byl roven stavu logické nuly. Je také označován jako **maxterm**.

Pro účely úpravy logických funkcí je důležité, že maxtermy jsou negací mintermů.

Úplná součtová forma logické funkce je dána součtem základních součinových členů (mintermů), ve kterých logická funkce nabývá logické hodnoty 1:

$$F_i = \sum_{m=0}^{N} Minterm_m = Minterm_m + Minterm_{m+1} + ... + Minterm_N$$

Úplná součinová forma logické funkce je dána součinem základních součtových členů (maxtermů), ve kterých logická funkce nabývá logické hodnoty 0:

$$F_i = \prod_{m=0}^{N} Maxterm_m = Maxterm_m + Maxterm_{m+1} + ... + Maxterm_N$$

Úplná součtová i součinová forma logické funkce obsahují v každém členu všechny proměnné a nejsou proto z hlediska realizace základními logickými členy vhodné. Vstupují do minimalizačních procedur, jejichž snahou je získat minimální formu logického výrazu, která bude obsahovat minimální počet členů a v každém z nich minimální počet proměnných.

4.7.2 Seznam stavových indexů

Seznam stavových indexů představuje zjednodušený zápis pravdivostní tabulky. Stavovým indexem se rozumí dekadická hodnota kombinace binárních vstupních proměnných. Logickou funkci pak lze zapsat jako seznam stavových indexů vstupních kombinací, pro něž logická funkce nabývá hodnoty 1, nebo seznam indexů, pro něž nabývá hodnoty 0. Stavové indexy lze vyjádřit buď v součtové, nebo v součinové formě.

Součtová forma obsahuje stavové indexy, ve kterých daná funkce nabývá hodnoty jedna:

$$F_i = \sum(n_0, n_1, ...n_N)$$

U kterých platí $F_i(N) = 1$ (funkce F_i na stavovém indexu N je rovna jedné).

Součinová forma obsahuje stavové indexy, které v dané funkci nabývají hodnoty nula.

$$F_i = \prod(n_0, n_1, ...n_N)$$

U kterých platí $F_i(N) = 0$ (funkce F_i na stavovém indexu N je rovna nule).

4.7.3 Logický výraz

Logický výraz je popisem logické funkce pomocí logických proměnných ve formě **analytického popisu**. Logická funkce definuje chování dané výstupní proměnné. Mintermy a maxtermy jsou dílčí logické výrazy, které říkají za jakých okolností nabude výstup logické hodnoty jedna nebo nula. Maxtermy nebo mintermy pak tvoří logický výraz popisující chování logické funkce. V závislosti na tom jestli je pro realizaci logického výrazu použita součtová nebo součinová forma jsou použity mintermy nebo maxtermy. Použití součtové nebo součinové formy vyplývá z počtu jedniček nebo nul pro všechny vstupní kombinace logických hodnot. Pokud se v oboru hodnot dané funkce vyskytuje více jedniček než nul je z důvodu kratšího a tím jednoduššího logického výrazu výhodnější použít součtovou formu, pokud se v oboru hodnot vyskytuje více nul než jedniček je naopak vhodnější součinová

forma. Každému logickému výrazu odpovídá jednoznačně **obvodová struktura**, logický výraz tedy lze považovat za model struktury logického obvodu:

$$y = f(a, b, c, ...)$$

4.7.4 Zobrazení pomocí map

4.8 Popis logických členů

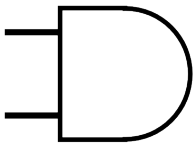
Složitější logické funkce se realizují pomocí základních logických členů, jejichž spojováním lze získat komplexní logický obvod požadovaných charakteristik. Pro kreslení technických schémat se používají schématické značky těchto základních členů.

4.8.1 Logický součin - AND

Logický součin, označovaný zkratkou AND je logická funkce dvou nebo více vstupních logických proměnných. Výstupem logické funkce AND je logická hodnota jedna *pouze v případě, že se na vstupu nacházejí samé jedničky.*

$$y = a \cdot b$$

V technických schématech se pro značení funkce AND používá nejčastěji schématická značka:



Popis chování funkce AND v pravdivostní tabulce:

$$\begin{bmatrix} N & A & B & AND \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 \end{bmatrix}$$

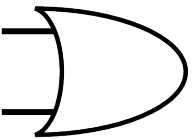
4.8.2 Logický součet - OR

Logický součet, označovaný zkratkou OR je logická funkce dvou nebo více logických proměnných. Výstupní hodnotou logické funkce OR je logická hodnota jedna v případě, že je alespoň jeden ze vstupů je nastavena na hodnotu jedna.

$$y = a + b$$

V technických schématech se pro značení funkce OR

používá nejčastěji schématická značka:



Popis chování funkce AND v pravdivostní tabulce:

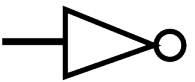
$$\begin{bmatrix} N & A & B & OR \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 3 & 1 & 1 & 1 \end{bmatrix}$$

4.8.3 Negace - NOT

Negace je nejjednodušší logickou funkcí. **Negace**, označována zkratkou NOT je logická funkce jedné logické proměnné. Výsledkem funkce negace je opačná logická hodnota, než je přivedená na vstup.

$$y = \overline{a}$$

V technických schématech se pro značení funkce NOT používá nejčastěji schématická značka:



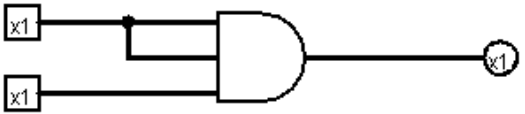
Popis chování funkce NOT v pravdivostní tabulce:

$$\begin{bmatrix} N & A & NOT \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

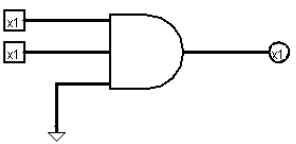
4.9 Ošetření nezapojených vstupů

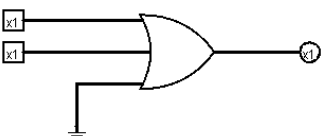
Logická hradla jsou realizována jako vícevstupé integrované obvody. Často je v praxi problém jak zapojit nevyužité vstupy hradel. Rušení doprovázené provozem logických obvodů (rychlá změna logických stavů na vstupech hradla) může způsobit náhodné stavy na těchto vstupech a tím i náhodné chování celého systému.

První možností je nepoužitý vstup spojit s jedním se zapojených vstupů. Tím se ale zvyšuje proudové zatížení předchozího výstupu, ale je to možnost nejjednodušší a nejpoužívanější.



Druhou možností je propojit nezapojený vstup k napájecímu napětí tak, aby tímto připojením nebyla změněna logická funkce pro daný obvod. Pro hradla NOR a OR se nezapojený vstup připojuje na logickou hodnotu 0, tedy k zápornému pólu napětí (zem). Pro NAND a AND se nezapojený vstup připojuje na logickou hodnotu 1, tedy ke kladnému pólu napětí. To vychází ze zákona neutrality nuly a jedničky





Pokud je nutné realizovat kombinační funkce pomocí hradla jednoho druhu je nutné použít buď hradlo NOR nebo NAND. Tato hradla umožňuje realizovat libovolnou logickou funkci. Invertor lze z těchto hradel vytvorit jednoduše:

$$\begin{bmatrix} N & A & A & NAND & NOR \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

4.10 Logická algebra

Logická algebra je matematická disciplína, která je využívána při návrhu číslicových obvodů, algoritmů, ... Zahrnuje pravidla s logickými proměnnými a funkcemi. Při používání pravidel se využívají tři základní operace:

- Logický součet (disjunkce)
- Logický součin (konjunkce)
- Negace (inverze)

Tyto základní operace tvoří teoretický prostředek pro návrh (syntézu) logických obvodů s požadovaným chováním. Mezi binárními logickými proměnnými jsou definovány základní vztahy, které jsou využívány při minimalizaci logických funkcí, aby jejich obvodová realizace

byla co nejjednodušší. Vztahy binárních proměnných jsou definovány v duální formě - součtové a součinové.

4.10.1 Neutrálnost nuly a jedničky

Zákon pojednává o definici *neutrálního prvku* v operacích logického součtu a součinu:

$$a \cdot 1 = a$$

$$a + 0 = a$$

Neutrální prvek v binární logické operaci AND/OR ve výsledku vrátí hodnotu druhé operandu.

4.10.2 Zákon negace

Inverzní prvek je takový prvek, který je opačný k prvku původnímu. Zákon negace říká, že logická hodnota negovaného prvku je inverzní (opační) k logickému prvku původnímu.

$$\overline{1} = 0$$

$$\overline{0} = 1$$

4.10.3 Zákon dvojí negace

Zákon dvojí negace říká, že pokud je libovolná logická hodnota znegována dvakrát po sobě, je získána opět původní logická hodnota:

$$\overline{\overline{a}} = a$$

4.10.4 Agresivnost nuly a jedničky

Zákon agresivity nuly a jedničky pojednává o působení logické hodnoty nuly a jedničky v operacích AND a OR.

$$a \cdot 0 = 0$$

$$a + 1 = 1$$

Podle definice logického součtu platí, že pokud je jeden z operandů logická jednička, nezáleží na hodnotě druhého operandu a výsledkem bude vždy logická jednička.

Podle definice logického součinu platí, že pokud je jeden z operandů logická nula, nezáleží na hodnotě druhého operandu a výsledkem bude vždy logická nula.

4.10.5 Zákon vyloučení třetího

Zákon pojednává o možnosti pokrátit (zjednodušit) pomocí operace negace výraz s jednou logickou proměnnou. Negace v logické operaci invertuje hodnotu druhého operandu na jeho opačnou hodnotu, z toho vyplývá, že hodnota jednoho prvku je vždy rovna logické hodnotě 1. Pak již platí zákon agresivity nuly a jedničky. Jedná se o rozšíření pravidla agresivní nuly a jedničky.

$$a \cdot \overline{a} = 0$$

$$a + \overline{a} = 1$$

4.10.6 Zákon idempotence

Zákon idempotence říká, pro dva stejné vstupní operandy má daná binární operace stejnou výstupní hodnotu jako mají vstupní hodnoty. Dvě jedničky nebo dvě nuly dají dle zákonů logického součtu a součinu na výstup hodnotu vstupu:

$$a \cdot a = a$$

$$a + a = a$$

4.10.7 Zákon absorbce

Zákon absorbce pojednává o zbytečnosti druhé proměnné ve výrazu kombinující logický součet a logický součin

$$a + a \cdot b = a$$

$$a \cdot (a + b) = a$$

4.10.8 Zákon komutativní

Komutativní zákon pojednává o tom, že nezáleží na pořadí operandů v operacích logický součin a logický součet. Logický součet a logický součin se chovají v tomto ohledu stejně jako klasický algebraický součet a součin. To znamená, že výsledek není ovlivněn pořadím operandů ve výrazu.

$$a \cdot b = b \cdot a$$

$$a + b = b + a$$

4.10.9 Asociativní zákon

Ve výrazu s více než dvěma operandy nezáleží na pořadí výpočtu těchto binárních operátorů. Jedná se o rozšíření komutativního zákona.

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

$$a + (b + c) = (a + b) + c$$

4.10.10 Distributivní zákon

Je to vlastnost binární operace vůči jiné binární operaci, říkající, že lze danou operaci distribuovat přes jinou operaci.

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

V případě součinové formy platí stejná pravidla jako v klasické algebře, kdy daná proměnná násobí všechny proměnné uvnitř závorky.

Při dokazování platnosti distributivního zákona v součtové formě je využito zákona absorbce u členů ac a ab.

$$(a+b)\cdot(a+c) = (a\cdot a)+(a\cdot c)+(b\cdot a)+(b\cdot c)$$

$$\underbrace{(a\cdot a)}_{a}+(a\cdot c)+(b\cdot a)+(b\cdot c) = a+(a\cdot c)+(b\cdot a)+(b\cdot c)$$

$$\underbrace{a+(a\cdot c)}_{a}+(b\cdot a)+(b\cdot c) = a+(b\cdot a)+(b\cdot c)$$

$$\underbrace{a+(b\cdot a)}_{a}+(b\cdot c) = \underline{\underline{a+(b\cdot c)}}$$

4.10.11 Zákon absorbce negace

Zákon absorbce negace popisuje jak lze ve výrazu kde se nacházejí dvě stejné proměnné, přičemž jedna z nich je negována, jednoduše odstranit její negovanou část.

$$a\cdot(\overline{a}+b) = a\cdot b$$

$$a + (\overline{a} \cdot b) = a + b$$

Důkaz pro součinový tvar:

$$a \cdot (\overline{a} + b) = \underbrace{(a \cdot \overline{a})}_{0} + a \cdot b = a \cdot b$$

Důkaz pro součtový tvar:

$$a + (\overline{a} \cdot b) = \underbrace{(a + \overline{a})}_{1} \cdot (a + b) = a + b$$

Nejprve je výraz rozložen na dílčí výrazy pomocí distributivního zákona. Následně lze využít zákon vyloučení třetího a nakonec je využit zákon neutrality nuly a jedničky.

4.10.12 Vztah mezi logickým součtem a logickým součinem

Jedná se o vyjádření vztahu mezi logickým součtem a logickým součinem, kdy lze jednu operaci vyjádřit z druhé:

$$\overline{a + b} = \overline{a} \cdot \overline{b} \Leftrightarrow a + b = \overline{\overline{a} \cdot \overline{b}}$$

$$\overline{a \cdot b} = \overline{a} + \overline{b} \Leftrightarrow a \cdot b = \overline{\overline{a} + \overline{b}}$$

Je rozdíl mezi negací jednotlivých operandů dané operace a negací celého výrazu. Pokud jsou znegovány pouze jednotlivé proměnné výrazu, jsou jednotlivé výstupy zachovány, ale jsou vertikálně obracené:

N	A	B	$(A+B)$	$(A \cdot B)$	$\overline{A}$	$\overline{B}$	$(\overline{A}+\overline{B})$	$(\overline{A} \cdot \overline{B})$
0	0	0	0	0	1	1	1	1
1	0	1	1	0	1	0	1	0
2	1	0	1	0	0	1	1	0
3	1	1	1	1	0	0	0	0

Pokud je ale znegován celý výraz jsou znegovány jeho jednotlivé výstupy, platí:

N	A	B	$(A+B)$	$(A \cdot B)$	$(\overline{A+B})$	$(\overline{A \cdot B})$
0	0	0	0	0	1	1
1	0	1	1	0	0	1
2	1	0	1	0	0	1
3	1	1	1	1	0	0

Definice logického součtu říká: aby byl logický součet dvou logických proměnných pravdivý, musí být pravdivá

alespoň jedena z těchto proměnných. Logický součin dvou logických proměnných je nepravdivý, pokud je alespoň jedena z těchto vstupních proměnných nepravdivá. Z toho lze vyvodit určitý vztah mezi logickými operacemi součtu a součinu.

N	A	B	$(A+B)$	$(A\cdot B)$	$(\overline{A}+\overline{B})$	$(\overline{A}\cdot\overline{B})$	$(\overline{A+B})$	$(\overline{A$
0	0	0	0	0	1	1	1	
1	0	1	1	0	1	0	0	
2	1	0	1	0	1	0	0	
3	1	1	1	1	0	0	0	

Toto nepopisuje pouze vztah mezi logickým součtem a součinem, tedy metodu, díky které lze logický součin vyjádřit pomocí logického součtu a negace a naopak, ale také vztah mezi negací jednotlivých operandů výrazu a negací celého výrazu. Negace výrazu a negace proměnných ve výrazu jsou dvě různé věci a nelze je při minimalizaci opomenout (zaměnit). Toto je metoda, jak převést negaci výrazu na negaci proměnných a naopak. Toho se využívá ne jen při minimalizaci logických funkcí, ale také při konstrukci logických obvodů, kdy je třeba využít pouze logických hradel NOR (a) nebo NAND.

Při převodu funkce AND na funkci OR nebo naopak se využívá zákon dvojí negace, která nezmění platnost logického výrazu:

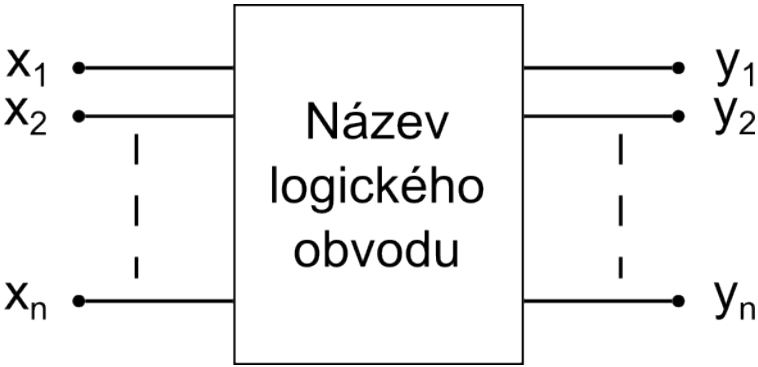
$$a \cdot b = \overline{\overline{a \cdot b}} = \overline{\overline{a} + \overline{b}}$$

Kapitola 5
Kombinační logické obvody

Kombinační logický obvod je obvod, jehož výstupní hodnota je závislá pouze na kombinaci vstupních hodnot. Tyto obvody jsou popisovány pomocí logických funkcí, které definují chování logického systému. Kombinační logické obvody lze rozdělit do několika základních skupin:

- Dekodéry
- Multiplexery a demultiplexery
- Komparátory
- Obvody pro aritmetické operace

Složitější logické obvody jsou pak sestavovány kombinací základních logických členů AND, OR a NOT. Pomocí těchto členů lze vytvořit libovolný logický obvod, ale schémata takovýchto logických obvodů by byla příliš rozsáhlá a složitá, proto bylo vytvořeno jednotné schéma kombinačního logického obvodu, které zapouzdřuje rozsáhlá a složitá schémata. Díky tomu je možné výsledné schémata výrazně zjednodušit. Obecné schéma kombinačního logického obvodu vypadá takto:



5.1 Syntéza kombinačních logických obvodů

Syntézou kombinačních logických obvodů se chápe postup, kdy ze slovního zadání je získán výsledný kombinační obvod, realizující požadované funkce. K tomu jsou využívány principy logiky a logické algebry. Postup syntézi kombinačních logických obvodů lze rozdělit do několika bodů:

- Slovní zadání logické funkce
- Popis logické funkce (nejčastěji pravdivostní tabulka)
- Minimalizace logické funkce

• Realizace kombinačního obvodu pomocí základních logických členů

5.2 Dekodéry

Dekodéry, popřípadě kodéry jsou kombinační logické obvody, které v závislosti na kombinaci vstupních proměnných generují určitý kód - kombinaci výstupních binárních stavů. Dekodéry se dělí podle kódu, které na svém výstupu generují:

5.2.1 Dekodér z binárního kódu do kódu 1 z N

Dekodér z binárního kódu do kódu 1 z N je logický obvod o n vstupech a až 2^n možných výstupů. Pomocí vstupních proměnných jsou adresovány jednotlivé výstupy. Jedná se tedy o logický obvod, který slouží jako přepínač. Nejjednodušším dekodérem tohoto typu je dekodér 1 ze 4. Pravdivostní tabulka má 2 vstupní a 4 výstupní proměnné:

$$\begin{bmatrix} N & A & B & Y_0 & Y_1 & Y_2 & Y_3 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Z pravdivostní tabulky lze pro jednotlivé výstupní proměnné získat logické funkce, které ale nelze již dále mini-

malizovat:

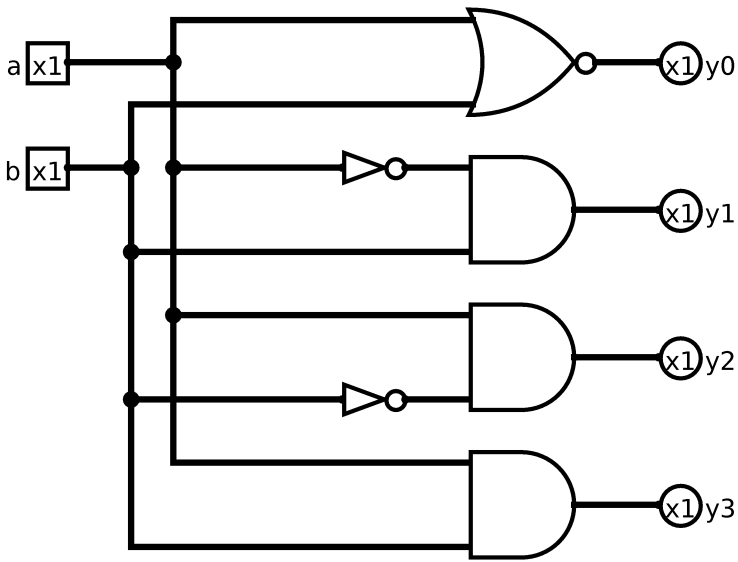
$$y_0 = \overline{a} \cdot \overline{b} = \overline{(a + b)}$$

$$y_1 = \overline{a} \cdot b$$

$$y_2 = a \cdot \overline{b}$$

$$y_2 = a \cdot b$$

Podle získaných logických funkcí lze získat výsledný obvod:



K mnoha logickým obvodům (ne jen k dekoderům) se často přidává ještě jeden nadbytečný vstup, který plní funkci vypínače. Tento vstup blokuje funkci obvodu pokud je nastaven na hodnotu logiké nuly. Pokud je tento vstup nastavený na hodnotu jedna, logický obvod reaguje na změny vstupu a plní svou funkci předepsanou logickými funkcemi. Pokud je ale na tomto vstupu hodnota nula,

celý obvod se chová jako vypnutý a nereaguje na změnu vstupních hodnot. Tento Vstup se obvykle označuje velkým písmenem E - Enable. Pravdivostní tabulka upraveného dekodéru:

$$\begin{bmatrix} N & E & A & B & Y_0 & Y_1 & Y_2 & Y_3 \\ 0 & 0 & 0 & 0 & X & X & X & X \\ 1 & 0 & ... & ... & ... & ... & ... & ... \\ 4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 5 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 6 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 7 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Obvod vykonává stejnou funkci, ale pouze pokud je vstup E na staven na hodnotu 1. Logické funkce pro jednotlivé výstupy:

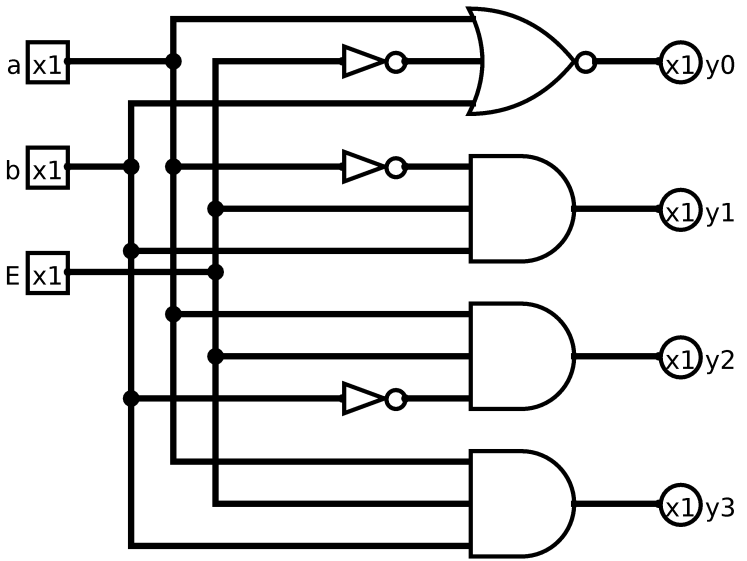
$$Y_0 = E \cdot \overline{A} \cdot \overline{B} = \overline{\overline{E} \cdot A + B}$$

$$Y_0 = E \cdot \overline{A} \cdot B$$

$$Y_0 = E \cdot A \cdot \overline{B}$$

$$Y_0 = E \cdot A \cdot B$$

Podle získaných logických funkcí lze získat výsledný obvod:

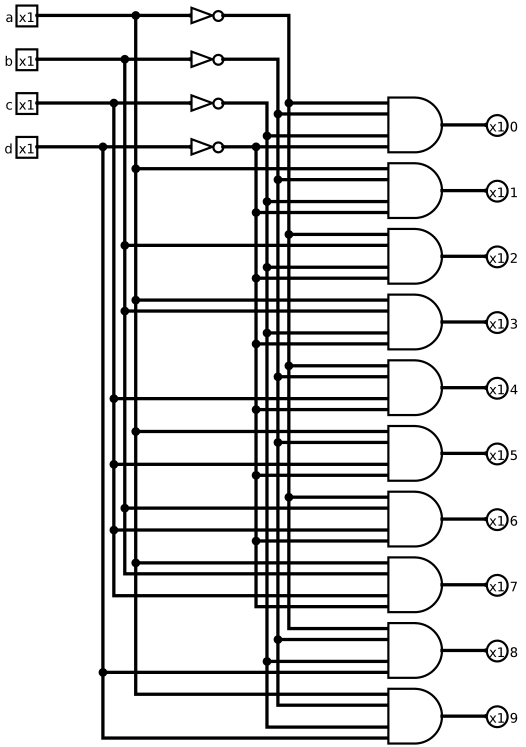


5.2.2 Kodér z BCD kódu na kód 1 z 10

Kodér z BCD kódu na kód 1 z 10 je převodník z BCD čísel na čísla desítková. Vstupem je čtyřbitové slovo v BCD

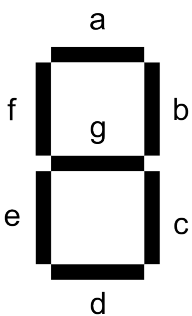
kódu a výstupem je logická hodnota 1 (nebo 0 v závislosti na logice výstupních hodnot) na jednom z deseti výstupů reprezentující číslo 0 - 9. Protože čtyřmi vstupy lze definovat 16 různých kombinací, při deseti výstupech nebudou všechny kombinace využity - kombinace 1010 až 1111 nevyvolají žádný výstup.

N	A	B	C	D	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y
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



5.2.3 Kodér pro sedmi-segmentové displeje

Kodér je navržen pro buzení (ovládání) zobrazovacích jednotek tvořených sedmi-segmentovými displeji. Jeho výstupní stavy rozsvěcují některé ze sedmi světelných segmentů tak, aby tvořily dekadické číslice 0 - 9 a hexadecimální číslice A - F. Tento kodér používá jako vstup čtyřbitové číslo v BCD kódu a výstupem je sedm signálů v sedmi-segmentovém kódu.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Segmenty jsou označeny písmeny a, b, c, ..., g. Stejné označení mají i výstupy kodéru.

Existují dva druhy sedmi-segmentových displejů - **se**

společnou anodou a **se společnou katodou**. V prvním případě jsou spojeny anody diodových segmentů, které se připojí na kladné napájecí napětí. Kodér pak rozsvítí jednotlivé segmenty přivedením logické nuly na vstupy sedmi-segmentového displeje. Svítící segmenty jsou označeny logickou nulou.

Displeje se společnou katodou představuje propojení všech katod a jejich připojení na záporné napájecí napětí. Segmenty se v tomto připojení rozsvítí přivedením logické jedničky na jednotlivé anody. Zapojení sedmi-segmentového displeje se společnou anodou je negací zapojení se společnou katodou. Svítící segmenty jsou označeny logickou jedničkou.

Příklad pravdivostní tabulky popisující dekodér 7-segmentového displeje se společnou anodou:

N	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0
10	1	0	1	0	0	0	0	1	0	0	0
11	1	0	1	1	1	1	0	0	0	0	0
12	1	1	0	0	0	1	1	0	0	0	0
13	1	1	0	1	1	0	0	0	0	1	0
14	1	1	1	0	0	1	1	0	0	0	0
15	1	1	1	1	0	1	1	1	0	0	0

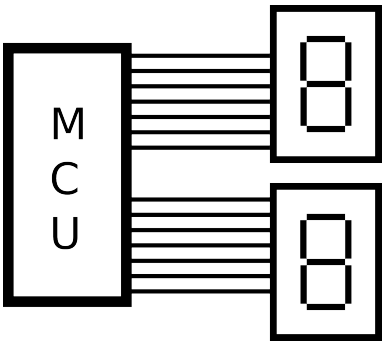
5.3 Zapojení 7-segmentového displeje

Zapojení jednoho 7-segmentového displeje je s pomocí dekodéru relativně jednoduché, ale efektivní zapojení více 7-segmentových displejů současně si žádá již komplexnější řešení, aby bylo k jeho ovládání použito co nejmnší množství ovládacích vodičů. Pro zapojení více

7-segmentových displejů je možné použít několik různých metod. K buzení signálů je nejčastěji používán mikrokontrolér (mcu).

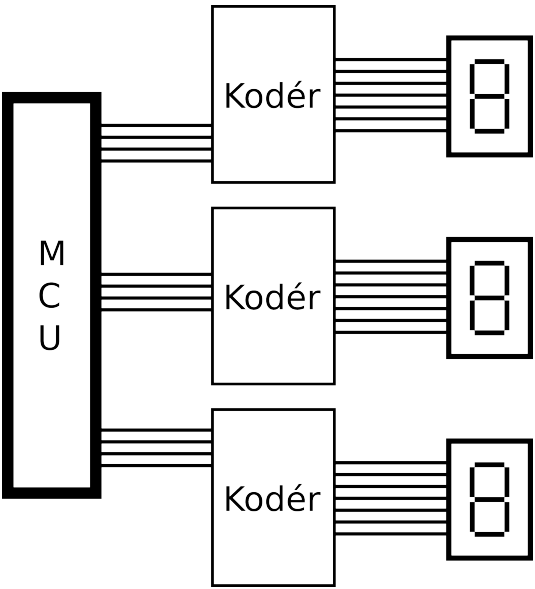
5.3.1 Přímé zapojení 7-segmentového displeje

V případě přímého zapojení 7-segmentového displeje je pro každý displej je vyvedeno 7 vývodů. Výhodou toho řešení je rychlá a jednoduchá realizace zapojení. Programovatelný mikrokontrolér obsahuje program, který převádí BCD kód do kódu sedmisegmentového displeje. Nevýhodou je množství potřebných výstupů mikrokontroléru. Tím je omezen maximální počet displejů, které lze k mikrokontroléru připojit. Zároveň jsou určité výpočetní zdroje potřeba na softwarový převod kódu BCD do kódu 7-segmentového displeje.



5.3.2 Připojení přes dekoder

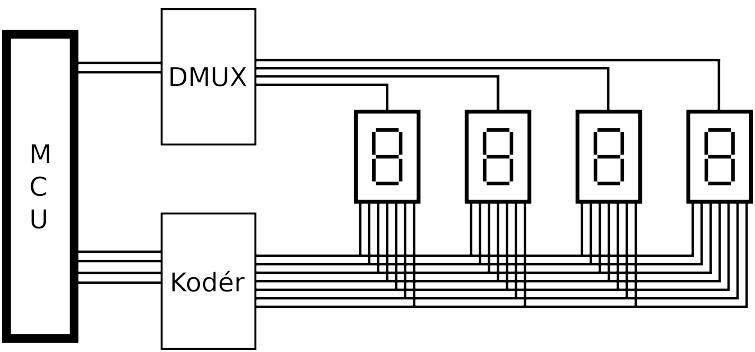
Řídící jednotka je nastavena, aby vysílala čtyřbitová binární čísla, která jsou posílání na dekodér kódu 7-segmentového displeje, který je napojeny na samostatný 7-segmentový displej. Tím, že je použito menší množství vývodů z řídící jednotky stoupá maximální počet displejů, které lze na řídící jednotku připojit za cenu vyšší složitosti výsledného zapojení.



5.3.3 Připojení na společnou sběrnici

Všechny displeje se připojí na společnou datovou sběrnici, na které je vysílán kód 7-segmentového displeje. Následně jsou všechny displeje ještě připojeny na řídící vodič, který určuje kdy má daný displej naslouchat na

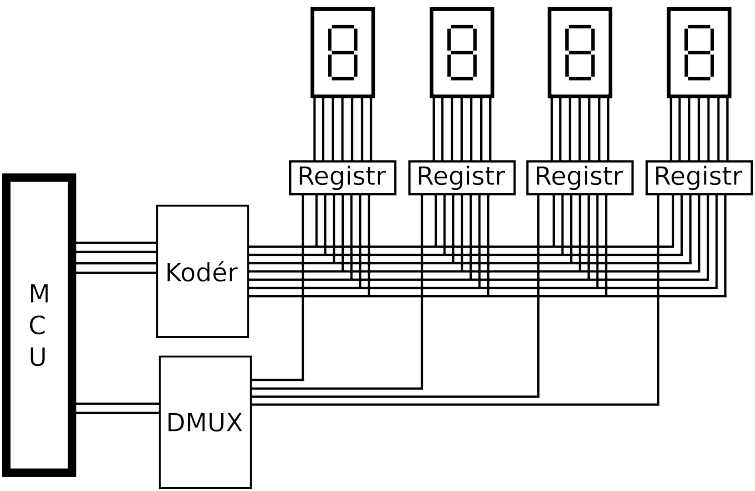
komunikační sběrnici. K tomuto přepínání slouží obvod, který se nazývá *demultiplexer*. Mikrokontrolér přepíná mezi jednotlivými displeji a zároveň vysílá data, která mají být na daném displeji zobrazena. Displeje jsou neustále přepínány tak rychle, že lidské oko nepostřehne, že není rozsvícen permanentně a zároveň každý displej zobrazuje pouze svá data. Nevýhodou tohoto řešení je nutnost cyklického přepínání mezi jednotlivými displeji. To vyžaduje určitý výkon MCU.



4.7.3 Rozšířené zapojení na společné sběrnici

Zapojení 7-segmentových displejů na společné datové sběrnici je rozšířeno o sadu registrů, které dlouhodobě

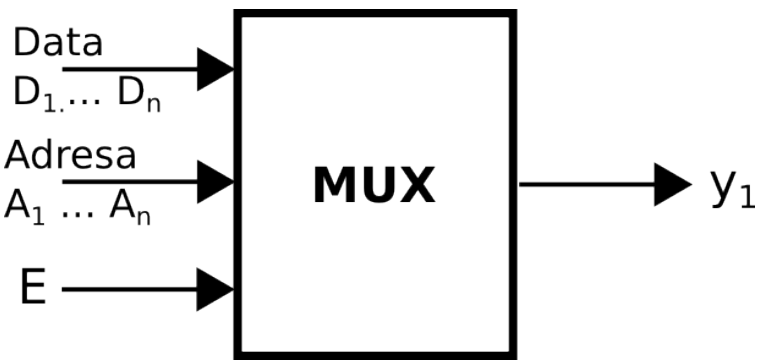
uchovávají zobrazovaná data pro daný 7-segmentový displej. Díky tomu nemusejí být data na displejích cyklicky oživována jako v případě prostého připojení na společné sběrnici. Také odpadá nutná složitost řídícího programu. Registry jsou napojeny na řídící sběrnici a jejich hodnota se změní s přivedením logické hodnoty 1 na vstup registru.



4.8 Multiplexer

Multiplexer je kombinační číslicový obvod, který

umožňuje přenášet data z několika vstupů (vždy pouze z jednoho) na jeden výstup. Jedná se o jakýsi přepínač signálů, který na jeden výstup dokáže podle nastavené adresy posílat bitová data z různých zdrojů. Obsahuje n datových vstupu $D_1, \ldots, D_n$ a k adresových vstupů $A_1, \ldots, A_k$. Většinou pak ještě obsahuje jeden vstup pro blokování obvodu označovaného jako E (Enable), který plní funkci vypínače. Obvod je aktivní s logickou hodnotou 1 na vstupu E. Platí, že s k adresovými vstupy lze obsloužit až $n = 2^k$ datových vstupů. Schématická značka multiplexeru vypadá takto:



Multiplexer se používá tam, kde má být z množství různých logických signálů vybrán jediný, například na vs-

tupech aritmeticko-logické jednotky, v obvodech pro převod paralelní informace na sériovou, ...

Při přenosu paralelní informace na sériovou je třeba vícebitová slova přenášet bit po bitu postupně v časově rozvinutém stavu. V takovém případě se na datové vstupy multiplexeru přivede vícebitové slovo a potom se postupně vystřídají všechny kombinace na adresových vstupech.

Činnost multiplexeru lze díky jedinému výstupu popsat pouze jedinou logickou funkcí, která má obecně tvar:

$$Y = (Adr_1 \cdot D_1 + Adr_2 \cdot D_2 + ... + Adr_n \cdot D_n) \cdot E$$

kde Adr_j označuje příslušnou kombinaci adresových vstupů A_1 až A_k a D_n označuje jednotlivé datové vstupy. Například kombinace pro výběr datového vstup D_1 je $\overline{(A_2 \cdot A_1)}$

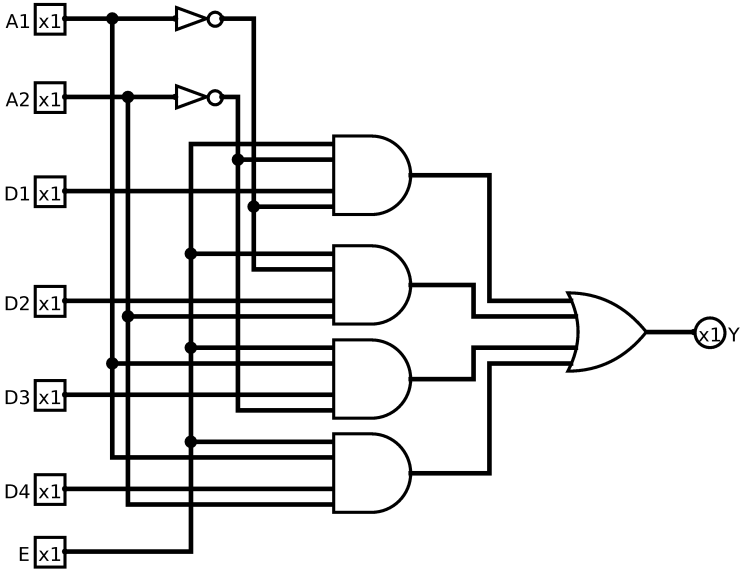
Pro multiplexer se čtyřmi datovými vstupy, dvěma adresovými vstupy a jedním blokovacím vstupem by pravdivostní tabulka obsahovala $2^7 = 128$ řádků. Počet řádků lze ale výrazně omezit pokud se vynechají řádky, kdy blokovací vstup E nabývá hodnoty 0 a tudíž na hodnotách ostatních vstupů nezáleží.

N	E	D_1	D_2	D_3	D_4	A_1	A_2	Y
0	0	X	X	X	X	0	0	0
1	1	1	X	X	X	0	0	1
2	1	0	X	X	X	0	0	0
3	1	X	1	X	X	0	1	1
4	1	X	0	X	X	0	1	0
5	1	X	X	1	X	1	0	1
6	1	X	X	0	X	1	0	0
7	1	X	X	X	1	1	1	1
8	1	X	X	X	0	1	1	0

Z pravdivostní tabulky lze získat logickou funkci ve tvaru:

$$Y = ((\overline{A_1} \cdot \overline{A_2} \cdot D_1) + (\overline{A_1} \cdot A_2 \cdot D_2) + (A_1 \cdot \overline{A_2} \cdot D_3) + (A_1 \cdot A_2 \cdot D_4))$$

Blokové schéma, které odpovídá pravdivostní tabulce a logické funkci z ní odvozené vypadá takto:

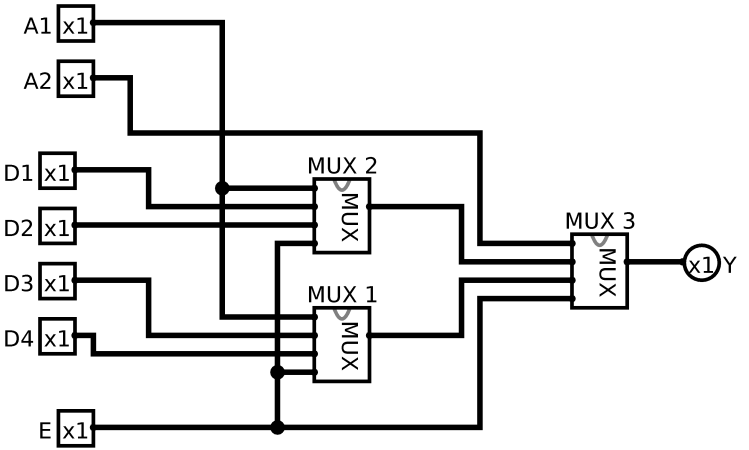


4.8.1 Paralelní spojování multiplexerů

Pokud by bylo potřeba přepínat mezi více vstupy než je dostupných na obvodu multiplexeru, je možné zapojit více multiplexerů paralelně. V paralelním zapojení jsou rozlišovány vstupní multiplexery a koncový multiplexer. Výstupy vstupních multipliplexerů jsou napojeny na vstupy

koncového multiplexeru. Při paralelním zapojení multiplexerů jde o to, že spodní část adresy tvoří adresy do jednotlivých dílčích multiplexerů a horní část adresy přepíná mezi vstupy koncového multiplexeru, na které jsou napojeny výstupy jednotlivých dílčích multiplexerů. To znamená, že lze na koncový multiplexer připojit pouze tolik vstupních multiplexerů kolik má datových vstupů. Adresové vstupy vstupních multiplexerů jsou propojené na stejnolehlých indexech, proto při dané adrese jsou aktivní všechny stejnolehlé datové vstupy jednotlivých vstupních multiplexerů (spodní část adresy), ale pouze v závislosti na adrese výstupního multiplexeru (horní část adresy) je výstup z konkrétního vstupního multiplexeru promítnut na výstup koncového multiplexeru.

Schéma jednoduchého paralelního zapojení multiplexerů se dvěma datovými vstupy:

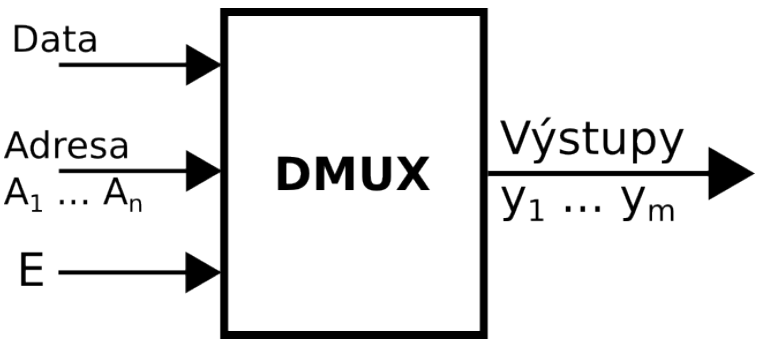


4.9 Demultiplexer

Demultiplexer je kombinační logický obvod, který plní přesně opačnou funkci než multiplexer. V závislosti na hodnotě adresových vstupů přenáší signál z jediného datového vstupu na některý z výstupů, zatímco na ostatních výstupech setrvává neaktivní stav. Demultiplexer lze popsat sadou logických funkcí (jedna pro každý výstup), které mají obecný tvar:

$$Y_i = Adr_i \cdot D \cdot E$$

kde Adr_i je kombinace adresových stupů $A_1, A_2, \ldots A_n$ s jejichž pomocí se vybírá jeden z výstupů $Y_1, Y_2, \ldots Y_m$. Schématická značka demultiplexeru vypadá takto:



Jedná se o podobný princip jako u dekodérů 1 z n. U dekodéru ale chybí datový vstup. Ten lze ale nahradit dodáním logického hradla NAND na vstup E, na který se připojí datový vstup D a řídící vstup E. Pouze pokud jsou oba vstupy nastaveny na logickou jedničku je na výstupu daném adresou také logická jednička.

Příklad pravdivostní tabulky demultiplexeru se čtyřmi

výstupy:

N	E	A_1	A_2	D	Y_1	Y_2	Y_3	Y_4
0	0	X	X	X	0	0	0	0
1	1	0	0	1	1	0	0	0
2	1	0	0	0	0	0	0	0
3	1	0	1	1	0	1	0	0
4	1	0	1	0	0	0	0	0
5	1	1	0	1	0	0	1	0
6	1	1	0	0	0	0	0	0
7	1	1	1	1	0	0	0	1
8	1	1	1	0	0	0	0	0

Z pravdivostní tabulky lze odvodit logické funkce:

$$Y_1 = \overline{A_1} \cdot \overline{A_2} \cdot D \cdot E$$

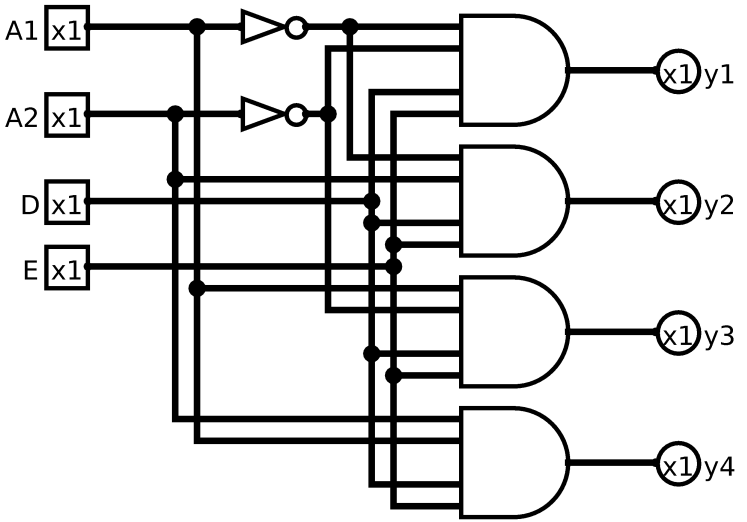
$$Y_1 = \overline{A_1} \cdot A_2 \cdot D \cdot E$$

$$Y_1 = A_1 \cdot \overline{A_2} \cdot D \cdot E$$

$$Y_1 = A_1 \cdot A_2 \cdot D \cdot E$$

Blokové schéma, které odpovídá pravdivostní tabulce a logickým

funkcím z ní odvozené vypadá takto:



4.9.1 Spojování demultiplexerů

Pokud by bylo potřeba přepínat v mezi více obvody než je dostupných výstupů na obvodu demultiplexeru, je možné zapojit více demultiplexerů paralelně. Řídící demultiplexer na základě horní části adresy přepíná mezi blokovacími vstupy E koncových demultiplexerů. Adresové vstupy koncových demultiplexerů jsou na stejnolehlých vstu-

pech propojeny a na základě adresy řídícího demultiplexeru je zapnutý pouze jeden z těchto koncových demultiplexerů.

4.10 Komparátory

Komparátor je kombinační logický obvod, který porovnává dvě binární čísla a generuje výstupní signál o jejich rovnosti nebo různosti. Nejjednodušší komparátor je logická funkce ekvivalence, která je označována jako XNOR. XNOR přijímá dvě jednobitová čísla a vrací na výstup logickou hodnotu jedna v případě, že tato čísla jsou stejná. Podobně funguje také logická funkce XOR. XOR vrací na výstup logickou hodnotu nula v případě, že vstupní jednobitová čísla jsou různá.

$$\begin{bmatrix} N & A & B & XOR & XNOR \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Funkce ekvivalence a nonekvivalence lze realizovat pomocí základních logických členů AND, OR a NOT:

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

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

Z toho vyplývá, že funkce XNOR je negace funkce XOR.

Schéma zapojení funkce XOR:

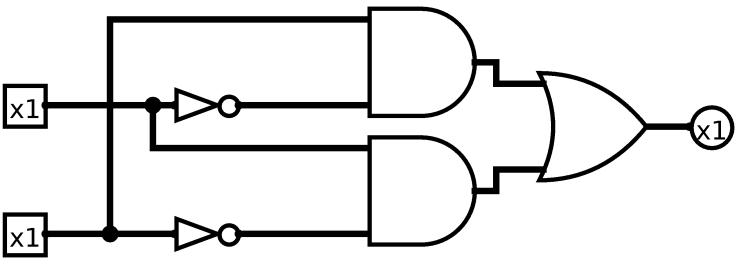
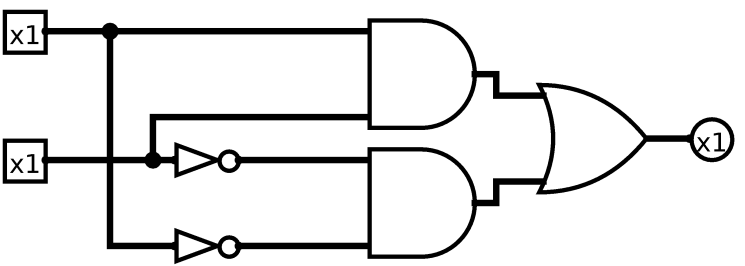


Schéma Zapojení funkce XNOR:



Obvody XOR a XNOR nacházejí uplatnění v komparátorech a obvodech pro aritmetické operace. Komparátor pro porovnáván dvou vícebitových čísel se konstruují pomocí obvodů XOR s aktivní nulou (pokud se čísla shodují výsledkem je logická nula) a nebo pomocí XNOR s aktivní jedničkou (pokud se čísla shodují je výsledkem logická jednička). Výstupy pak musejí být spojeny v jeden výstupy pomocí hradla AND v případě komparátoru konstruovaného z hradel XNOR s aktivní jedničkou nebo NAND s aktivní nulou a pomocí hradla OR v případě komparátoru konstruovaného pomocí hradel XOR s aktivní nulou nebo NOR s aktivní jedničkou.

4.10.1 Sudá a lichá parita

Parita je jeden ze způsobů jakým kontrolovat chyby při přenosu dat. Princip paritní kontroly je přidání redundantního (datově nadbytečného) bitu - **paritní bit** k přenášeným datům, který informuje o sudém nebo lichém počtu logických stavů v datovém slovu.

Rozlišuje se **sudá parita** a **lichá parita**. Sudá parita nastaví paritní bit na logickou hodnotu jedna, v případě, že se v datovém slově nachází lichý počet logických hodnot jedna (aby v přenesených datech bylo ve výsledku sudý počet jedniček). Lichá parita nastaví paritní bit na logickou hodnotu jedna v případě, že se v datovém slově nachází

sudý počet logických hodnot jedna (aby v přenesených datech bylo ve výsledku lichý počet jedniček)

pravdivostní tabulka sudé a liché parity pro tři datové bity:

N	x_2	x_1	x_0	Y_s	Y_l
0	0	0	0	0	1
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	0

Z pravdivostní tabulky je zjevné, že logická funkce liché parity je negací funkce sudé parity. Z předpisu lze získat logické funkce:

$Y_s =$

$Y_l =$

Kapitola 5
Obvody pro aritmetické operace

Aritmetické operace s binárními čísly jsou základní programové vybavení mikroprocesorů a hardwarového vybavení specializovaných integrovaných obvodů. Základní aritmetické operace jsou součet, rozdíl, součin a podíl. Veškeré operace lze softwarově realizovat pomocí operace součet. Tímto způsobem jsou řešeny jednodušší a levnější obvody. Aritmetické obvody bývají realizovány buď jako samostatné obvody a nebo se nacházejí v jednom kompaktním obvodu, který se nazývá **aritmeticko-logická jednotka**. Aritmeticko-logická jednotka je nedílnou součástí všech programovatelných procesorů.

5.1 Binární sčítačka

Sečtení dvou dvojkových čísel je založeno na sečtení všech dvojic dvojkových míst, které si vzájemně řádově odpovídají počínaje nejnižšími řády. Přitom je třeba brát v úvahu i *přenosy z nižších do vyšších řádů*, které během sčítání vznikají.

Při sčítání dvou dvojkových čísel se musejí nejdříve

sečíst hodnoty na nejnižším řádovém místě (na řádovém místě odpovídající pozici 20). Při tom může vzniknout aritmetický součet nula, jedna a dvě a vzato binárně v případě hodnoty dvě nestačí pro jeho reprezentaci pouze jeden bit. Vznikne přenos do vyššího řádu a pro jeho reprezentaci je třeba bity dva:

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$1 + 1 = 10$$

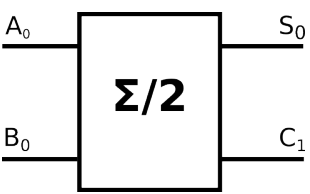
V případě $1 + 1 = 10$ vznikne přenos z řádu 2^0 do řádu 2^1. Tento přenos je třeba přičíst k binárním číslicím v řádu 2^1. Na základě těchto poznatků je třeba navrhnout logický obvod pro součet dvou jednobitových čísel, který bude obsahovat dva vstupy pro číslo A a pro číslo B a dva výstupy S pro součet odpovídající řádovému místu 2^0 a C (carry) pro přenos do vyššího řádu odpovídající řádovému místu 2^1. Tento obvod se nazývá **poloviční sčítačka** a je základem pro obvod sčítající vícebitová čísla. Pro sčítání vícebitových čísel je k poloviční sčítačce napojen ještě obvod, který se nazývá **úplná sčítačka**. Každá úplná sčítačka připojená k poloviční sčítačce rozšíří sčítaná čísla o jeden bit.

5.1.1 Poloviční sčítačka

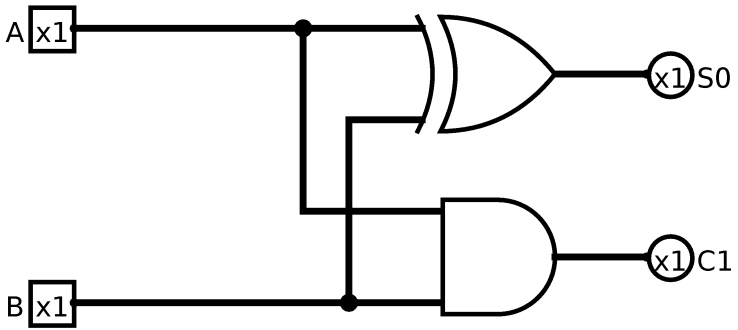
Poloviční sčítačka umožňuje sčítat dvě jedno-bitvá čísla a jejím výstupem je součet a přenos do vyššího řádu. Pravdivostní tabulka popisující činnost poloviční sčítačky vypadá takto:

$$\begin{bmatrix} N & A_0 & B_0 & C_1 & S_0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Schématická značka poloviční sčítačky:



Pro výstup C_1 je patrné, že je realizována jedním obvodem AND. Výstup S_0 je možné realizovat buď pomocí základních logických členů $S_0 = \overline{A_0} \cdot B_0 + A_0 \cdot \overline{B_0}$ a nebo pomocí logické funkce nonekvivalence $S_0 = XOR$

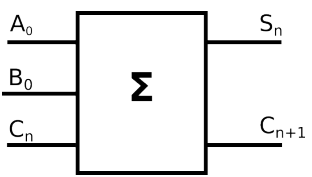


5.1.2 Úplná sčítačka

S použitím poloviční sčítačky lze sečíst pouze nejnižší řád dvou binárních čísel. Pro vyšší řádová místa se musí ke dvěma číslům A_n a B_n přičíst ještě přenos z nižšího řádu C_n. Teoreticky lze poloviční sčítačku nahradit úplnou sčítačkou, na kterou by se přivedlo na vstup C_0 konstantní hodnota nula. Pro praktické účely je tato obvodová realizace zbytečně složitá. Pravdivostní tabulka úplné sčítačky vypadá takto:

N	A_n	B_n	C_n	C_{n+1}	S_n
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

Schématická značka úplné sčítačky:



Z pravdivostní tabulky je patrné, že pokud alespoň dvě vstupní proměnné nabydou hodnoty jedna, musí být na výstupu C_{n+1} také hodnota jedna $(1 + 1 + 0 = 10)$.

Pokud pouze jedna nebo všechny vstupní proměnné nabydou hodnoty jedna, musí být na výstupu S_n také hodnota jedna $(1 + 0 + 0 = 1$ a $1 + 1 + 1 = 11)$. Z pravdivostní tabulky lze pro výstup S_n získat funkci:

$$S_n = (\overline{A_n} \cdot \overline{B_n} \cdot C_n) + (\overline{A_n} \cdot B_n \cdot \overline{C_n}) + (A_n \cdot \overline{B_n} \cdot \overline{C_n}) + (A_n \cdot B_n \cdot C_n)$$

$$S_n = C_n \cdot (\overline{A_n} \cdot \overline{B} + A_n \cdot B_n) + \overline{C_n} \cdot (\overline{A_n} \cdot B_n + A_n \overline{B_n})$$

$$S_n = C_n \cdot (\overline{A_n \oplus B_n}) + \overline{C_n} \cdot (A_n \oplus B_n)$$

$$S_n = \overline{C_n} + (A_n \oplus B_n) + C_n + \overline{(A_n \oplus B_n)}$$

$$S_n = (A_n \oplus B_n \oplus C_n) + (A_n \oplus B_n \oplus C_n)$$

$$S_n = A_n \oplus B_n \oplus C_n$$

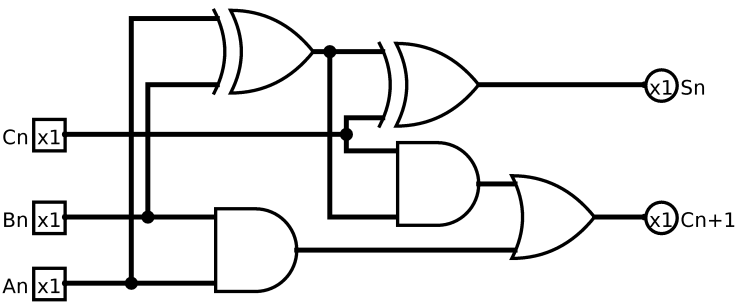
Pro výstup $C_n + 1$ lze z pravdivostní tabulky získat funkci:

$$C_{n+1} = (\overline{A_n} \cdot B_n \cdot C_n) + (A_n \cdot \overline{B_n} \cdot C_n) + (A_n \cdot B_n \cdot \overline{C_n}) + (A_n + B_n + C_n)$$

$$C_{n+1} = C_n \cdot (\overline{A_n} \cdot B_n + A_n \cdot \overline{B_n}) + A_n \cdot B_n \cdot (C_n + \overline{C_n})$$

$$C_{n+1} = C_n \cdot (A_n \oplus B) + A_n \cdot B_n$$

Chéma úplné sčítačky podle rovnic:

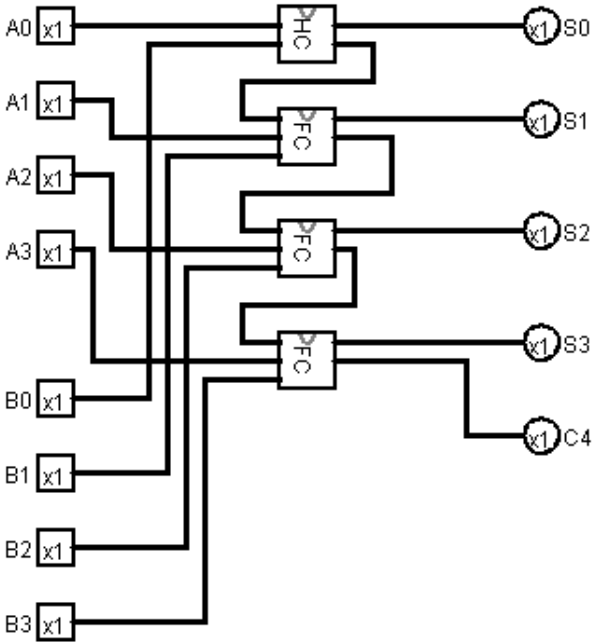


Vzhledem k tomu, že hodnoty proměnných přicházející na vstup úplné sčítačky mají stejnou váhu řádového místa, není třeba mezi vstupy této sčítačky rozlišovat. Hodnoty na výstupu jsou vždy závislé jen na tom kolik jedniček se na vstupech sčítačky vyskytne a ne na tom na kterých vstupech tyto jedničky jsou. Naproti tomu výstupy sčítačky

zaměnitelné nejsou, protože odpovídají hodnotám s různou vahou řádových míst.

5.1.3 Paralelní sčítačka

Paralelní sčítačky označují kombinační číslicový obvod, který vykonává sčítání nebo odčítání vícebitových binárních čísel. Paralení sčítačky se skládají z úplných sčítaček a jedné poloviční sčítačky (nebo jedné úplné sčítačky, které je na vstup C přivedena konstantní hodnota 0). V případě úplných sčítaček je výstup C_{n-out} přiveden na vstup C_{n-in} následující úplné sčítačky.



Paralelní sčítačka pracuje nad množinou **přirozených čísel**. Vstupem a výstupem paralelní sčítačky tak jsou binární čísla v **přímem binárním kódu**.

5.1.4 Rozšířená paralelní sčítačka

Rozšířená paralelní sčítačka je rozšířena o část, která umožňuje převést vstupní binární operandy do kódu dvojkového doplňku a realizovat tak současně operaci rozdíl. Rozšírená paralelní sčítačka tak pracuje nad množinou **celých čísel**. Vstupem a výstupem rozšírené paralelní sčítačky jsou tak čísla v **kódu dvojkovém doplňku**.

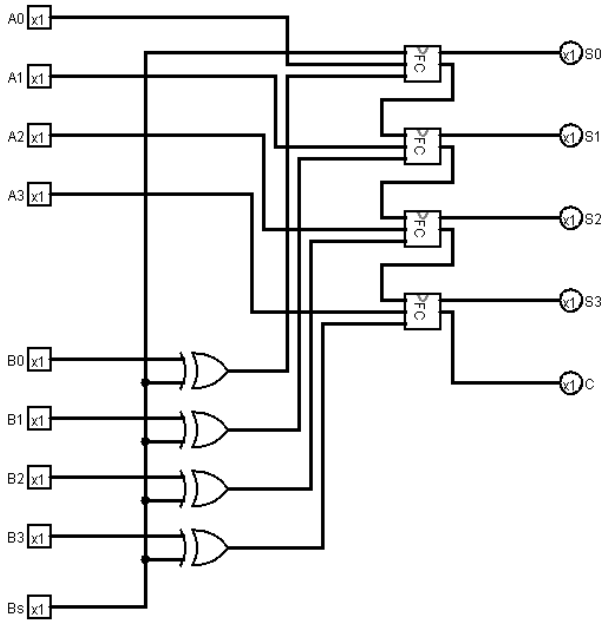
Rozšířená paralelní sčítačka disponuje řídícím vstupem s (sign - znaménko), kterým je možné sčítačku přepínat mezi režimi pro součet celích (znaménkových) čísel a přirozených (bezznaménkových) čísel. U rozšířené paralelní sčítačky bylo nutné nahradit na číselném řádu 2^0 poloviční sčítačku za úplnou sčítačku kvůli procesu převodu operandu B do dvojkového doplňku.

Rozšířená paralelní sčítačka funguje tak, že umožňuje pomocí řídícího vstupu s, invertovat bity druhého operandu a zároveň k součtu přidat hodnotu jedna. Díky tomu je druhý operand převeden do kódu dvojkového doplňku a logická sčítačka tak realizuje operaci rozdíl bez nutnosti syntézy nového obvodu.

$$A + (\overline{B} + 1) = A + (-B) = A - B$$

Na vstupy B_n jsou připojeny logická hradla XOR a jako řídící vstup s. V případě, že je hodnota vstupu s v logické

nule, pak se na vstup logické sčítačky dostane hodnota, která se nachází na vstupech B_n. V případě, že je vstup s v logické jedničce, pak hradla XOR fungují jako invertory, které znegují hodnotu vstupů B_n a zároveň je přivedena jednička na vstup úplné sčítačky na číselném řádu 2^0. V takovém případě dojde k převodu operandu B do kódu dvojkového doplňku a k odečtení jeho záporné hodnoty od hodnoty operandu A.



U programovatelných obvodů bývá očas potřeba odčítat dvě záporná čísla. To je možné buď s rozsáhlou úpravou obvodu pro logický součet/rozdíl, aby bylo možné převést v jednom kroku obě čísla do dvojkového doplňku a nebo převést nejprve první operand do dvojkového doplňku a uložit

jej do dočaného uložiště a následně od tohoto čísla odečíst druhé číslo.

5.1.5 Přetečení

Přetečení (overflow) je stav, kdy při aritmetické operaci součet/rozdíl nelze výsledek zobrazit pomocí platného počtu bitů ve kterém byla zobrazena jednotlivá vstupní čísla. Takový stav nastává při sčítání dvou větších čísel a nebo při rozdílu dvou čísel. Jedná se o chybu při sčítání dvou čísel v kódu dvojkového doplňku, na kterou je nutné vhodným způsobem reagovat.

Kód dvojkového doplňku umožňuje reprezentovat binární čísla v množině celých čísel. MSB binárních čísel v reprezentaci dvojkového doplňku se říká **znaménkové bity**. V případě kladných čísel mají celá čísla na pozici MSB vždy hodnotu *nula* a záporná čísla vždy hodnotu *jedna*:

$$01100010_{(2)} = 98_{(10)}$$

$$10011110_{(2)} = -98_{(10)}$$

Při přetečení je indikován stav (situace) kdy je nutné rozšířit řádové pozice pro reprezentaci znaménkového bitu. Přenos do vyššího řádu (carry) na pozici $n + 1$, kde n je

počet platných vstupních bitů číselných operandů operace, pak udává znaménko výsledné číselné hodnoty. Příznak přetečení a přenosu do řádu $n + 1$ jsou zaznamenávány v aritmeticko-logických jednotkách pro speciální použití při dalších operacích.

Přetečení se využívá při výpočetech hodnot, které nemohou být reprezentovány počtem bitů se kterými pracuje logická sčítačka. Taková čísla jsou sčítána v několika krocích.

5.2 Logická násobička

Logická nísobička může být realizována buď pomocí kombinační logiky a nebo pomocí sekvenční logiky. Tato dvě řešení se vzájemně liší rychlostí provádění výpočtu a složitostí obvodové realizace.

Kombinační logická násobička pracuje stejně jako při ručním násobení. To znamená, že každý řád jednoho čísla vynásobí každý řád druhého čísla a jednotlivé mezivýsledky jsou vzájemně sečteny.

Sekvenční logická násobička pracuje na principu postupného výpočtu a ukládání mezivýsledků do registrů. Oproti kombinační logické násobičce není tak rychlá (výpočet není proveden v jednom kroku), ale její obvodová realizace je jednodušší.

Podle vstupních operandů se rozlišují násobičky v **pevné řádové čárce** a v **plovoucí řádové čárce**. Při násobení dvou znaménkových čísel v dvojkové soustavě je možné vynásobit nejprve absolutní hodnoty čísel a do výsledku přidat znaménko (polaritu), a nebo násobit přímo čísla se znaménkem.

V případě logických násobiček nastvává problem při násobení velkých čísel. Při násobení x-bytového násobence s y-bitovým násobitele může vzniknou k-bitový součin, který může být až $k = x + y$ bitový. Tím nastává vysoké riziko přetečení výsledku.

Protože se logický součin dvou vstupních logických proměnných chová jako klasický součin dvou bitových hodnot slouží jako násobící člen při násobení každého číselného řádu jednoho čísla každého číselného řádu druhého čísla ve všech typech logických násobiček.

5.2.1 Kombinační logická násobička

Pro součet jednotlivých mezivýsledků jsou následně použity poloviční a poloviční sčítačky.

5.3 Modulo

5.4 Základní parametry elektronických logických obvodů

Softwarově simulovaná logická hradla mají tu vlastnost, že se chovají jako ideální logický prvek, který nerespektuje fyzikální vlastnosti reálných obvodů. Každé logické hradlo má nějaké parametry, které je nutné v praxi respektovat:

• **Vstupní odběr** I_{vst} - proud odebíraný jedním vstupem logického hradla.

• **Zatížitelnost vstupů** N - udává maximální možný počet vstupů, které lze napojit na výstup daného hradla, aby byly zaručeny napěťové úrovně pro korektní rozeznání logických úrovní, hodnoty dob náběhu sestuou a šíření signálu ze vstupu na výstup. Zatížitelnost se udává jako počet hradel, které lze na výstup připojit.

• **Doba poklesu** t_p **a doba náběhu** t_n - udávají čas potřebný k poklesu výstupního signálu na výstupu z 90% na 10% a čas potřebný k náběhu výstupního signálu z 10% na 90%. Doba poklesu a náběhu říká jak rychle hradlo dokáže přepínat na výstupu mezi logickými stavy.

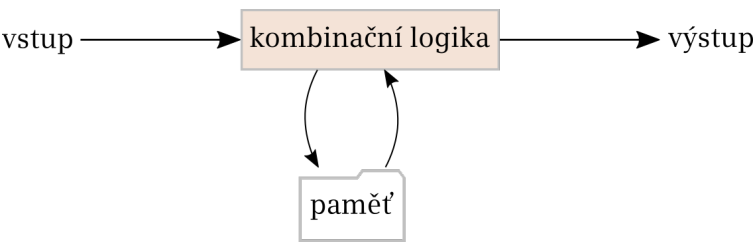
• **Doba šíření signálu** $t_š$ - též zpoždění signálu je doba, která udává čas potřebný pro reakci hradla na hodnotu na vstupu, aby se změny projevyly na jeho výstupu.

• **Šumová imunita** - Vlivem rušení může dojít k indukování parazitního napětí na vstupech logického hradla. Šumová imunita je taková velikost napětí, která může vniknout na

vstup hradla, aniž by došlo k reakci na výstupu na tento parazitní signál. Šumová imunita může být různá pro log 1 a pro log 0. Pro log 0 je šumová imunita dána rozdílem nejnižšího a nejvyššího napětí, které dané hradlo registruje jako log 0. Pro log 1 je šumová imunita dána jako rozdíl nejnižší a nejvyšší hodnoty vstupního napětí, které dané hradlo registruje jako log 1.

Kapitola 6
Sekvenční logické obvody

Sekvence je chápána jako časová posloupnost vstupních hodnot a vnitřních stavů. V případě sekvenčních logických obvodů závisí okamžitá hodnota výstupů ne jen na aktuálních stavech vstupů, ale také na vnitřních stavech sekvenčního obvodu. Vnitřní stav sekvenčního obvodu je ovlivněn hodnotami předchozích vstupů. Sekvenční logické obvody obsahují paměťové prvky, které uchovávají předchozí stav obvodu a určitým způsobem na něj reagují.



Sekvenční logické obvody lze popsat dvěma rovnicemi, rovnicí pro výstupy:

$$Y_i = f(\{X_j\}, \{Q_p^t\})$$

a rovnicí pro následující vnitřní stav (přechodová funkce):

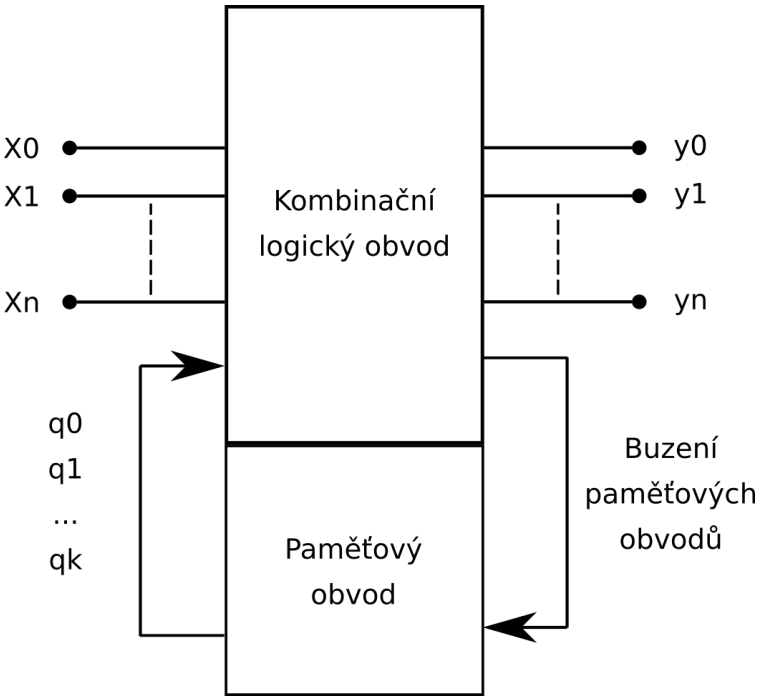
$$Q_p^{t+1} = g(\{X_j\}, \{Q_p^t\})$$

kde Y_i vyjadřuje množinu výstupů sekvenčního obvodu $Y_i = (y_0, y_1, ...y_m)$, X_j značí množinu vstupů $X_j = (x_0, x_1, ...x_n)$ a Q_p značí množinu vnitřních stavů $Q_p = (q_0, q_1, ...q_k)$. Logické proměnné $q_0, q_1, ...q_k$ se nazývají **vnitřní proměnné** a jsou tvořeny jednotlivými výstupy klopných obvodů. Množina všech výstupů klopných obvodů v čase t tvoří vnitřní stav sekvenčního logického obvodu. Index t rozlišuje současný vnitřní stav Q_p^t a $t+1$ označuje následující vnitřní stav sekvenčního ovbodu Q_p^{t+1}

Celý popis výkonu sekvenčního obvodu lze popsat pomocí schématu:

$$X_j \rightarrow Q_p \rightarrow f(X_j, Q_p) \rightarrow Y_i$$

Tato schéma říká, že na základě vstupních proměnných je pomocí přechodové funkce vygenerována množina vnitřních stavů a pomocí obvodové funkce $f(X_j, Q_p)$ je vygenerována množina výstupních hodnot.



Z obecného blokového schématu kombinačního logického obvodu je patrné, že má kombinační (řídící) část, která generuje hodnoty výstupů $y_0, y_1, ..., y_m$ a dále budící sig-

nály klopných obvodů. Druhá část sekvenčního obvodu je tvořena klopnými obvody, které na základě budících signálů generuje vnitřní proměnné sekvenčního obvodu. Vstupními signály kombinační části obvodu jsou $x_0, x_1, ..., x_n$ a vnitřní proměnné $q_0, q_1, ..., q_n$ z výstupů paměťové čáti.

Sekvenční logický obvod, který je plně popsán rovnicenim pro výstupy a pro následující vnitřní stavy se nazývají **Mealyho sekvenční obvody** (automat). V případě, že výstup sekvenčního obvodu je plně určen jeho vnitřním stavem, nazývá se takový obvod **Moorův sekvenční obvod** (automat).

6.1 Synchronní a asynchronní sekvenční obvody

Sekvenční logické obvody lze rozdělit:

• **Synchronní** - synchronní sekvenční obvody jsou synchronizovány samostatnými taktovacími signály (čítače, posuvné registry, ...).

• **Asynchronní** - asynchronní sekvenční obvody generují svůj výstup okamžitě a zpoždění nastává pouze šířením signálu obvodem (klopné obvody).

V logických obvodech vznikají mezi jednotlivými stupni logických členů zpoždění. Toto zpoždění výstupního signálu

vůči vstupnímu signálu je dáno způsobem realizace logického obvodu. Spínací součástky nemají čas nutný k sepnutí a rozepnutí nulový, proto musí nutně dojít k časovému posunutí reakce výstupního signálu vůči vstupnímu signálu. Po ustálení hodnot na vstupech tedy nějakou dobu trvá, něž se na výstupech objevý správné hodnoty. Toto zpoždění tedy závisí na zpoždění jednotlivých elementárních členů. Kvůli tomu v sekvenčních obvodech vzniká možnot výskytu dočasných nesprávných signálů uvnitř obvodu i na výstupech obvodu. Tyto nesprávné dočasné vnitřní stavy se nazývají **hazardy**. Návrh asynchronního obvodu musí být proveden tak, aby k těmto hazardům nedocházelo. To se provádí buď přepracováním vnitřního uspořádání zapojení obvodu a nebo dodáním taktovacího signálu, který jednotlivé členy obvodu vzájemně sinchronizuje.

6.2 Syntéza sekvenčních logických obvodů

Aby bylo možné syntetizovat libovolný sekvenční logické obvody je nejprve nutné je nějakým způsobem popsat. Sekvenční logické obvody lze popsat několika různými způsoby. V praxi je většinou pro přehlednost využívána kombinatě více způsobo:

- Pravdivostní tabulka
- Časový diagram

- Diagran přechododů (stavový diagram)
- Vývojová tabulka
- Tabulka vnitřních stavů

Pravdivostní tabulka je nejjednodušší možností jak popsat sekvenční logický obvod. Narozdíl od kombinačních obvodů je nutné v pravdivostní tabulce vypsat seznam vnitřních stavů Q_y^p.

$$\begin{bmatrix} N & S & R & Q^{t+1} & \overline{Q^{t+1}} \\ 0 & 0 & 0 & Q^t & \overline{Q^t} \\ 1 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & (1) & (1) \end{bmatrix}$$

Časový diagram je tvořen časovým průbehem stavů vstupů a výstupů logického systému. Časový diagram obsahuje časový průběh všech kombinací vstupních hodnot a jejich reakcí na výstupu. Díky tomu je možné úplně popsat chování daného obvodu. Příklad časového diakgramu pro funkci NOR:

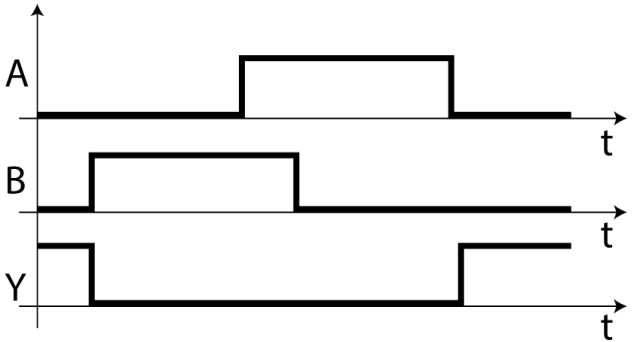
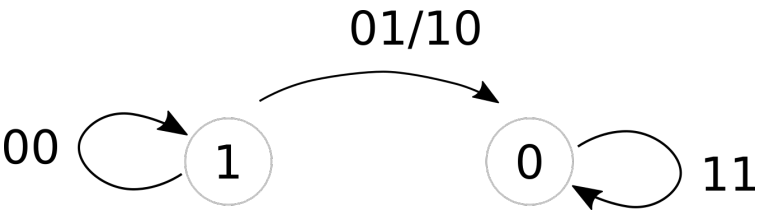


Diagram přechodů je grafický nástroj, který umožňuje zobrazit změnu vnitřních stavů a hodnot výstupů na základě vstupních chodnot. Diabgram přechodů se skládá z kroužků, které definují aktuální stav systému (vnitřní stavy a hodnoty na výstupu). Jednotlivé stavy jsou propojeny orientovanými úsečkami (křivkami), které definují přechod mezi jednotlivými stavy. Přechod mezi dvěma stavi je reakcí na událost. Událost definuje změnu vstupních hodnot logického systému. Vstupní hodnoty logického systému jsou zobrazeny u orientované úsečky přechodu a definují typ události, která inicializovala daný přechod. Diagram přechodů může zobrazovat pouze vnitřní stavy, nebo pouze stavy výstupů a nebo oboje dohromady. V případě, že

daný přechod platí pro dvě nebo více vstupních kombinací, jsou vzájemně odděleny lomítkem. Statový diagram funkce NOR:



6.3 Zpětná vazba

Zpětná vazba logických obvodů je tvořena přivedením výstupního signálu opět na řídící vstup daného obvodu. Díky tomu je ovlivněn nový výstup obvodu. Vstup, na který je přiveden výstup ze stejného logického obvodu se nazývá **zpětnovazební vstup**, Výstup, který je přiveden na zpětnovazební vstup se nazývá **zpětnovazební výstup**. Za **zpětnovazební logický obvod** je považován každý logický obvod, který má alespoň jeden řídící vstup a alespoň jeden zpětnovazební vstup. Speciální skupina zpětnovazebních logických obvodů se nazývá klopné obvody, které tvoří základ sekvekvenčních logických obvodů a dělí se do tří skupin:

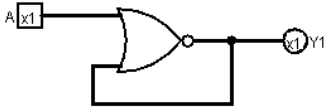
• **Bistabilní klopný obvod** - klopný obvod, který se může nacházet ve dvou stabilních stavech. Jejich stav se mění skokem v druhý v případě, že je na jejich vstup přiveden budící signál. Bistabilní klopné obvody se používají jako paměťové prvky.

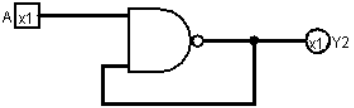
• **Monostabilní klopný obvod** - klopný obvod, který má jeden stabilní stav. Po přivedení vstupního řídícího signálu se přepne do nestabilního stavu ze kterého se sám po nějaké době sám vrátí do stabilního stavu. Monostabilní klopné obvody se používají jako zpožďovací prvek.

• **Astabilní klopný obvod** - nemají žádný stabilní stav a jeho výstupní signál tvoří periodické impulsy. Jejich frekvence a amplituda závisí na parametrech obvodu. Používá se jako generátor sinchronizačního signálu.

6.3.1 Zpětná vazba a oscilace

Zpětná vazba ale může způsobit stav, který se nazývá **oscilace sekvenčního obvodu**. Oscilace sekvenčního obvodu je způsobena nedefinovaným stavem obvodu, kdy se cyklicky přepíná mezy logickými stavy zpětnovazebního vstupu (ovlivňuje sám sebe). Tak dojde k situaci, že je-li na zpětnovazebním výstupu log 1, dojde řídícím vstupem k přepnutí výstupu na hodnotu log 0, která ale na řídícím zpětnovazebním vstupu způsobí přepnutí na log 1.





$$\begin{bmatrix} N & A & Y_1 & Y_2 & \overline{A+Y_1} & \overline{A \cdot Y_2} \\ 0 & 0 & 0/1 & 1 & 0/1 & 1 \\ 1 & 1 & 0 & 0/1 & 0 & 0/1 \end{bmatrix}$$

Z pravdivostní tabulky je patrné, že oscilační obvod s výstupem Y_1 přejde ze stabilního stavu do oscilačního při nastavní řídícího vstupu A do logické hodnoty 0. Naproti tomu oscilační obvod s výstupem Y_2 přejde ze stabilního stavu do oscilačního při nastavní řídícího vstupu A do logické hodnoty 1.

Zpětnovazební, astabilní klopné obvody mají své uplatnění při generování periodického signálu, který může být využit jako synchronizační, časovací signál.

Každý oscilační sekvenční obvod se může nacházet v jednom ze dvou stavů- **stabilní stav** a **oscilační** (nestabilní) **stav**. Na základě řídícího vstupu je možné přecházet z jednoho stavu do druhého.

6.3.2 Popis zpětnovazebních logických obvodů

Při popisu zpětnovazebních logických obvodů je důležité jakým způsobem zpětnovazební vstup obvodu opětovně ovlivňuje jeho výstup, respektive chování logického obvodu.

6.4 Klopné obvody

Klopný obvod je nejjednodušší sekvenční číslicový obvod, který funguje jako paměťový člen. Úkolem klopných obvodů je zaznamenat přítomnost přechdoné informace a uchovat tento stav i poté co informace zmizí. Klopní obvod, respektive jeho paměťový efekt je tvořen spojením dvou (nebo více) oscilačních obvodů, kdy jeden přepíná druhý a naopak.

Kapitola 7
Konečný automat

Konečný automat je výpočetní model primitivního počítače, který se skládá z několika stavů a z několika přechodů a který dokáže přijmout nebo zamítnout předané slovo na vstupu systému. Statové automaty mají široké spektrum použití od konstrukce jednoduchých jednoúčelových logických systémů, základ pro složitější programovatelné obvody až po softwarové algoritmy.