Vysoká škola báňská - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

LOGICKÉ OBVODY

pro kombinované a distanční studium

Zdeněk Diviš
Zdeňka Chmelíková
Iva Petříková

Ostrava 2003

© Prof. Ing. Zdeněk Diviš, CSc., Ing. Zdeňka Chmelíková, Ing. Iva Petříková, 2003

Fakulta elektrotechniky a informatiky

VŠB – Technická univerzita Ostrava

OBSAH LOGICKÝCH OBVODŮ

1 ČÍSELNÉ SOUSTAVY 7

1.1 Převod kladných celých čísel 9

1.1.1 Metoda postupného odečítání vah 9

1.1.2 Metoda postupného dělení základem 12

1.2 Převod čísel kladných desetinných 15

1.2.1 Metoda postupného odečítání vah 15

1.2.2 Metoda postupného násobení základem 16

1.3 Vztah mezi binární, oktální a hexadecimální soustavou 18

2 BOOLEOVA ALGEBRA 21

2.1 Booleovské funkce 31

2.2 Způsoby zápisu booleovských funkcí 35

2.2.1 Tabulkové, vektorové a číselné zápisy 35

2.2.2 Zápis logické funkce přiřazením výrazu 40

2.3 Minimalizace Booleovských funkcí 44

2.3.1 Algebraická minimalizace 45

2.3.2 Minimalizace pomocí Karnaughových map 48

2.3.3 Minimalizace metodou Mc-Cluskey 55

3 LOGICKÉ KOMBINAČNÍ OBVODY 62

3.1 Návrh logických kombinačních obvodů 64

3.2 Číslicové integrované obvody 65

3.3 Realizace logických kombinačních obvodů pomocí vícevstupových hradel NAND 68

3.4 Realizace pomocí dvouvstupových hradel NAND 69

3.5 Realizace pomocí dvouvstupových hradel NOR 70

3.6 Realizace pomocí hradla AND-OR-INVERT 71

3.7 Realizace pomocí hradel NAND s otevřeným kolektorem ... 72

3.8 Realizace kombinačních obvodů pomocí paměťových prvků ... 73

3.9 Hazardní stavy ... 85

3.10 Ošetření vstupních signálů ... 93

4 LOGICKÉ SEKVENČNÍ OBVODY ... 96

4.1 Analýza logických sekvenčních obvodů ... 104

4.1.1 Analýza sekvenčních obvodů bez paměťového členů ... 105

4.1.2 Analýza sekvenčních obvodů s paměťovými členy ... 107

4.2 Návrh synchronních sekvenčních obvodů ... 134

4.3 Standardní zapojení logických obvodů ... 153

4.4 Návrh generátoru binárních posloupností ... 166

5 ARITMETICKO-LOGICKÁ JEDNOTKA ... 176

5.1 Způsob zobrazování celých čísel ... 177

5.1.1 Vyjádření záporných čísel jednotkovým doplňkem ... 177

5.1.2 Vyjádření záporných čísel ve dvojkovém doplňku ... 178

5.2 Sčítání ... 179

5.3 Odčítání ... 185

5.3.1 Odčítání s jednotkovým doplňkem ... 185

5.3.2 Odčítání s dvojkovým doplňkem ... 187

5.4 Násobení ... 189

5.5 Dělení ... 191

5.6 Porovnávání ... 192

5.7 Zobrazování čísel v pohyblivé řádové čárce ... 195

ÚVODEM LOGICKÝCH OBVODŮ

Tyto texty jsou určeny pro studenty prvního ročníku kombinovaného studia bakalářského studijního programu Informační technologie a pro studenty 2. ročníku bakalářského studijního programu Elektrotechnika, sdělovací a výpočetní technika. Svojí strukturou odpovídají textům určeným pro distanční vzdělávání. Orientaci v textu má usnadnit jednotná struktura kapitol spolu s používáním odpovídajících symbolů Kromě teoretických základů z oblasti logických systémů je zde množství podrobně komentovaných řešených příkladů. Pochopení dané problematiky si může student vyzkoušet na mnoha neřešených příkladech, které jsou pravidelně do textu zařazovány. Texty jsou napsány tak, aby byly srozumitelné i pro studenty, kteří se dosud s problematikou logických obvodů nesetkali a nevyžadují se žádné znalosti z tohoto oboru.

CÍLE PŘEDMĚTU LOGICKÉ OBVODY

Po úspěšném a aktivním absolvování předmětu LOGICKÉ OBVODY

- budete umět používat další číselné soustavy (binární, oktální a hexadecimální),
- budete umět navrhovat kombinační obvody pomocí různých typů hradel nebo pomocí paměťových prvků,
- budete umět analyzovat zapojení s logickými obvody a navrhovat sekvenční obvody, posuvné registry, čítače a generátory binárních posloupností,
- budete umět zapisovat a číst data z různých typů paměťových prvků,
- získáte základní znalosti o nejběžnějších integrovaných obvodech,
- budete schopni realizovat zapojení kombinačních a sekvenčních logických obvodů.

PRŮVODCE STUDIEM 1

V průběhu semestru budou Vaše znalosti ověřovány formou dvou samostatných prací a formou testu. První samostatná práce bude představovat návrh kombinačního obvodu, druhá návrh sekvenčního obvodu. Zadání je v obou případech jednotné a je uvedeno v těchto textech. Realizované funkce má však každý student jiné a obdrží je na tutoriálu nebo formou e-mailu. V průvodci studiem Vám poradíme, kdy byste měli být schopni určitou část zadání vypracovat. Správnost Vašeho návrhu si ověříte sami při realizaci.

SAMOSTATNÁ PRÁCE 1

a) Ze zadané závislosti mezi 6 vstupními a 4 výstupními proměnnými vytvořte pravdivostní tabulku, vyjádřete funkce v součtovém tvaru a pomocí Booleovy algebry je zjednodušte.

b) Minimalizujte zadané funkce pomocí metody Mc-Cluskey.

c) Podle získaných rovnic dle bodu a) a dle bodu b) nakreslete síť pro realizaci funkcí pomocí hradel NAND, resp. NOR.

1 ČÍSELNÉ SOUSTAVY

ČAS POTŘEBNÝ KE STUDIU

Předpokládaný čas k prostudování kapitoly Číselné soustavy je **5 hodin**.

RYCHLÝ NÁHLED DO PROBLEMATIKY KAPITOLY ČÍSELNÝCH SOUSTAV

Realizace logických systémů mechanickými prvky využívala matematickou soustavu se základem 10 a všechny funkce, například sčítání, byly realizovány v soustavě s tímto základem.

Realizace pomocí elektronických prvků, počínaje elektronkami, přes diody, tranzistory, integrované obvody až po mikropočítačové obvody, začala logické systémy chápat jako číslicové systémy, a protože logická hodnota v číslicových systémech se nazývá bit (Binary Digit – dvojková číslice), začala se k popisu vektorů proměnných používat dvojková soustava se symboly 0, 1, resp.(0, I). K vyjádření velikosti vektorů se využívá termín *n*-bitové slovo, kde *n* znamená počet proměnných.

Délka zápisu stavu, například vstupního vektoru ve dvojkové soustavě, dále vedla k přehlednějšímu zapisování a čtení informace, a to v osmičkové a později šestnáctkové soustavě.

CÍLE KAPITOLY ČÍSELNÉ SOUSTAVY

- Budete umět zapsat čísla v různých číselných soustavách,
- získáte ucelený přehled o problematice číselných soustav a o používaných metodách pro převody čísel mezi soustavami,
- budete schopni vybrat a použít nejvhodnější metodu pro převod čísel a v následujících kapitolách budete schopni vektory logických proměnných v různých číselných soustavách zapsat.

KLÍČOVÁ SLOVA KAPITOLY ČÍSELNÉ SOUSTAVY

Binární soustava, oktální soustava, hexadecimální soustava, dekadická soustava, číselná soustava, koeficient, váha, základ, bit, polynom.

Osmičková soustava používá číselné hodnoty 0, 1, 2, 3, 4, 5, 6, 7 a šestnáctková soustava symboly 0, 1, …, A, B, C, D, E, F, kde symboly A, B, C, D, E, F reprezentují desítkové symboly 10, 11, 12, 13, 14 a 15. Stejně jako v desítkové soustavě je možné realizovat matematické operace i v jiných soustavách, viz. kapitola 9.

Jestliže použijeme k zápisu stavů v uvedených soustavách stejných matematických zvyklostí jako v desítkové soustavě, můžeme obecně každé číslo N o základě B psát jako součet součinů

$$N_B = a_n \cdot B^n + a_{n-1} \cdot B^{n-1} + \ldots + a_1 \cdot B^1 + a_0 \cdot B_0 + b_{-1} \cdot B^{-1} + \ldots + b_{-m} \cdot B^{-m} \qquad (1\text{-}1)$$

kde koeficienty a_n, a_{n-1}, ... b_{-m} představují symboly ze soustavy o základu B a mocnina základu představuje jeho váhu. Při běžném zápisu čísel se však tato váha neuvádí a píše se

$$N_B = a_n a_{n-1} \ldots a_1 a_0 , b_{-1} \ldots b_{-m} \qquad (1\text{-}2)$$

Je zřejmé, že pomocí výrazu *(1-2)* nelze vyjádřit libovolná reálná čísla, ale pouze čísla kladná a racionální. Přesnost tohoto zápisu je potom dána váhou nejnižšího členu výrazu, v tomto případě B^{-m} . Jinými slovy to znamená, že desetinná čísla nelze v jiné soustavě vyjádřit vždy se stejnou přesností. Například číslo $N = 2975{,}23_{10}$ lze zapsat jako

$$N = 2 \cdot 10^3 + 9 \cdot 10^2 + 7 \cdot 10^1 + 5 \cdot 10^0 + 2 \cdot 10^{-1} + 3 \cdot 10^{-2} \qquad (1\text{-}3)$$

přičemž přesnost zápisu je dána členem s váhou 10^{-2} , což je 0,01. Stejně tomu je i v jiných soustavách.

K zápisu logických hodnot není zapotřebí zápisu v desetinném tvaru, ale plně postačuje vyjádření ve tvaru celého kladného čísla. Výraz *(1-1)* se potom redukuje na tvar

$$N_B = a_n \cdot B^n + a_{n-1} \cdot B^{n-1} + \ldots + a_1 \cdot B^1 + a_0 \cdot B^0 \qquad (1\text{-}4)$$

s přesností ± 1 . Pokud číslicový systém pracuje s desetinným číslem, potom je desetinná čárka pevně definována a nezobrazuje se speciálním symbolem. Rozsah takto zobrazovaných čísel N je v desítkové soustavě v intervalu 0 - 1, $N \in (0,1)$. Číslu N_{10} = 0,5 zobrazenému ve dvojkové soustavě ve 4 bitech odpovídá tvar I000, což je tvar stejný jako pro celé kladné číslo N_{10} =16. Je proto zřejmé, že forma zobrazení je dána vzájemnou dohodou.

ÚLOHY K ŘEŠENÍ 1-1

Pomocí sčítání násobků mocnin převeďte

a) z binární do desítkové soustavy číslo N_2 = 110111100
b) z oktální do desítkové číslo N_8 = 1620
c) z hexadecimální do binární soustavy číslo N_{16} = 3E7

1.1 Převod kladných celých čísel

Jestliže máme číslo N o základu B_1 a chceme ho převést na číslo o základu B_2 a tato čísla jsou celá kladná, pak pro převod použijeme následující nejběžnější metody:

- metoda postupného odečítání vah
- metoda postupného dělení základem

1.1.1 Metoda postupného odečítání vah

Tato metoda vychází ze vztahu

$$N_{B1} = a_n \cdot B_1^n + a_{n-1} \cdot B_1^{n-1} + \ldots + a_1 \cdot B_1^1 + a_0 \cdot B_1^0 \qquad (1\text{-}5)$$

Metoda spočívá v hledání koeficientů $a_n, a_{n-1}, \ldots a_1, a_0$ postupným odčítáním zmenšujících se vah $B_2^n, B_2^{n-1}, \ldots B_2^1, B_2^0$. Je vlastně hledána mocnina se základem B_2 menší nebo rovna zbytku převáděného čísla. Algoritmus výpočtu na počátku předpokládá hodnoty všech koeficientů $a_n, a_{n-1}, \ldots a_1, a_0 = 0$. Od čísla N_{B1} se odečte nejbližší nižší váha B_2^n a dostaneme zbytek ${}^1N_{B1}$.

$${}^1N_{B1} = N_{B1} - B_2^n \qquad (1\text{-}6)$$

Když ${}^1N_{B1} \geq B_2^n$, potom $a_n = a_n + 1$ a opětovně odčítáme váhu B_2^n. V opačném případě tj. ${}^1N_{B1} < B_2^n$ již váha B_2^n nemůže být ve zbytku obsažena (nepřeváděli bychom již číslo kladné) a přejdeme na hledání koeficientu a_{n-1}. Platí, že ${}^2N_{B1} = {}^1N_{B1} - B_2^{n-1}$ a opětovně $a_{n-1} = a_{n-1} + 1$, když ${}^2N_{B1} \geq B_2^{n-1}$ a nebo

přejdeme na odčítání nižší váhy, když ${}^{2}N_{B1} < B_2^{n-1}$. Algoritmus končí stanovením koeficientu a_0.

Z uvedeného vyplývá, že koeficient a_n představuje kolikrát od čísla N_{B1} mohu odečíst váhu B_2^n, aby výsledek byl kladný nebo se rovnal nule. Analogicky toto platí i pro zbývající koeficienty.

ŘEŠENÝ PŘÍKLAD

Metodou postupného odečítání vah převeďte číslo $N_{10} = 186_{10}$ do binární a oktální soustavy.

Řešení příkladu

Převod do binární soustavy

Koeficienty $a_n, a_{n-1}, \ldots a_1, a_0$ mohou nabývat pouze hodnoty 0 nebo 1 a odpovídající váhy mají hodnotu:

Váha	Rozdíl	Koeficient a_n
$2^8 = 256$	186 - 256 = -70	$a_8 = 0$
$2^7 = 128$	186 - 128 = 58	$a_7 = 1$
$2^6 = 64$	58 - 64 = -6	$a_6 = 0$
$2^5 = 32$	58 - 32 = 26	$a_5 = 1$
$2^4 = 16$	26 - 16 = 10	$a_4 = 1$
$2^3 = 8$	10 - 8 = 2	$a_3 = 1$
$2^2 = 4$	2 - 4 = -2	$a_2 = 0$
$2^1 = 2$	2 - 2 = 0	$a_1 = 1$
$2^0 = 1$	0 - 1 = -1	$a_0 = 0$

Z tabulky vyplývá, že $N_{B2} = 10111010_2$.

Převod do oktální soustavy.

Koeficient a_n může nabývat hodnot od 0 do 7.

Váha	Rozdíl	Koeficient a_n
$8^3 = 512$	186 - 512 = -326	$a_3 = 0$
$8^2 = 64$	186 - 64 =122	$a_2 = 1$
	122 - 64 =58	$a_2 = 2$
$8^1 = 8$	58 - 8 = 50	$a_1 = 1$
	50 - 8 = 42	$a_1 = 2$
	42 - 8 = 34	$a_1 = 3$
	34 - 8 = 26	$a_1 = 4$
	26 - 8 = 18	$a_1 = 5$
	18 - 8 = 10	$a_1 = 6$
	10 - 8 = 2	$a_1 = 7$
$8^0 = 1$	2 - 1 = 1	$a_1 = 1$
	1 - 1 =0	$a_0 = 2$

Můžeme psát, že $N_{B2} = 272_8$.

*

ÚLOHY K ŘEŠENÍ 1-2

Pomocí metody postupného odečítání vah převeďte z desítkové soustavy

a) do binární soustavy ${}^1N_{10} = 458$, ${}^2N_{10} = 133$ a ${}^3N_{10} = 95$,

b) do oktální soustavy ${}^1N_{10} = 912$, ${}^2N_{10} = 256$ a ${}^3N_{10} = 612$

c) do hexadecimální soustavy ${}^1N_{10} = 1112$, ${}^2N_{10} = 847$ a ${}^3N_{10} = 412$

1.1.2 Metoda postupného dělení základem

Předpokládáme, že máme číslo N o základu B_1 a chceme ho vyjádřit jako číslo o základu B_2. Odvození metod převodu vychází ze zápisu čísla v novém základu

$$N_{B2} = a_n \cdot B_2^n + a_{n-1} \cdot B_2^{n-1} + \ldots + a_1 \cdot B_2^1 + a_0 \cdot B_2^0 \qquad (1\text{-}7)$$

Jestliže tento výraz budeme dělit základem B_2, výsledkem bude podíl P^0 a zbytek Z^0, pro který bude platit $Z^1 > B_2$. Můžeme proto psát, že

$$N_{B2} = \left[a_n \cdot B_2^{n-1} + a_{n-1} \cdot B_2^{n-2} + \ldots + a_1 \cdot B_2^0\right] \cdot B_2^1 + a_0 \cdot B_2^0 \qquad (1\text{-}8)$$

resp.

$$N_{B2} = P^0 \cdot B_2^1 + Z^0 \qquad (1\text{-}9)$$

přičemž je zřejmé, že platí

$$Z^0 = a_0 \cdot B_2^0 = a_0 \qquad (1\text{-}10)$$

a tedy zbytek Z^0 představuje přímo koeficient a_0. Pro polynom P^0 platí

$$P^0 = a_n \cdot B_2^{n-1} + a_{n-1} \cdot B_2^{n-2} + \ldots + a_1 \cdot B_2^0 \qquad (1\text{-}11)$$

Ke stanovení koeficientu a_1 vydělíme polynom P^0 základem a dostaneme

$$P^0 = \left[a_n \cdot B_2^{n-2} + a_{n-1} \cdot B_2^{n-3} + \ldots + a_2 \cdot B_2^0\right] \cdot B_2^1 + a_1 \cdot B_2^0 \qquad (1\text{-}12)$$

odkud

$$P^1 = a_n \cdot B_2^{n-2} + a_{n-1} \cdot B_2^{n-3} + \ldots + a_2 \cdot B_2^0 \qquad (1\text{-}13)$$

$$Z^1 = a_1$$

Dalším dělením polynomu P^1 základem B_2 získáme postupně koeficienty $a_2, a_3, \ldots$ hledaného čísla v pořadí od nejnižší k nejvyšší váze.

ŘEŠENÝ PŘÍKLAD

Metodou postupného dělení základem převeďte číslo 1739_{10} do hexadecimální soustavy.

Řešení příkladu

Dílčí podíl P^n	Zbytek Z^n	Koeficient a_n
1739 : 16 = 108	11	a_0 = B
108 : 16 = 6	12	a_1 = C
6 : 16 = 0	6	a_2 = 6

Z tabulky je zřejmé, že číslo $1739_{10} = 6CB_{16}$.

*

ŘEŠENÝ PŘÍKLAD

Metodou postupného dělení základem vyjádřete číslo $N = 1653_8$ v hexadecimálním základě.

Řešení příkladu

Protože zadané číslo není v desítkové soustavě, je nutné si uvědomit, že převod se může uskutečnit buď:

- přímým převodem v osmičkové soustavě,
- převodem čísla N do desítkové soustavy pomocí sčítání mocnin a následným převodem z desítkové soustavy.

Přímý převod v osmičkové soustavě. Základ nové soustavy vyjádříme v soustavě, ve které budeme převod realizovat, tj. 20_8.

Dílčí podíl P^n	Zbytek Z^n	Koeficient a_n
$1653_8 : 20_8 = 72_8$	13	$a_0 = \mathrm{B}$
$72_8 : 20_8 = 3_8$	12	$a_1 = \mathrm{A}$
$3_8 : 20_8 = 0_8$	3	$a_2 = 3$

Na základě výpočtu lze psát, že číslo $N = 1653_8 = 3\mathrm{AB}_{16}$.

Převod přes desítkovou soustavu

$$N = 1653_8 = 1 \cdot 8^3 + 6 \cdot 8^2 + 5 \cdot 8^1 + 3 \cdot 8^0 = 939$$

Dílčí podíl P^n	Zbytek Z^n	Koeficient a_n
939 : 16 = 58	11	$a_0 = \mathrm{B}$
58 : 16 = 3	10	$a_1 = \mathrm{A}$
3 : 16 = 0	3	$a_2 = 3$

Z tabulky je vidět, že jsme dospěli ke stejnému výsledku $N = 3\mathrm{AB}_{16}$.

*

ÚLOHY K ŘEŠENÍ 1-3

Pomocí metody postupného dělení základem převeďte z desítkové

a) do binární soustavy čísla ${}^1N_{10} = 79$, ${}^2N_{10} = 111$ a ${}^3N_{10} = 249$

b) do oktální soustavy čísla ${}^1N_{10} = 898$, ${}^2N_{10} = 169$ a ${}^3N_{10} = 1963$

c) do hexadecimální soustavy čísla ${}^1N_{10} = 1246$, ${}^2N_{10} = 1990$ a ${}^3N_{10} = 1963$

1.2 Převod čísel kladných desetinných

Pro číslo $N < 1$ o základu B můžeme podle vztahu (1-1) psát

$$N_B = b_{-1} \cdot B^{-1} + b_{-2} \cdot B^{-2} + \ldots + b_{-m+1} \cdot B^{-m-1} + b_{-m} \cdot B^{-m} \qquad (1\text{-}14)$$

kde koeficient b_{-m} je koeficient s nejmenší váhou. Číslo N_B pak můžeme analogicky zapsat jako $N_B = 0, b_{-1}, b_{-2}, \ldots b_{-m+1}, b_{-m}$, přičemž jeho přesnost je dána váhou B^{-m} .

Převod takovýchto čísel je možné uskutečnit:

- metodou postupného odečítání vah
- metodou postupného násobení základem

1.2.1 Metoda postupného odečítání vah

Metoda postupného odčítání vah je analogická s metodou pro celá kladná čísla a je uvedena v následujícím příkladě.

ŘEŠENÝ PŘÍKLAD

Převeďte číslo N =0,853 z dekadické do binární soustavy s přesností 5 bitů.

Řešení příkladu

Váha	Rozdíl	Koeficient b_m
$2^{-1} = 0{,}5$	0,853 - 0,5 = 0,353	$b_{-1} = 1$
$2^{-2} = 0{,}25$	0,353 - 0,25 = 0,103	$b_{-2} = 1$
$2^{-3} = 0{,}125$	0,103 - 0,125 = -0,022	$b_{-3} = 0$
$2^{-4} = 0{,}0625$	0,103 - 0,0625 = 0,0405	$b_{-4} = 1$
$2^{-5} = 0{,}03125$	0,0405 - 0,03125 = 0,00925	$b_{-5} = 1$

Z tabulky je vidět, že $N = 0{,}853_{10} = 0{,}11011_2$.

*

1.2.2 Metoda postupného násobení základem

Metoda postupného násobení základem vychází ze vztahu (1-14), který po vynásobení základem B dostane tvar:

$$N_B \cdot B = b_{-1} + b_{-2} \cdot B^{-1} + \ldots + b_{-m+1} \cdot B^{-m+2} + b_{-m} \cdot B^{-m+1} = b_{-1} + S^1 \qquad (1\text{-}15)$$

kde S^1 představuje dílčí součin polynomu. Dalším vynásobením zbytku získáme koeficient b_{-2} a dílčí součin S^2 .

$$S^1 \cdot B = b_{-2} + b_{-3} \cdot B^{-1} + \ldots + b_{-m+1} \cdot B^{-m+3} + b_{-m} \cdot B^{-m+2} = b_{-2} + S^2 \qquad (1\text{-}16)$$

Uvedený algoritmus je možné opakovat až do zadané délky polynomu nebo stanovené přesnosti zobrazení v počtu bitů.

ŘEŠENÝ PŘÍKLAD

Metodou postupného násobení základem převeďte číslo $0{,}634_{10}$ do binární, oktální a hexadecimální soustavy

Řešení příkladu

Převod do binární soustavy s přesností 7 bitů:

Dílčí součin S^n	Koeficient b_{-m}
0,634 x 2 = 1,268	$b_{-1} = 1$
0,268 x 2 = 0,536	$b_{-2} = 0$
0,536 x 2 = 1,072	$b_{-3} = 1$
0,072 x 2 = 0,144	$b_{-4} = 0$
0,144 x 2 = 0,288	$b_{-5} = 0$
0,288 x 2 = 0,576	$b_{-6} = 0$
0,756 x 2 = 1, 152	$b_{-7} = 1$

Převod do oktální soustavy s přesností 9 bitů:

Dílčí součin S^n	Koeficient b_{-m}
0,634 x 8 = 5,072 0,072 x 8 = 0,576 0,576 x 8 = 4,608	$b_{-1} = 5$ $b_{-2} = 0$ $b_{-3} = 4$

Převod do hexadecimální soustavy s přesností 12 bitů:

Dílčí součin S^n	Koeficient b_{-m}
0,634 x 16 = 10,144 0,144 x 16 = 2,304 0,304 x 16 = 4, 864	b_{-1} = A $b_{-2} = 2$ $b_{-3} = 4$

Číslo N = $0{,}634_{10} = 0{,}1010001_2 = 0{,}504_8 = 0{,}A24_{16}$.

*

ÚLOHY K ŘEŠENÍ 1-4

Zvolte si metodu a převeďte z desítkové soustavy

a) do binární soustavy čísla ${}^1N_{10} = 0{,}379$, ${}^2N_{10} = 0{,}029$ a ${}^3N_{10} = 0{,}411$

b) do oktální soustavy čísla ${}^1N_{10} = 0{,}123$, ${}^2N_{10} = 0{,}289$ a ${}^3N_{10} = 0{,}659$

c) do hexadecimální soustavy čísla ${}^1N_{10} = 0{,}288$, ${}^2N_{10} = 0{,}271$ a ${}^3N_{10} = 0{,}398$

1.3 Vztah mezi binární, oktální a hexadecimální soustavou

Předpokládejme, že máme celé kladné číslo N vyjádřené v binární soustavě polynomem

$$N_2 = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 \qquad (1\text{-}17)$$

V souladu s kapitolou 1.2.1 pro převod do oktální soustavy vydělíme tento polynom základem nové soustavy, tj. $2^3 = 8$. Potom dostaneme

$$\begin{aligned} N_2 &= \left[a_n \cdot 2^{n-3} + a_{n-1} \cdot 2^{n-4} + \ldots\right] \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 = \\ &= P^0 \cdot 2^3 + Z^0 \end{aligned} \qquad (1\text{-}18)$$

kde P^0 představuje celočíselný podíl polynomu a Z^0 je zbytek. Protože Z^0 představuje člen s nejmenší váhou v oktální soustavě, platí

$$Z^0 = a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 \qquad (1\text{-}19)$$

a lze konstatovat, že tento člen je dán součtem 3 nejnižších vah v binární soustavě. Minimální hodnota výrazu *(1-19)* je pro $a_2 = a_1 = a_0 = 0$ a maximální hodnota pro $a_2 = a_1 = a_0 = 1$. Zbytek Z^0 nabývá hodnot 0 až 7, což znamená, že obsahuje všechny symboly potřebné pro zobrazení v oktální soustavě. Dalším vydělením polynomu P^0 váhou 2^3 získáme zbytek Z^1, který představuje v oktální soustavě váhu 8^1 .

Pro převod z binární do oktální soustavy lze tedy postupovat tak, že binární číslo rozdělíme na skupiny po 3 symbolech (bitech) od desetinné čárky směrem vlevo a tyto vyjádříme v symbolech oktální soustavy.

Pro převod do hexadecimální soustavy vyjdeme opět ze vztahu *(1-17)*, který po vydělení základem hexadecimální soustavy, tj. 2^4 má tvar

$$\begin{aligned} N_2 &= \left[a_n \cdot 2^{n-4} + a_{n-1} \cdot 2^{n-5} + \ldots + a_4\right] \cdot 2^4 + a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 = \\ &= P^1 \cdot 2^4 + Z^0 \end{aligned} \qquad (1\text{-}20)$$

Zbytek $Z^0 = a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$ charakterizuje člen s nejnižší váhou v hexadecimální soustavě. Pro převod čísla lze tedy binární číslo rozdělit na skupiny po 4 symbolech (bitech) od desetinné čárky směrem vlevo a tyto vyjádřit v symbolech hexadecimální soustavy.

ŘEŠENÝ PŘÍKLAD

Převeďte číslo N = 10110110001_2 do oktální a hexadecimální soustavy.

Řešení příkladu

Převod do oktální soustavy:
N_2 = 10_110_110_001 odkud $N_8 = 2661_8$.

Převod do hexadecimální soustavy:
N_{16} = 101_1011_0001 odkud $N_{16} = 5B1_{16}$.

*

ŘEŠENÝ PŘÍKLAD

Pomocí převodu přes binární soustavu převeďte z oktální do hexadecimální soustavy číslo N_8 = 705 a z hexadecimální do oktální soustavy číslo N_{16} = 10F

Řešení příkladu

$N_8 = 705 \rightarrow N_2 = 111_000_101 \rightarrow N_2 = 1_1100_0101 \rightarrow N_{16} = 1C5$

$N_{16} = 10F \rightarrow N_2 = 1_0000_1111 \rightarrow N_2 = 100_001_111 \rightarrow N_8 = 417$

Pokud se převod kladných čísel N<1 z binární soustavy do soustavy oktální, resp. hexadecimální uskutečňuje metodou postupného násobení, znamená to, že polynom *(1-14)* vynásobíme základem umocněným na druhou pro převod do oktální soustavy nebo základem umocněným na třetí pro převod do hexadecimální soustavy. Prakticky to odpovídá rozdělení polynomu do skupin po třech, resp. po čtyřech symbolech (bitech) od desetinné čárky vpravo a vyjádření těchto skupin odpovídajícími symboly.

ŘEŠENÝ PŘÍKLAD

Převeďte číslo N = $0{,}2735_8$ do soustavy hexadecimální.

Řešení příkladu

$N = 0{,}010_111_011_101_2 = 0{,}0101_1101_1101_2 = 0{,}5DD_{16}$

*

ÚLOHY K ŘEŠENÍ 1-5

Pomocí přímého převodu převeďte

a) z binární do oktální soustavy číslo N_2 = 110111100
b) z oktální do binární soustavy číslo N_8 = 754
c) z binární do hexadecimální soustavy N_2 = 1011111001
d) z hexadecimální do binární N_{16} = C5

SHRNUTÍ KAPITOLY ČÍSELNÉ SOUSTAVY

Shrnutí

V kapitole Číselné soustavy jsme uvedli různé typy číselných soustav a používané metody pro převody čísel mezi nimi, a to pro kladná celá čísla a kladná desetinná čísla. Dále jsou zde uvedeny vztahy mezi binární, oktální a hexadecimální číselnou soustavou.

Jak uvidíme v následujících kapitolách, je zvládnutí problematiky kapitoly Číselné soustavy nezbytné zejména při zápisu vektorů proměnných v binární, oktální a hexadecimální soustavě, při vektorovém zápisu logické funkce a v mikropočítačové technice.

2 BOOLEOVA ALGEBRA

ČAS POTŘEBNÝ KE STUDIU

Předpokládaný čas k prostudování kapitoly Booleova algebra je **20 hodin.**

RYCHLÝ NÁHLED DO PROBLEMATIKY KAPITOLY BOOLEOVA ALGEBRA

Booleova algebra představuje jeden z možných matematických prostředků, pomocí něhož lze pracovat s logickými (booleovskými) proměnnými, které nabývají pouze dvou hodnot. Tyto hodnoty se označují symboly **0** a **I** nebo ***L*** a ***H*** (z angličtiny „low“a „high“) podle nižší a vyšší signální veličiny, která zobrazuje logické hodnoty.

Logická proměnná může v určitém logickém systému například vyjadřovat, je-li vypínač sepnut (I) nebo rozepnut (0), je-li nějaká fyzikální veličina větší nebo menší než daná hodnota apod.

Základní význam při popisu chování kombinačních obvodů, kterým je věnována kapitola 3, mají dvouhodnotové funkce s dvouhodnotovými proměnnými. Takovéto funkce se nazývají booleovské funkce.

Kapitola je zaměřena na způsoby zápisu booleovských funkcí a na nejběžnější metody jejich minimalizace.

CÍLE KAPITOLY BOOLEOVA ALGEBRA

- Budete umět definovat logickou proměnnou, funkci rovnosti a logické operátory, napsat pravdivostní tabulku logické funkce, zapsat funkci do Karnaughovy mapy a vyjádřit funkci pomocí booleovského výrazu,
- získáte přehled o různých možnostech zápisu logických funkcí a přehled o používaných metodách minimalizace logických funkcí,
- budete schopni uplatnit zákony Booleovy algebry při úpravě a zjednodušování booleovských výrazů,
- budete schopni upravovat booleovské výrazy do žádaného tvaru a minimalizovat booleovské funkce vhodnou minimalizační metodou.

KLÍČOVÁ SLOVA KAPITOLY BOOLEOVA ALGEBRA

Klíčová slova

Logická proměnná, funkce rovnosti, logické operátory, logický součin, logický součet, negace (komplement), asociativní zákon, komutativní zákon, distributivní zákon, zákon absorpce, De Morganovy zákony, zákon absorpce konsensu, booleovská funkce, minimalizace, algebraická minimalizac, Mc-Cluskey, Karnaughova mapa.

K ZAPAMATOVÁNÍ 1

Booleova algebra je definována souborem postulátů a teorémů, které jsou tvořeny:

- logickými proměnnými
- funkcemi rovnosti
- logickými operátory

DEFINICE LOGICKÉ PROMĚNNÉ 2-1

Df

Jestliže x je logická proměnná, která může nabývat pouze dvou hodnot (0,1), musí platit:

$x = 1$ když $x \neq 0$ a $x = 0$ když $x \neq 1$

DEFINICE FUNKCE ROVNOSTI 2-2

Df

Když uvažujeme dvě logické proměnné x_0 a x_1 a současně platí, že $x_0 = 1$ a $x_1 = 1$ nebo $x_0 = 0$ a $x_1 = 0$, lze psát, že $x_0 = x_1$ a znamená to, že tyto proměnné se sobě rovnají.

DEFINICE LOGICKÝCH OPERÁTORŮ 2-3

Pro dvě libovolné logické proměnné x_0 a x_1 jsou definovány základní logické operace - logický součin a logický součet.

Operátor logického součinu

$x_0 \cdot x_1 = 1 \Leftrightarrow x_0 = 1$ a zároveň $x_1 = 1$

$x_0 \cdot x_1 = 0 \Leftrightarrow x_0 = 0$ nebo $x_1 = 0$ nebo $x_0 = x_1 = 0$

Operátor logického součtu

$x_0 + x_1 = 1 \Leftrightarrow x_0 = 1$ nebo $x_1 = 1$ nebo $x_0 = x_1 = 1$

$x_0 + x_1 = 0 \Leftrightarrow x_0 = 0$ a zároveň $x_1 = 0$

K ZAPAMATOVÁNÍ 2

Logický komplement (negace)

$$\bar{x} = 0 \Leftrightarrow x = 1$$
$$\bar{x} = 1 \Leftrightarrow x = 0$$

Další zákony a věty Booleovy algebry jsou:

- komutativní zákon
- asociativní zákon
- distributivní zákon
- zákon absorpce
- zákon absorpce konsenzu
- De Morganovy zákony
- zákon spojení

Komutativní zákon ***2-1***

V

$$x_0 \cdot x_1 = x_1 \cdot x_0$$
$$x_0 + x_1 = x_1 + x_0$$

Asociativní zákon ***2-2***

V

$$x_0 \cdot (x_1 \cdot x_2) = (x_0 \cdot x_1) \cdot x_2$$
$$x_0 + (x_1 + x_2) = (x_0 + x_1) + x_2$$

Distributivní zákon ***2-3***

V

$$x_0 \cdot (x_1 + x_2) = x_0 \cdot x_1 + x_0 \cdot x_2$$
$$x_0 + (x_1 \cdot x_2) = (x_0 + x_1) \cdot (x_0 + x_2)$$

Zákon absorpce ***2-4***

V

$$x_0 + (x_0 \cdot x_1) = x_0$$
$$x_0 \cdot (x_0 + x_1) = x_0$$

Důkaz

$$x_0 + (x_0 \cdot x_1) = x_0 \cdot (1 + x_1) = x_0$$
$$x_0 \cdot (x_0 + x_1) = x_0 \cdot x_0 + x_0 \cdot x_1 = x_0 \cdot (1 + x_1) = x_0$$

Zákon absorpce konsenzu ***2-5***

V

$$x_2 \cdot x_1 + \overline{x_2} \cdot x_0 + x_1 \cdot x_0 = x_2 \cdot x_1 + \overline{x_2} \cdot x_0$$
$$(x_2 + x_1) \cdot (\overline{x_2} + x_0) \cdot (x_1 + x_0) = (x_2 + x_1) \cdot (\overline{x_2} + x_0)$$

Důkaz

Poslední člen na levé straně první rovnice vynásobíme výrazem

$(x_2 + \overline{x_2})$, roznásobíme a vytkneme:

$$x_2 \cdot x_1 + \overline{x_2} \cdot x_0 + x_1 \cdot x_0 = x_2 \cdot x_1 + \overline{x_2} \cdot x_0 + x_1 \cdot x_0 \cdot 1 =$$
$$= x_2 \cdot x_1 + \overline{x_2} \cdot x_0 + x_1 \cdot x_0 \cdot (x_2 + \overline{x_2}) =$$
$$= x_2 \cdot x_1 + \overline{x_2} \cdot x_0 + x_0 \cdot x_1 \cdot x_2 + x_0 \cdot x_1 \cdot \overline{x_2} =$$
$$= x_2 \cdot x_1 \cdot (1 + x_0) + \overline{x_2} \cdot x_0 \cdot (1 + x_1) = x_2 \cdot x_1 + \overline{x_2} \cdot x_0$$

V druhé rovnici roznásobíme závorky současně na levé i pravé straně
a postupně upravujeme pomocí zavedených postulátů:

$$(x_2 + x_1) \cdot (\overline{x_2} + x_0) \cdot (x_1 + x_0) = (x_2 + x_1) \cdot (\overline{x_2} + x_0)$$
$$(x_2 \cdot \overline{x_2} + x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot x_0) \cdot (x_1 + x_0) = x_2 \cdot \overline{x_2} + x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot x_0$$
$$(0 + x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot x_0) \cdot (x_1 + x_0) = 0 + x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot x_0$$
$$x_2 \cdot x_0 \cdot x_1 + x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot \overline{x_2} \cdot x_0 + x_1 \cdot x_0 + x_1 \cdot x_0 = x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot x_0$$
$$x_2 \cdot x_0 \cdot (x_1 + 1) + x_1 \cdot x_0 \cdot (\overline{x_2} + 1 + 1) + x_1 \cdot \overline{x_2} + x_1 \cdot x_0 = x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot x_0$$
$$x_2 \cdot x_0 \cdot 1 + x_1 \cdot x_0 \cdot 1 + x_1 \cdot \overline{x_2} = x_2 \cdot x_0 + x_1 \cdot \overline{x_2} + x_1 \cdot x_0$$
$$x_2 \cdot x_0 + x_1 \cdot x_0 + x_1 \cdot \overline{x_2} = x_2 \cdot x_0 + x_1 \cdot x_0 + x_1 \cdot \overline{x_2}$$
$$x_2 \cdot x_0 + x_1 \cdot \overline{x_2} = x_2 \cdot x_0 + x_1 \cdot \overline{x_2}$$

De Morganovy zákony **2-6**

De Morganovy zákony převádí operaci logického součinu na operaci logického součtu a naopak.

$$\overline{x_0 \cdot x_1} = \overline{x_0} + \overline{x_1}$$
$$\overline{x_0 + x_1} = \overline{x_0} \cdot \overline{x_1}$$

Důkaz

Mějme dvě booleovské proměnné x_0, x_1 a booleovské výrazy $x_0 + x_1$, $x_0 \cdot x_1$, $\overline{x_0 + x_1}$, $\overline{x_0 \cdot x_1}$, $\overline{x_0} + \overline{x_1}$ a $\overline{x_0} \cdot \overline{x_1}$. Vyjádříme-li logické hodnoty uvedených proměnných a výrazů, vyplývá platnost zákonů z následující tabulky:

x_0	x_1	$\overline{x_0}$	$\overline{x_1}$	$x_0 + x_1$	$x_0 \cdot x_1$	$\overline{x_0 + x_1}$	$\overline{x_0 \cdot x_1}$	$\overline{x_0} + \overline{x_1}$	$\overline{x_0} \cdot \overline{x_1}$
0	0	1	1	0	0	1	1	1	1
0	1	1	0	1	0	0	1	1	0
1	0	0	1	1	0	0	1	1	0
1	1	0	0	1	1	0	0	0	0

Zákon spojení **2-7**

$$x_1 \cdot x_0 + \overline{x_1} \cdot x_0 = x_0$$
$$(x_1 + x_0) \cdot (\overline{x_1} + x_0) = x_0$$

Důkaz:

$$x_1 \cdot x_0 + \overline{x_1} \cdot x_0 = x_0 \cdot (x_1 + \overline{x_1}) = x_0 \cdot 1 = x_0$$

$$(x_1 + x_0) \cdot (\overline{x_1} + x_0) = (x_1 \cdot \overline{x_1}) + (x_1 \cdot x_0) + (x_0 \cdot \overline{x_1}) + (x_0 \cdot x_0) =$$
$$= 0 + (x_1 \cdot x_0) + (x_0 \cdot \overline{x_1}) + x_0 = x_0 \cdot (x_1 + \overline{x_1} + 1) = x_0 \cdot (1 + 1) = x_0 \cdot 1 = x_0$$

K ZAPAMATOVÁNÍ 3

$0 \cdot 0 = 0$	$0 + 0 = 0$
$0 \cdot 1 = 1 \cdot 0 = 0$	$0 + 1 = 1 + 0 = 1$
$1 \cdot 1 = 1$	$1 + 1 = 1$
$x \cdot 1 = 1 \cdot x = x$	$0 + x = x + 0 = x$
$0 \cdot x = x \cdot 0 = 0$	$x + 1 = 1 + x = 1$
$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$
$x \cdot x = x$	$x + x = x$
$\overline{\overline{x}} = x$	

ŘEŠENÝ PŘÍKLAD

Zjednodušte booleovský výraz.

$$(x_2 + x_1 + x_0) \cdot (x_2 + \overline{x_1} + x_0)$$

Řešení příkladu

Roznásobíme a upravíme:

$$(x_2 + x_1 + x_0) \cdot (x_2 + \overline{x_1} + x_0) =$$

$$= \overbrace{x_2 \cdot x_2}^{X_2} + x_2 \cdot \overline{x_1} + x_2 \cdot x_0 + x_1 \cdot x_2 + \overbrace{x_1 \cdot \overline{x_1}}^{0} + x_1 \cdot x_0 + x_0 \cdot x_2 + x_0 \cdot \overline{x_1} + \overbrace{x_0 \cdot x_0}^{X_0} =$$

$$= \left(x_2 + \overbrace{x_2 \cdot \overline{x_1} + x_1 \cdot x_2}^{X_2 \cdot (X_1 + \overline{X_1})} + \overbrace{x_2 \cdot x_0 + x_0 \cdot x_2}^{X_2 \cdot X_0} + \overbrace{x_1 \cdot x_0 + x_0 \cdot \overline{x_1}}^{X_0 \cdot (X_1 + \overline{X_1})} + x_0 \right) =$$

$$= \left(\overbrace{x_2 + x_2}^{X_2} + \overbrace{x_0 + x_0}^{X_0} + x_2 \cdot x_0 \right) = x_2 + x_0 \overbrace{(x_2 + 1)}^{1} = \underline{\underline{(x_2 + x_0)}}$$

Výsledek lze psát přímo uplatněním zákona spojení. Výše uvedený postup je možné považovat za důkaz zákona.

*

ŘEŠENÝ PŘÍKLAD

Zjednodušte výraz $\overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot x_0$ pomocí Booleovy algebry.

Řešení příkladu

$$\overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot x_0 = x_1 \cdot x_0 \cdot \left(\overline{x_2} + x_2\right) + x_2 \cdot \overline{x_1} \cdot x_0 =$$

$$= x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot x_0 = x_0 \cdot \left(x_1 + x_2 \cdot \overline{x_1}\right) = x_0 \cdot \left(\left(x_1 + x_2\right) \cdot \left(x_1 + \overline{x_1}\right)\right) =$$

$$= x_0 \cdot \left(x_1 + x_2\right) = \underline{\underline{x_0 \cdot x_1 + x_0 \cdot x_2}}$$

K ZAPAMATOVÁNÍ 4

Vložení výrazu

Z předchozího už víme, že $x + x = x$. Také změníme-li například výraz $x_1 \cdot x_0$ na tvar $x_1 \cdot x_0 + x_1 \cdot x_0$, bude logická hodnota obou výrazů stejná.

V této souvislosti je vhodné si uvědomit, že ne vždy je výhodné se okamžitě snažit počet členů a proměnných při úpravě výrazu ihned snižovat. Na počátku nebo i v průběhu dalších úprav nám může pomoci, vložíme-li do výrazu člen, který hodnotu výrazu nezmění, ale další úpravy zjednoduší.

ŘEŠENÝ PŘÍKLAD

Pomocí Booleovy algebry upravte výraz:
$\overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot x_0$.

Řešení příkladu

Algebraickou úpravu si zjednodušíme vložením vhodného členu:

$$\overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot x_0 =$$
$$= \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot x_0 + \underline{x_2 \cdot x_1 \cdot x_0} =$$
$$= x_1 \cdot x_0 \cdot \left(\overline{x_2} + x_2\right) + x_2 \cdot x_0 \cdot \left(\overline{x_1} + x_1\right) = \underline{\underline{x_1 \cdot x_0 + x_2 \cdot x_0}}$$

Upravovaný výraz je stejný jako v předchozím řešeném příkladu – porovnejte postup při zjednodušování. *

ŘEŠENÝ PŘÍKLAD

Pomocí Booleovy algebry a De Morganových zákonů zjednodušte booleovský výraz.

$$\left(x_3 + x_2 \cdot x_1\right) \cdot \left(\overline{x_2} + \overline{x_1 \cdot x_0}\right) + \overline{x_2 + \overline{x_1}}$$

Řešení příkladu

Na odpovídající členy aplikujeme nejprve De Morganovy zákony, roznásobíme a zjednodušíme.

$$\left(x_3 + x_2 \cdot x_1\right) \cdot \left(\overline{x_2} + \overline{x_1 \cdot x_0}\right) + \overline{x_2 + \overline{x_1}} = \left(x_3 + x_2 \cdot x_1\right) \cdot \left(\overline{x_2} + \overline{x_1} + \overline{x_0}\right) + \overline{x_2} \cdot x_1 =$$
$$x_3 \cdot \overline{x_2} + x_3 \cdot \overline{x_1} + x_3 \cdot \overline{x_0} + 0 + 0 + x_2 \cdot x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 = x_3 \cdot \overline{x_1} + x_3 \cdot \overline{x_0} +$$
$$x_2 \cdot x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 = x_3 \cdot \overline{x_1} + x_3 \cdot \overline{x_0} + x_1 \cdot \left(x_2 \cdot \overline{x_0} + \overline{x_2}\right) = x_3 \cdot \overline{x_1} + x_3 \cdot \overline{x_0} +$$
$$x_1 \cdot \left(\overline{x_0} + \overline{x_2}\right) = x_3 \cdot \overline{x_1} + x_3 \cdot \overline{x_0} + x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1 = x_3 \cdot \overline{x_1} + x_1 \cdot \overline{x_0} + \overline{x_2} \cdot x_1$$

*

ÚLOHY K ŘEŠENÍ 2-1

Pomocí booleovy algebry zjednodušte následující výrazy:

a) $\overline{x_2} \cdot x_1 + \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_0}$

b) $(x_2 + x_0) \cdot (x_2 + x_1) \cdot (x_1 + x_0) \cdot (x_2 + \overline{x_1})$

c) $(x_2 + x_1 + \overline{x_0}) \cdot (\overline{x_2} + \overline{x_1} + x_0)$

d) $((x_1 + x_2) + (\overline{x_1 + x_2})) \cdot x_3$

e) $\overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0} + \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot x_0$

f) $x_2 \cdot x_1 \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot x_0 + x_2 \cdot x_1 \cdot x_0$

g) $\overline{(\overline{x_2} + \overline{x_1}) \cdot (\overline{x_1} \cdot x_2)} + (\overline{x_2} + \overline{x_1}) \cdot \overline{x_1} \cdot x_2$

h) $x_3 \cdot \overline{x_2} \cdot x_1 \cdot x_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0 + x_3 \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x_0}$

ÚLOHA K ŘEŠENÍ 2-2

Pomocí booleovy algebry upravte následující výraz tak, aby obsahoval pouze logické součty.

$(\overline{x_2} + x_0) \cdot (x_1 + x_0) \cdot (\overline{x_1} + \overline{x_0})$

ÚLOHY K ŘEŠENÍ 2-3

Dokažte, že platí:

$$\overline{(\overline{x_2 + \overline{x_0}}) + (\overline{\overline{x_2} + x_0})} + \overline{x_1} = \overline{x_2 + \overline{x_1} + x_0} + \overline{\overline{x_2} + \overline{x_1} + \overline{x_0}}$$

2.1 Booleovské funkce

Booleovské funkce jsou dvouhodnotové funkce s dvouhodnotovými proměnnými.

Booleovská funkce

Úplná booleovská funkce $f(x_{n-1},..,x_1,x_0)$ o n proměnných $x_{n-1},..x_1,x_0$ je zobrazení

Úplná booleovská funkce

$$f: \{0,1\}^n \rightarrow \{0,1\} \qquad (2\text{-}4)$$

kde $(0,1)^n$ - je množina, která tvoří definiční obor booleovské funkce,

(0,1) - je množina, která tvoří obor hodnot booleovské funkce

Definičním oborem funkce je tedy množina všech n-tic s prvky 0 a 1, kterým odpovídají všechna možná uspořádání n-tic hodnot proměnných $x_{n-1},..x_1,x_0$. Obor tedy obsahuje právě 2^n n-tic.

Definiční obor úplné booleovské funkce

Booleovská funkce přiřazuje každé n-tici hodnot proměnných určitou hodnotu 0 a 1.

Booleovské funkci dvou proměnných odpovídá určité zobrazení $\{0,1\}^2 \rightarrow \{0,1\}$, kde $\{0,1\}^2 = \{0,0\},\{0,1\},\{1,0\},\{1,1\}$ je množina všech dvojic hodnot proměnných. Příklad konkrétní booleovské funkce f je uveden na obr. 2-1.

x_1	x_0	f
0	0	0
0	1	1
1	0	0
1	1	1

Obr. 2-1: Příklad konkrétní booleovské funkce dvou proměnných

Booleovské funkce jedné proměnné

Z definice booleovské funkce je zřejmé, že pro n proměnných existuje $2^{(2^n)}$ různých booleovských funkcí. Pro jednu proměnnou existují tedy 4 různé funkce, které jsou uvedeny na obr. 2-2.

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Obr. 2-2: Booleovské funkce jedné proměnné

Booleovské funkce dvou proměnných

Analogicky pro dvě vstupní proměnné existuje 16 různých funkcí, obr. 2-3.

x_1	x_0	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Obr. 2-3: Booleovské funkce dvou proměnných

Podrobnějším studiem obr. 2-3 zjistíme, že obsahuje některé velmi známé funkce, např. f_8 je známá Piercova funkce, f_{14} je Schäfferova funkce, f_9 je ekvivalence atd.

Neúplná booleovská funkce

Při řešení praktických úloh návrhu logických kombinačních obvodů se setkáváme s neúplnými booleovskými funkcemi. Neúplná booleovská funkce $f(x_{n-1},\ldots,x_1,x_0)$ je zobrazení

$$f\colon Q \rightarrow \{0,1\} \qquad (2\text{-}5)$$

kde $Q \subset \{0,1\}^n$.

Definiční obor neúplné booleovské funkce

Definičním oborem neúplné booleovské funkce je vlastně podmnožina Q množiny $\{0,1\}^n$. Neúplná funkce proto není definována ve všech bodech oboru $\{0,1\}^n$ úplné booleovské funkce. Obor Q je například tvořen množinou hodnot proměnných $Q = \{000,010,100,110\}$, v nichž je definována (není definována v bodech 001,011,101 a 111). Tabulkové vyjádření takové funkce je uvedeno na obr. 2-4.

x_2	x_1	x_0	f
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	1

Obr. 2-4: Příklad neúplné booleovské funkce

Rozšířená booleovská funkce

Pro řešení praktických úkolů je potom výhodnější využívat obecněji definovanou booleovskou funkci. Rozšířená booleovská funkce $f(x_{n-1},...,x_1,x_0)$ je zobrazení

$$f\colon \{0,1\}^n \rightarrow \{0,1,X\} \qquad (2\text{-}6)$$

kde symbol X se interpretuje jako hodnota neurčitá, tj. libovolná hodnota (0 nebo 1).

x_2	x_1	x_0	f_1	f_2	f_3
0	0	0	0	0	0
0	0	1	X	0	0
0	1	0	1	1	1
0	1	1	X	1	1
1	0	0	0	0	0
1	0	1	X	1	0
1	1	0	1	1	1
1	1	1	X	0	1

Obr. 2-5: Příklad rozšířené booleovské funkce f_1 s rovnocennými úplnými booleovskými funkcemi f_2 a f_3

Množina rozšířených booleovských funkcí s n proměnnými proto obsahuje 3^{2^n} různých funkcí a zobrazuje i dříve definovanou neúplnou booleovskou funkci. Příklad rozšířené booleovské funkce je uveden na obr. 2-5.

Z uvedeného je zřejmé, že booleovské funkce realizující zobrazení

Booleovská funkce

$$F: A \rightarrow U \qquad (2\text{-}7)$$

kde $A = \{0,1\}^n$ a $U = \{0,1\}^m$ jsou množiny vstupních nebo výstupních vektorů obvodu s n vstupními a m výstupními proměnnými a umožňují popis kombinačního obvodu.

Pro logický kombinační obvod, který má m výstupních proměnných a n vstupních proměnných platí:

$$y_i = f_i(x_{n-1},\ldots x_0) \qquad (2\text{-}8)$$

kde $i = 1,2,3\ldots m$.

Například pro chování kombinačního obvodu se dvěma vstupními a dvěma výstupními proměnnými lze psát:

$$y_0 = f_1(x_1,x_0) \qquad (2\text{-}9)$$

$$y_1 = f_2(x_1,x_0)$$

Závěr booleovských funkcí

Je třeba poznamenat, že v chování kombinačních obvodů se velmi často vyskytuje potřeba použití rozšířené booleovské funkce, a to především v těchto případech:

- určité vstupní vektory se nevyskytují, tj. okolí (prostředí) obvodu je negeneruje,
- hodnota výstupní proměnné není definovaná pro některé vstupní vektory.

Je však třeba připomenout, že při popisu chování kombinačního obvodu pomocí booleovských funkcí se nepostihují jeho dynamické vlastnosti. Vztahy mezi proměnnými vyjadřují ustálený stav, tj. stav po akceptování změn vstupních a výstupních proměnných.

2.2 Způsoby zápisu booleovských funkcí

Každý matematický zápis funkce se ve své podstatě vyznačuje svojí jednoznačností. Protože však pro zápis booleovských funkcí nelze použít známé způsoby z matematiky, zapisují se jiným jednoznačným způsobem.

2.2.1 Tabulkové, vektorové a číselné zápisy

Tabulkový zápis

Tabulkový zápis, resp. zápis pomocí pravdivostní tabulky je nejznámější způsob zápisu. Tabulka pro úplnou booleovskou funkci f na obr. 2-6 a) obsahuje pro n vstupních proměnných 2^n kombinací logických hodnot, a proto musí mít 2^n řádků. Je zřejmé, že tento zápis je vhodný pro menší počet vstupních proměnných. Například pro 8 vstupních proměnných vychází až 256 řádků a takovýto zápis vzhledem ke svým rozměrům ztrácí na přehlednosti.

Pro snížení počtu řádků se proto někdy používá tzv. zhuštěný zápis. Princip zhuštěného zápisu spočívá v použití symbolu X i pro hodnoty vstupních proměnných. V tomto případě symbol X znamená, že logická hodnota výstupní proměnné je stejná pro logickou hodnotu dané vstupní proměnné 0 nebo 1. Princip zhuštěného zápisu funkce f je zřejmý z obr. 2-6 b).

a) Úplný zápis

n	x_2	x_1	x_0	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

b) Zhuštěný zápis

n	x_2	x_1	x_0	f
0,2	0	X	0	0
1,3	0	X	1	1
4,5	1	0	X	1
7	1	1	1	0

Obr. 2-6: Způsoby tabulkového zápisu funkce f

Číselný zápis booleovské funkce využívá skutečnost, že logické hodnoty se ztotožňují s číselnou hodnotou. Logická hodnota I se ztotožní s číselnou hodnotou 1, logická hodnota 0 se ztotožní s číselnou hodnotou 0. Na základě záměny logické hodnoty za číselnou lze vstupní *n*-tici chápat jako číslo vyjádřené ve dvojkové soustavě. Takovéto číslo se potom nazývá index. Předpokladem využití číselného zápisu však je pevně definované pořadí vstupních proměnných. Toto pořadí se uvádí v závorce za symbolem funkce, přičemž zároveň vyjadřuje i váhové pořadí proměnných, zleva doprava, od nejvyšší váhy k váze nejnižší. Například *n*-tice $x_2x_1x_0 = 0I0 = 010_2 = 2_{10}$.

Číselný zápis

Existují dvě základní formy číselného zápisu:

Disjunktivní číselný zápis funkce- za symbolem rovnosti se uvádí symbol D a v závorce jsou potom uvedeny indexy vstupní *n*-tice, vyjádřené v desítkové soustavě, v nichž booleovská funkce nabývá hodnoty logické I.

Disjunktivní zápis

Například funkce *f* z obr. 2-6 bude vyjádřena takto:

$$f(x_2, x_1, x_0) = \mathrm{D}(1,3,4,5,6) \qquad (2\text{-}10)$$

Konjunktivní číselný zápis funkce- za symbolem rovnosti je uveden symbol K a v závorce jsou uvedeny indexy v nichž booleovská funkce nabývá logické hodnoty 0.

Konjunktivní zápis

Jako příklad opět zápis funkce *f* z obr. 2-6:

$$f(x_2, x_1, x_0) = \mathrm{K}(0,2,7) \qquad (2\text{-}11)$$

Je zřejmé, že neúplnou booleovskou funkci nelze tímto způsobem zapsat a pro rozšířené booleovské funkce je nutné zavést z důvodu jednoznačnosti zápisu opět konvenci. Konvence spočívá v tom, že za symbolem D, resp. K jsou v závorce uvedeny dva soupisy indexů čísel. První soupis odpovídá disjunktivnímu (konjunktivnímu) zápisu. Druhý soupis je vnořen pomocí závorek do prvního soupisu a obsahuje indexy vstupních *n*-tic, ve kterých booleovská funkce má hodnotu X.

Při této konvenci lze pro funkci f_1 z obr. 2-5 psát:

$$f_1(x_2, x_1, x_0) = \mathrm{D}(2,6(1,3,5,7)) \qquad (2\text{-}12)$$

$$f_1(x_2, x_1, x_0) = \mathrm{K}(0,4(1,3,5,7)) \qquad (2\text{-}13)$$

Vektorový zápis

Vektorový zápis booleovské funkce využívá skutečnost, že logické hodnoty funkce jsou uspořádány v řádcích a uvažují se jako hodnoty číselné. Pořadí vstupních proměnných je uvedeno v závorce za symbolem funkce a určuje váhu proměnné. První hodnota za symbolem rovnosti odpovídá nejvyššímu indexu a poslední hodnota nejnižšímu indexu vstupní *n*-tice. Hodnoty booleovské funkce jsou rovněž psány sestupně zleva doprava.

Vektorový zápis booleovské funkce z obr. 2-6 lze potom psát ve tvaru:

$$f(x_2, x_1, x_0) = 01111010 \qquad (2\text{-}14)$$

Protože funkční hodnoty v tomto případě představují číslo ve dvojkové soustavě, je možné tento zápis modifikovat zkráceně i v jiné číselné soustavě.

Ekvivalentní zápisy vztahu *(2-14)* tudíž jsou:

$$\begin{aligned} f(x_2, x_1, x_0) &= 172_8 \\ f(x_2, x_1, x_0) &= 7\mathrm{A}_{16} \end{aligned} \qquad (2\text{-}15)$$

V případě rozšířené booleovské funkce se v zápisu mezi použitými číselnými hodnotami 0 a 1 vyskytuje ještě symbol X.

Například funkce f_1 z obr. 2-5:

$$f_1(x_2, x_1, x_0) = \mathrm{X1X0X1X0} \qquad (2\text{-}16)$$

Neúplnou booleovskou funkci nelze tímto způsobem zapsat.

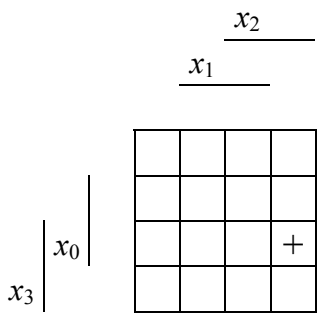
Geometrický zápis

Geometrický zápis čísel pomocí mapy. Mapa představuje geometrické znázornění definičního oboru $\{0,1\}^n$ booleovské funkce v rovině pomocí čtverečků, přičemž každé *n*-tici oboru je použitým kódováním přiřazen čtvereček. Mapa tedy tvoří síť obsahující 2^n čtverečků. Příklad mapy pro zápis funkce čtyř proměnných a způsob přiřazení čtverečků jednotlivým čtveřicím hodnot proměnných x_3, x_2, x_1, x_0 (tj. způsob kódování) je zřejmý z obr. 2-7.

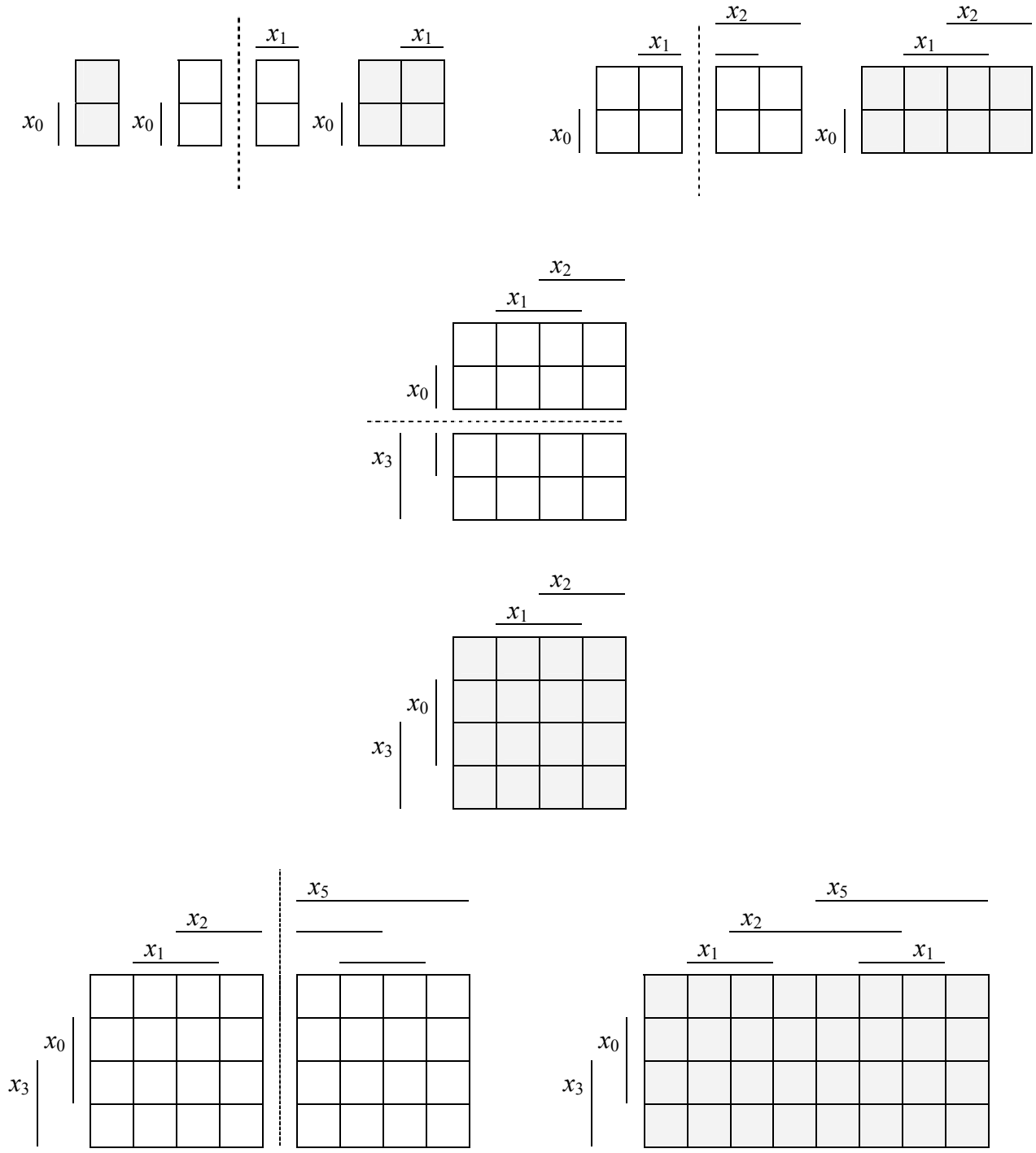
Karnaughova mapa

Pomocí kódovacích čar na levém a horním okraji mapy a dle připsaných proměnných jsou definovány čtverečky, ve kterých jednotlivé vstupní proměnné nabývají hodnoty logické 0 nebo I. Předpokládá se, že v oblasti nacházející se pod čarou příslušné proměnné tato proměnná nabývá hodnotu log I a mimo tuto oblast hodnotu log 0. Čtvereček označený + proto odpovídá vstupní *n*-tici II0I. Takto uspořádaná mapa se nazývá Karnaughova mapa.

Postup vytvoření Karnaughovy mapy pro libovolný počet proměnných a zajištění správného kódování vychází z mapy pro 1 proměnnou a je uveden na obr. 2-8. Algoritmus vytvoření mapy pro $n+1$ proměnných vychází z mapy pro *n* proměnných a spočívá ve vytvoření zrcadlového obrazu mapy a jeho připojení k původní mapě. Nový zrcadlový obraz se označí novou proměnnou. Z hlediska přehledu a orientace se používají mapy nejvíce pro 5 proměnných.



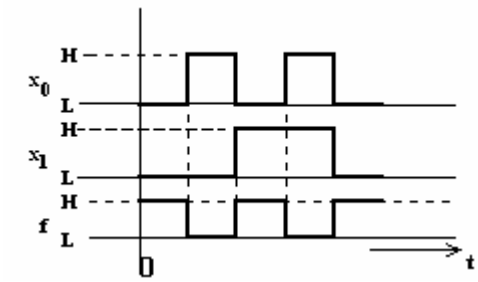
Obr. 2-7: Mapa pro 4 proměnné



Obr. 2-8: Příklad vytvoření Karnaughových map

Časový průběh představuje grafické znázornění, kde logické hodnotě 0 je přiřazena čára o nízké úrovni, označená *L*, a logické hodnotě I čára o vysoké úrovni, označena *H*. Tato forma zadání se nejvíce používá v návrhových systémech pro prvky typu PLD - Programmable Logic Devices. Příklad tohoto zobrazení je na obr. 2-9, přičemž zobrazená funkce je ekvivalentní zápisu: **Časový průběh**

$$f(x_1, x_0) = \mathrm{D}(0,2) \qquad (2\text{-}17)$$



Obr. 2-9: Časový průběh booleovské funkce

2.2.2 Zápis logické funkce přiřazením výrazu

Na základě definice logických operátorů 2-3 je zřejmé, že každá booleovská funkce $f(x_{n-1},...,x_1,x_0)$ může být vyjádřena ve dvou tvarech, a to buď pomocí součtu, anebo součinu jiných jednodušších funkcí.

Nechť pro $k \geq 1$ jsou funkce $f, g_1, g_2, ..., g_k$ rozšířené booleovské funkce o n proměnných. Potom pro **součtový tvar** lze psát **Součtový tvar**

$$f = g_1 + g_2 + ... + g_k \qquad (2\text{-}18)$$

kde g_i (i=1,2,…,k) jsou funkce, které splňují tyto podmínky:

- funkce g_i jsou implikanty (podmnožiny) funkce f,
- každý jednotkový bod funkce f je pokrytý alespoň jedním implikantem.

Jestliže bereme v úvahu nejdříve body oboru, ve kterých má funkce f hodnotu log I, musí mít alespoň jedna funkce g_i hodnotu log I, což vyplývá z definice

operace součtu. V bodech oboru, kde funkce *f* má hodnotu log 0, nesmí mít žádná funkce g_i hodnotu log I. Funkce g_i se rovněž nazývá minterm.

V případě, že funkce g_i bude funkcí vstupních proměnných $g_i = g(x_{n-1},...,x_1,x_0)$ a bude funkcí úplnou, musí být vztah mezi vstupními proměnnými definován operací součinu. Jinými slovy to znamená, že v bodě, kde funkce *f* má hodnotu log I, musí součin logických hodnot vstupních proměnných mít hodnotu log I. Z toho vyplývá, že funkce g_i může obsahovat nejen proměnné přímé, ale i jejich komplementy. Je-li *i*-tý vstupní term charakterizovaný funkcí g_i a tato bude obsahovat proměnnou, která má hodnotu 0, musí být tato komplementována, neboť musí platit, že $g_i = \mathrm{I}$.

K ZAPAMATOVÁNÍ 5

Základní **součtový tvar** funkce *f* je tedy dán **součtem** základních **součinů** přímých nebo negovaných proměnných.

Součinový tvar

Nechť pro $k \geq 1$ jsou funkce $f, h_1, h_2, ..., h_k$ rozšířené booleovské funkce o *n* proměnných. Potom pro **součinový tvar** lze psát:

$$f = h_1 \cdot h_2 \cdot ... \cdot h_k \qquad (2\text{-}19)$$

kde h_i (i=1,2,…,k) jsou funkce, které splňují tyto podmínky:

- funkce h_i jsou implikanty funkce *f*,
- každý nulový bod funkce *f* je pokrytý aspoň jedním implikantem.

Stejně jako v případě součtového tvaru lze i pro součinový tvar psát, že pro nulové body funkce *f* musí mít aspoň jedna funkce h_i hodnotu log 0. V bodech, kde funkce *f* má hodnotu log I, musí mít všechny funkce h_i hodnotu log I. Funkce h_i se někdy nazývá maxterm.

Vzájemný vztah mezi jednotlivými tvary je potom dán negací. Když máme funkci $\overline{f}$ a soubor implikantů $g_1, g_2, ..., g_k$ funkce $\overline{f}$ tak, že platí $\overline{f} = g_1 + g_2 + ... + g_k$, potom podle De Morganových zákonů je

$f = \overline{g_1} \cdot \overline{g_2} \cdot \ldots \cdot \overline{g_k}$. Funkce $\overline{g_i}$ pro $i = 1,2,\ldots,k$ jsou implikanty funkce f a platí:

$$h_i = \overline{g_i} \qquad (2\text{-}20)$$

K ZAPAMATOVÁNÍ 6

Základní **součinový tvar** je dán **součinem** základních **součtů** přímých nebo negovaných proměnných.

ŘEŠENÝ PŘÍKLAD

Pro závislost $f(x_2, x_1, x_0) = 01100110$ napište základní součtový tvar a základní součinový tvar funkce.

Řešení příkladu

Funkci vyjádříme nejdříve kombinační tabulkou:

x_2	x_1	x_0	$f(x_2, x_1, x_0)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Základní součtový tvar je definován pro body, kdy $f = 1$:

Vstupní kombinace			Dílčí součin (minterm)
x_2	x_1	x_0	
0	0	1	$\overline{x_2} \cdot \overline{x_1} \cdot x_0$
0	1	0	$\overline{x_2} \cdot x_1 \cdot \overline{x_0}$
1	0	1	$x_2 \cdot \overline{x_1} \cdot x_0$
1	1	0	$x_2 \cdot x_1 \cdot \overline{x_0}$

Potom součtový tvar funkce je:
$f = \overline{x_2} \cdot \overline{x_1} \cdot x_0 + \overline{x_2} \cdot x_1 \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot x_0 + x_2 \cdot x_1 \cdot \overline{x_0}$.

Základní součinový tvar je definován pro body, kdy $f = 0$:

Vstupní kombinace			Dílčí součet (maxterm)
x_2	x_1	x_0	
0	0	0	$x_2 + x_1 + x_0$
0	1	1	$x_2 + \overline{x_1} + \overline{x_0}$
1	0	0	$\overline{x_2} + x_1 + x_0$
1	1	1	$\overline{x_2} + \overline{x_1} + \overline{x_0}$

Odtud $f = (x_2 + x_1 + x_0) \cdot (x_2 + \overline{x_1} + \overline{x_0}) \cdot (\overline{x_2} + x_1 + x_0) \cdot (\overline{x_2} + \overline{x_1} + \overline{x_0})$.

*

2.3 Minimalizace Booleovských funkcí

Minimalizace Booleovských funkcí

Logická funkce vyjádřena úplnou základní součtovou nebo součinovou formou z pravdivostní tabulky není jediným možným vyjádřením realizace popisované logické funkce. Tato základní logická funkce je samozřejmě správná, ale z hlediska praktické realizace není minimální. Obsahuje nadbytečné prvky, které neovlivní logický výsledek rovnice.

Většinou lze nalézt algebraické vyjádření, které povede ke snížení počtu operací. Nalezené vyjádření je mnohdy podstatně jednodušší a vhodnější pro realizaci. Snížením počtu operací (součtů nebo součinů) se vyhneme složitému zápisu funkce a pro praxi tím docílíme zmenšení složitosti obvodu. Principem minimalizace je tedy odstranění přebytečných proměnných.

Výsledkem této činnosti (minimalizace) je proto nalezení minimálního výrazu. Minimálním výrazem rozumíme takový výraz, který má nejmenší četnost vyskytujících se proměnných. Četností proměnných v tomto případě rozumíme algebraický součet proměnných vyskytujících se ve výrazu, a to bez ohledu na index a bez ohledu na skutečnost, zda proměnná je v přímé formě nebo ve formě negované.

Minimálnost

Pojem minimálnosti lze definovat z mnoha hledisek. Jedním z hledisek může být počet operací ovlivňující složitost zapojení, tedy počet prvků součástkové základny potřebný pro praktickou realizaci, který ovlivňuje celkovou finanční náročnost realizace. Počet prvků má vliv i na spolehlivost obvodu a ovlivňuje celkovou dobu průchodu signálu obvodem. Tato doba se nazývá zpoždění a představuje reakci výstupní proměnné na změnu logické hodnoty vstupní proměnné. Je patrné, že čím větší počet stupňů má výsledné zapojení, tím je také větší doba zpoždění.

ÚKOL K ZAMYŠLENÍ 1

Zvažte, jaké další problémy mohou nastat pokud by byla praktická realizace provedena na základě neminimalizovaného výrazu.

Minimálnost

Metody minimalizace

Pro vlastní minimalizaci Booleovských funkcí vyjádřených pomocí základního booleovského výrazu se nejčastěji používají následující metody:

- algebraická minimalizace,
- minimalizace pomocí Karnaughových map,
- minimalizační metody Mc-Cluskey

2.3.1 Algebraická minimalizace

Metoda algebraické minimalizace vychází z aplikace postulátů Booleovy algebry, ze znalosti teorémů a jejich aplikace na zápis logické funkce. Úroveň konečného zjednodušení je dána především zkušenostmi a intuicí. Z tohoto důvodu se tato metoda využívá pro menší počet proměnných. Metoda algebraické minimalizace je vhodná pro maximálně 4 proměnné. Pro větší počet proměnných se tato metoda nedoporučuje a je vhodnější použít jiný postup (např. metoda Mc-Cluskey).

Algebraická minimalizace

ŘEŠENÝ PŘÍKLAD

Vyjádřete v úplném součtovém a součinovém tvaru funkci F_0 (x_2, x_1, x_0) = D(3, 4, 6, 7). Minimalizujte ji pomocí Booleovy algebry.

Řešení příkladu

Nejprve sestavíme na základě zadání pravdivostní tabulku:

index	x_2	x_1	x_0	$\boldsymbol{F_0}$	minterm	maxterm
0	0	0	0	**0**		$x_2 + x_1 + x_0$
1	0	0	1	**0**		$x_2 + x_1 + \overline{x}_0$
2	0	1	0	**0**		$x_2 + \overline{x}_1 + x_0$
3	0	1	1	**1**	$\overline{x}_2 \cdot x_1 \cdot x_0$	
4	1	0	0	**1**	$x_2 \cdot \overline{x}_1 \cdot \overline{x}_0$	
5	1	0	1	**0**		$\overline{x}_2 + x_1 + \overline{x}_0$
6	1	1	0	**1**	$x_2 \cdot x_1 \cdot \overline{x}_0$	
7	1	1	1	**1**	$x_2 \cdot x_1 \cdot x_0$	

.

Na základě pravdivostní tabulky poté můžeme psát logickou funkci v úplném součtovém tvaru:

$$F_0 = \overline{x}_2 \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_2 \cdot x_1 \cdot \overline{x}_0 + x_2 \cdot x_1 \cdot x_0$$

Tuto logickou funkci zjednodušíme pomocí algebraických úprav:

$$F_0 = x_1 \cdot x_0 \cdot \left(\overbrace{\overline{x}_2 + x_2}^{1} \right) + x_2 \cdot \overline{x}_0 \cdot \left(\overbrace{\overline{x}_1 + x_1}^{1} \right) = \underline{\underline{x_1 \cdot x_0 + x_2 \cdot \overline{x}_0}}$$

Na základě pravdivostní tabulky můžeme také psát logickou funkci v úplném součinovém tvaru:

$$F_0 = (x_2 + x_1 + x_0) \cdot (x_2 + x_1 + \overline{x}_0) \cdot (x_2 + \overline{x}_1 + x_0) \cdot (\overline{x}_2 + x_1 + \overline{x}_0)$$

Tuto logickou funkci zjednodušíme pomocí zákonů Booleovy algebry na minimalizovaný tvar (použijeme zákona spojení):

$$F_0 = \underline{\underline{(x_2 + x_0) \cdot (x_1 + \overline{x}_0)}}$$

*

ŘEŠENÝ PŘÍKLAD

Minimalizujte a vyjádřete v úplném součtovém a součinovém tvaru funkci F_0 (x_2, x_1, x_0) = K (1, 3, 4, 6). Získané výsledky porovnejte, případnou úpravou dokažte, že jsou ekvivalentní.

Řešení příkladu

Opět sestavíme pravdivostní tabulku funkce F_0.

index	x_2	x_1	x_0	$\boldsymbol{F_0}$	minterm	maxterm
0	0	0	0	**1**	$\overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0$	
1	0	0	1	**0**		$x_2 + x_1 + \overline{x}_0$
2	0	1	0	**1**	$\overline{x}_2 \cdot x_1 \cdot \overline{x}_0$	
3	0	1	1	**0**		$x_2 + \overline{x}_1 + \overline{x}_0$
4	1	0	0	**0**		$\overline{x}_2 + x_1 + x_0$
5	1	0	1	**1**	$x_2 \cdot \overline{x}_1 \cdot x_0$	
6	1	1	0	**0**		$\overline{x}_2 + \overline{x}_1 + x_0$
7	1	1	1	**1**	$x_2 \cdot x_1 \cdot x_0$	

.

Sestavíme úplný součtový tvar funkce, který zjednodušíme :

$$F_0 = \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + x_2 \cdot \overline{x}_1 \cdot x_0 + x_2 \cdot x_1 \cdot x_0$$

$$F_0 = \overline{x}_2 \cdot \overline{x}_0 \cdot \left(\overbrace{x_1 + \overline{x}_1}^{1} \right) + x_2 \cdot x_0 \cdot \left(\overbrace{x_1 + \overline{x}_1}^{1} \right) = \underline{\underline{\overline{x}_2 \cdot \overline{x}_0 + x_2 \cdot x_0}}$$

Sestavíme úplný součinový tvar funkce, který zjednodušíme (můžeme opět aplikovat zákon spojení jako v předchozím příkladu):

$$F_0 = \left(x_2 + x_1 + \overline{x}_0\right) \cdot \left(x_2 + \overline{x}_1 + \overline{x}_0\right) \cdot \left(\overline{x}_2 + x_1 + x_0\right) \cdot \left(\overline{x}_2 + \overline{x}_1 + x_0\right)$$

$$F_0 = \underline{\underline{\left(x_2 + \overline{x}_0\right) \cdot \left(\overline{x}_2 + x_0\right)}}$$

Tento výraz roznásobíme a upravíme:

$$F_0 = \left(\overbrace{x_2 \cdot \overline{x}_2}^{0} + x_2 \cdot x_0 + \overline{x}_0 \cdot \overline{x}_2 + \overbrace{\overline{x}_0 \cdot x_0}^{0} \right) = \underline{\underline{\left(x_2 \cdot x_0 + \overline{x}_0 \cdot \overline{x}_2\right)}}$$

Touto úpravou jsme dokázali, že funkce získaná na základě úpravy úplného součtového tvaru je shodná s funkcí získanou na základě úpravy úplného součinového tvaru.

*

ÚLOHY K ŘEŠENÍ 2-4

Vyjádřete v úplném součtovém a součinovém tvaru zadanou funkci a minimalizujte ji pomocí Booleovy algebry:

a) $F_1\ (x_2, x_1, x_0) = \mathrm{K}(0,7)$

b) $F_2\ (x_3, x_2, x_1, x_0) = (\mathrm{CD33})_{16}$

PRŮVODCE STUDIEM 2

Po prostudování této kapitoly jste připraveni vypracovat bod a) první samostatné práce.

2.3.2 Minimalizace pomocí Karnaughových map

Způsob sestavení mapy zajišťuje určité kódování proměnných (Grayův kód), čímž je docíleno toho, že sousední čtverečky se v mapě liší pouze v jedné proměnné. Tuto podmínku splňují i čtverečky krajních řádků a sloupců. Princip zjednodušení spočívá v grafickém sloučení sousedních čtverečků a jejich popsání pomocí vstupních proměnných. Grafický postup v podstatě odstraňuje méně přehledné zápisy dílčích součtů nebo součinů.

Minimalizace logické funkce pomocí Karnaughovy mapy se vykonává na základě následujících pravidel:

- musí být pokryty všechny čtverečky, v nichž funkce nabývá hodnotu log I (pro součtový tvar) nebo hodnotu log 0 (pro součinový tvar),
- smyčka musí zahrnovat 2^n sousedních čtverečků, (kde n = 0,1,2,3,…..),
- snažíme se dosáhnout minimálního počtu smyček (minimální počet dílčích součinů nebo součtů),
- snažíme se dosáhnout minimálního počtu proměnných v jednotlivých součinech nebo součtech.

Tímto se z daného součinu, resp. součtu vyloučí proměnná, která mění svůj stav.

V případě neúplné funkce je postup stejný, pouze nedefinované hodnotě se vytvořenou smyčkou přiřadí konkrétní logická hodnota log I nebo log 0.

ŘEŠENÝ PŘÍKLAD

Zjednodušte funkci $f(x_3, x_2, x_1, x_0) = \mathrm{CD33}_{16}$

Řešení příkladu

Pro danou funkci sestavíme pravdivostní tabulku a odpovídající Karnaughovu mapu:

Pravdivostní tabulka

n	x_3	x_2	x_1	x_0	f
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

Karnaughova mapa

$x_3 x_0 \backslash x_2 x_1$			
	1	0	0	1
	1	0	0	1
	0	1	1	0
	1	1	1	0

Součtový tvar - smyčkou označíme sousední čtverečky, v nichž funkce nabývá hodnotu log I a tyto smyčky vypíšeme jako dílčí součiny.

Součtový tvar

Mapa s vyznačením dílčích součinů

$x_3 x_0 \backslash x_2 x_1$					
	1	0	0	1	B
B	1	0	0	1	
	0	1_1	1_2	0	
	1	1_3	1_4	0	
	C	A			

Dílčí součin označený smyčkou A je tvořen čtverečky s označením $_{1,2,3,4}$ a tedy platí:

$$A = x_3\bar{x}_2x_1x_0 + x_3x_2x_1x_0 + x_3\bar{x}_2x_1\bar{x}_0 + x_3x_2x_1\bar{x}_0 =$$
$$= x_3x_1x_0(\bar{x}_2 + x_2) + x_3x_1x_0(\bar{x}_2 + x_2) = x_3x_1(\bar{x}_0 + x_0) = x_3x_1$$

Pro smyčky B a C lze z mapy přímo psát:

$$B = \bar{x}_3\bar{x}_1$$

$$C = x_3\bar{x}_2\bar{x}_0 .$$

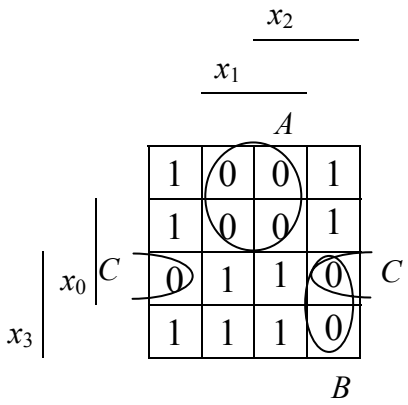
Výsledná funkce má potom tvar:

$$f = A + B + C = x_3x_1 + \bar{x}_3\bar{x}_1 + x_3\bar{x}_2\bar{x}_0$$

Minimální součtový tvar funkce

Součinový tvar - smyčkou označíme sousední čtverečky, v nichž funkce nabývá hodnotu log 0 a tyto smyčky vypíšeme jako dílčí součty.

Součinový tvar



Mapa s vyznačením dílčích součtů

Pro jednotlivé smyčky platí:

$$A = x_3 + \overline{x}_1$$

$$B = \overline{x}_3 + \overline{x}_2 + x_1$$

$$C = \overline{x}_3 + x_1 + \overline{x}_0$$

Výslednou funkci dostaneme jako součin dílčích součtů:

$$f = A \cdot B \cdot C = \left(x_3 + \overline{x_1}\right) \cdot \left(\overline{x_3} + \overline{x_2} + x_1\right) \cdot \left(\overline{x_3} + x_1 + \overline{x_0}\right)$$

Minimální součinový tvar funkce

Správnost řešení potvrdíme převedením součinového tvaru na součtový roznásobením závorek.

$$f = (x_3\overline{x}_3 + x_3\overline{x}_2 + x_3x_1 + \overline{x}_3\overline{x}_1 + \overline{x}_2\overline{x}_1 + \overline{x}_1x_1).(\overline{x}_3 + x_1 + \overline{x}_0) =$$
$$= (\overline{x}_3x_3\overline{x}_2 + \overline{x}_3x_3x_1 + \overline{x}_3\overline{x}_1 + \overline{x}_3\overline{x}_2\overline{x}_1 + x_3\overline{x}_2x_1 + x_3x_1 + \overline{x}_3\overline{x}_1x_1 +$$
$$+ \overline{x}_2\overline{x}_1x_1 + x_3\overline{x}_2\overline{x}_0 + \overline{x}_3\overline{x}_1\overline{x}_0 + \overline{x}_2\overline{x}_1\overline{x}_0 =$$
$$= \overline{x}_3\overline{x}_0(1 + \overline{x}_2 + \overline{x}_0) + x_3x_1(\overline{x}_2 + 1) + x_3\overline{x}_2\overline{x}_0 + \overline{x}_2\overline{x}_1x_0 =$$
$$= \overline{x}_3\overline{x}_1 + x_3x_1 + x_3\overline{x}_2\overline{x}_0 + \overline{x}_2\overline{x}_1\overline{x}_0$$

a po aplikování absorpce konsenzu $f = x_3x_1 + \overline{x}_3\overline{x}_1 + x_3\overline{x}_2\overline{x}_0$

*

ČÁST PRO ZÁJEMCE 1

Kapitola 3 je věnována praktické realizaci logického kombinačního obvodu. Jako praktický příklad návrhu kombinačního obvodu je zde uveden *NÁPOJOVÝ AUTOMAT*. Už v tuto chvíli byste byli schopni vyřešit v tomto příkladu body a) a b).

ŘEŠENÝ PŘÍKLAD

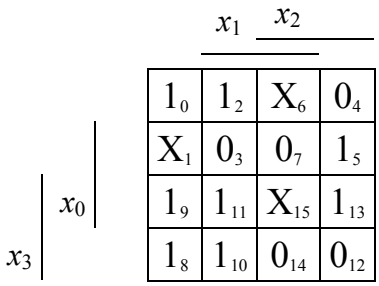
Pro funkci $F(x_3, x_2, x_1, x_0) = \text{X01011110X1001X1}$ sestavte pomocí Karnaughovy mapy minimální součtový a součinový tvar.

Řešení příkladu

Pro danou funkci sestavíme pravdivostní tabulku a odpovídající Karnaughovu mapu:

n	x_3	x_2	x_1	x_0	f
0	0	0	0	0	1
1	0	0	0	1	X
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	X
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	X

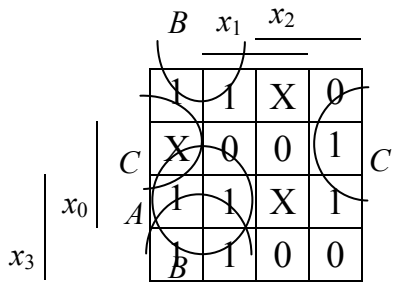
Pravdivostní tabulka



Karnaughova mapa

Pro součtový tvar zvolíme smyčky:

Součtový tvar



Mapa s vyznačením smyček pro součtový tvar funkce

Pro jednotlivé smyčky můžeme psát:

$A = x_3\bar{x}_2 \qquad B = \bar{x}_2\bar{x}_0 \qquad C = \bar{x}_1 x_0$

Výsledný minimální součtový tvar funkce je:

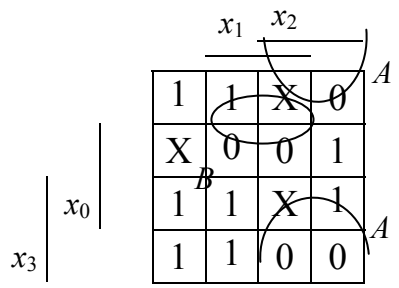
$$F = x_3\bar{x}_2 + \bar{x}_2\bar{x}_0 + \bar{x}_1 x_0$$

Minimální součtový tvar funkce

Ze zvolených smyček je zřejmé, že term s pořadovým číslem 1 bude mít hodnotu log I a zbývající nedefinované hodnoty budou mít hodnotu log 0.

Smyčky pro součinový tvar:

Součinový tvar



Mapa s vyznačenými smyčkami pro součinový tvar funkce

Dílčí smyčky:

$A = \bar{x}_2 + x_0 \qquad B = x_3 + \bar{x}_1 + \bar{x}_0$

Výsledná funkce má potom tvar:

$$F = (\bar{x}_2 + x_0)\cdot(x_3 + \bar{x}_1 + \overline{x_0})$$

Minimální součinový tvar funkce

Z obrázku je zřejmé, že pouze term s pořadovým číslem 6 bude mít hodnotu log 0. Porovnáme-li výsledné výrazy součtového a součinového tvaru lze konstatovat, že se ani po potřebné úpravě nebudou shodovat. Rozdílnost vyplývá z přiřazení různých logických hodnot místo hodnoty neurčité.

*

ÚLOHA K ŘEŠENÍ 2-5

Minimalizujte funkci v součtovém tvaru pomocí Karnaughovy mapy $f(x_2, x_1, x_0) = 11101000$.

ÚLOHA K ŘEŠENÍ 2-6

Minimalizujte funkci v součinovém tvaru pomocí Karnaughovy mapy $f(x_3, x_2, x_1, x_0) = 110010100100111$.

2.3.3 Minimalizace metodou Mc-Cluskey

Tuto minimalizační metodu je výhodné používat v případě funkcí s více než čtyřmi vstupními proměnnými. Metoda vychází ze vzájemného porovnávání všech vstupních vektorů, způsobem každý s každým, a to ve dvou etapách. V první etapě se vyhledají a roztřídí vstupní vektory do skupin podle počtu lišících se logických hodnot vstupních proměnných. Ve druhé etapě se porovnávají vektory, které se mohou lišit jen v jedné proměnné. V případě, že se dva porovnávané vektory liší v jedné proměnné, lze konstatovat, že na této proměnné nezávisí a je možné tyto vektory sloučit do jednoho. Jestliže se dva vektory liší v proměnné x_i, lze obecně pro tyto vektory psát :

$$\begin{aligned} & x_n \cdot \ldots \cdot x_{i+1} \cdot x_i \cdot x_{i-1} \cdot \ldots \cdot x_0 + x_n \cdot \ldots \cdot x_{i+1} \cdot \overline{x_i} \cdot x_{i-1} \cdot \ldots \cdot x_0 = \\ & = x_n \cdot \ldots \cdot x_{i+1} \cdot \left(x_i + \overline{x_i}\right) \cdot x_{i-1} \cdot \ldots \cdot x_0 = x_n \cdot \ldots \cdot x_{i+1} \cdot x_{i-1} \cdot \ldots \cdot x_0 \end{aligned} \qquad (2\text{-}21)$$

Dalším porovnáním vektorů zůstanou pouze ty, které není možné již dále zjednodušit a jsou zahrnuty do výsledného tvaru funkce. Tyto vektory nazýváme implikanty. Vzhledem k tomu, že implikant může současně pokrýt více vektorů, je pro minimální tvar funkce nutné provést kontrolu pokrytí vektory. Na základě této kontroly se potom vyloučí stejné implikanty a sestaví se výraz pro minimální tvar funkce. Metodu je možné aplikovat jak pro sestavení funkce v součtovém tvaru, tak i v součinovém tvaru. Postup metody bude zřejmý z následujícího zjednodušení rozšířené funkce

$$F(x_3, x_2, x_1, x_0) = \mathrm{D}(0,2,5,8,9,10,11,13(1,6,15)) \qquad (2\text{-}22)$$

Pravdivostní tabulka funkce je uvedena na obr. 2-10. Zároveň je rozšířena o sloupec obsahující index vstupní *n*-tice, označený *K*.

Při sestavování výrazu funkce v součtovém tvaru se provádí porovnávání vstupních vektorů, v nichž má funkce hodnotu log I. Vektory, v nichž funkce není definována se do řešení nemusí zahrnout, ale v tom případě je jim apriorně přiřazena výstupní hodnota log 0 a nemohou být využity při zjednodušování. Z tohoto důvodu jsou do řešení zahrnuty a o jejich případném vypuštění se rozhodne až při kontrole pokrytí. Řešení tedy probíhá jako by tyto vektory měly logickou hodnotu 1. Z pravdivostní tabulky z obr. 2-10 proto dostaneme pravdivostní tabulku pro součtový tvar funkce, obr. 2-11.

K	x_3	x_2	x_1	x_0	f
0	0	0	0	0	1
1	0	0	0	1	X
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	X
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	X

Obr. 2-10: Pravdivostní tabulka kombinačního obvodu

K	x_3	x_2	x_1	x_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
5	0	1	0	1
6	0	1	1	0
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
13	1	1	0	1
15	1	1	1	1

Obr. 2-11: Pravdivostní tabulka – součtový tvar

První etapa porovnávání

První etapa spočívá v porovnání vektorů z obr. 2-11 a vytváření skupin, které se mohou lišit pouze v jedné proměnné. Lze proto jednoznačně oddělit skupiny vektorů, lišících se od sebe ve více proměnných. Kritériem pro vytvoření takovýchto skupin může být např. počet hodnot log I ve vektoru. Výsledek prvního porovnání je uveden na obr. 2-12.

K	x_3	x_2	x_1	x_0	Počet hodnot 1
0	0	0	0	0	žádná
1	0	0	0	1	jedna
2	0	0	1	0	
8	1	0	0	0	
5	0	1	0	1	dvě
6	0	1	1	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	tři
13	1	1	0	1	
15	1	1	1	1	čtyři

Obr. 2-12: První etapa porovnávání

Druhá etapa porovnávání

Význam uspořádání vektorů podle obr. 2-12 spočívá v tom, že postačuje porovnávat pouze vektory jedné skupiny s vektory skupiny vyšší. V našem případě porovnáváme vektor s indexem 0 s vektory s indexy 1, 2 a 8, dále vektor s indexem 1 s vektory s indexy 5, 6, 9 a 10, vektor s indexem 2 s vektory s indexy 5, 6, 9, 10 atd. až po porovnání vektoru s indexem 13 s vektorem s indexem 15. Můžeme-li vektory sloučit, zapíšeme oba indexy do sloupce K a logickou hodnotu pro lišící se proměnnou označíme symbolem „-“. Výsledek tohoto porovnávání je uveden na obr. 2-13. V případě, kdy některý z vektorů není možné sloučit s jiným, označíme jej jako implikant a zahrneme do tabulky pokrytí.

Implikant

Systém porovnávání, každý vektor s každým, připouští výskyt stejných vektorů v jedné skupině. Projeví se to rovněž ve sloupci K, kde jsou všechny indexy sloučených vektorů, ale v odlišném pořadí. Je-li tomu tak, můžeme libovolný z implikantů vzhledem k duplicitě z dalšího porovnávání vyloučit. V tomto případě tomu tak není, a proto přikročíme k dalšímu porovnávání vektorů, čímž získáme tabulku uvedenou na obr. 2-14.

K	x_3	x_2	x_1	x_0	Počet hodnot 1	
0, 1	0	0	0	-	žádná	
0, 2	0	0	-	0		
0, 8	-	0	0	0		
1, 5	0	-	0	1	jedna	
1, 9	-	0	0	1		
2, 6	0	-	1	0		(A)
2, 10	-	0	1	0		
8, 9	1	0	0	-		
8, 10	1	0	-	0		
5, 13	-	1	0	1	dvě	
9, 11	1	0	-	1		
9,13	1	-	0	1		
10, 11	1	0	1	-		
11, 15	1	-	1	1	tři	
13, 15	1	1	-	1		

Obr. 2-13: Vzájemné porovnávání vektorů ve skupinách

K	x_3	x_2	x_1	x_0	Počet hodnot 1
0,1,8,9	-	0	0	-	žádná
0,2,8,10	-	0	-	0	
0,8,1,9	-	0	0	-	
0,8,2,10	-	0	-	0	
1,5,9,13	-	-	0	1	jedna
1,9,5,13	-	-	0	1	
8,9,10,11	1	0	-	-	
8,10,9,11	1	0	-	-	
9,11,13,15	1	-	-	1	dvě
9,13,11,15	1	-	-	1	

Obr. 2-14: Vzájemné porovnávání vektorů ve skupinách

Z obr. 2-13 jsme ke sloučení s jiným vektorem nepoužili vektor s indexy 2, 6. Jde tedy o implikant a označíme ho písmenem (A). V obr. 2-14 se nacházejí shodné vektory, např. 0, 1, 8, 9 a 0, 8, 1, 9. K dalšímu slučování použijeme pouze jeden z nich.

Na obr. 2-15 je potom uvedena tabulka, ze které jsou vypuštěny stejné vektory. Jestliže pokračujeme v porovnávání zjistíme, že další sloučení není možné. Znamená to tedy, že jednotlivé vektory jsou implikanty a označíme je dalšími písmeny abecedy (B-F).

Uvedeným postupem jsme nalezli všechny možné implikanty, které lze použít k sestavení booleovského výrazu. Dalším úkolem je z těchto odvozených implikantů vybrat pouze ty, které jsou nezbytné k sestavení minimálního výrazu odpovídajícího dané booleovské funkci.

Hledání minimálního výrazu vychází z tabulky pokrytí sestavené pro vstupní vektory, v nichž má booleovská funkce hodnotu log I a X. Řádky tabulky pokrytí, obr. 2-16, odpovídají odvozeným implikantům a sloupce odpovídají indexům vstupních vektorů, ve kterých funkce nabývá hodnoty log I a X. Pořadí indexů ve sloupcích je výhodné volit tak, že nejdříve jsou uvedeny indexy, v nichž má funkce hodnotu log I a potom hodnotu X. Obsah tabulky potom udává, jak daný implikant pokrývá požadované vstupní vektory, přičemž pokrytí je vyznačeno symbolem X.

Vzhledem k tomu, že k sestavení minimálního booleovského výrazu má rozhodující vliv část tabulky, kde funkce nabývá hodnotu log I, je druhá část tabulky pouze informativní. Pracujeme proto s první částí tabulky a význam druhé části nastává v okamžiku, kdy je zapotřebí rozhodnout se mezi více implikanty pokrývající shodné indexy v první části tabulky. V tomto případě je zapotřebí volit implikant pokrývající více indexů v celé tabulce.

Z tabulky pokrytí je zřejmé, že některý index pokrývá jediný implikant (index 5 a implikant D) a některé indexy můžeme pokrýt několika implikanty. V prvním případě se jedná o tzv. nevyhnutelný implikant, to je implikant, který musí být obsažen ve výsledném výrazu funkce. Jako nevyhnutelný implikant vzhledem k indexu 5 je implikant (D), čímž zároveň pokryje indexy 9 a 13. Pro pokrytí indexu 2 můžeme volit mezi implikantem (A) a (C). Vzhledem k většímu počtu pokrytí indexů v první části tabulky volíme (C), čímž zároveň pokryjeme indexy 0, 8 a 10. Jediný index, který nemá pokrytí implikantem (C) a (D) je index 11. Můžeme proto volit mezi implikantem (E) a (F). Protože každý implikant pokrývá 4 indexy celé tabulky, musí být obě řešení ekvivalentní.

Výsledný minimální tvar funkce potom bude:

$$^{1}F = (C) + (D) + (E) \quad \text{nebo} \qquad (2\text{-}23)$$

$$^{2}F = (C) + (D) + (F)$$

K	x_3	x_2	x_1	x_0	Počet hodnot 1	
0,1,8,9	-	0	0	-	žádná	(B)
0,2,8,9	-	0	-	0		(C)
1,5,9,13	-	-	0	1	jedna	(D)
8,9,10,11	1	0	-	-		(E)
9,11,13,15	1	-	-	1	dvě	(F)

Obr. 2-15: Vzájemné porovnávání vektorů ve skupinách

F=1,X	0	2	5	8	9	10	11	13	1	6	15
A		X								X	
B	X			X	X				X		
C	X	X		X		X					
D			X		X			X	X		
E				X	X	X	X				
F					X		X	X			X

Obr. 2-16: Tabulka pokrytí

Výpis dílčích součinů pro jednotlivé implikanty se provede z tabulky uvedené na obr. 2-13 a obr. 2-15, přičemž platí:

$$
\begin{aligned}
(C) &= \overline{x_2} \cdot \overline{x_0} \\
(D) &= \overline{x_1} \cdot x_0 \\
(E) &= x_3 \cdot \overline{x_2} \\
(F) &= x_3 \cdot x_0
\end{aligned}
\qquad (2\text{-}24)
$$

Dosazením *(2-24)* do *(2-23)* dostaneme minimální tvary funkcí:

$$
{}^{1}F = \overline{x_2} \cdot \overline{x_0} + \overline{x_1} \cdot x_0 + x_3 \cdot \overline{x_2} \qquad (2\text{-}25)
$$

$$^{2}F = \overline{x_2} \cdot \overline{x_0} + \overline{x_1} \cdot x_0 + x_3 \cdot x_0$$

Je třeba konstatovat, že oba výrazy realizují stejnou rozšířenou booleovskou funkci a jsou minimální. Stejného výsledku lze dosáhnout i metodou zjednodušování pomocí Karnaughových map.

Pro hledání minimálního výrazu funkce v součinovém tvaru se vychází z vektorů, v nichž má funkce hodnotu log 0 a X.

ÚLOHA K ŘEŠENÍ 2-7

Vyjádřete v součinovém tvaru rozšířenou funkci $F(x_3,x_2,x_1,x_0)$ = D (6,7,9,11,13,15(1,4)) a minimalizujte ji pomocí metody Mc-Cluskey.

ÚLOHA K ŘEŠENÍ 2-8

Vyjádřete v součtovém tvaru funkci $F(x_3,x_2,x_1,x_0) = (114734)_8$ a minimalizujte ji pomocí metody Mc-Cluskey.

PRŮVODCE STUDIEM 3

Po prostudování této kapitoly jste připraveni vypracovat bod b) samostatné práce.

SHRNUTÍ KAPITOLY KOMBINAČNÍ OBVODY

Shrnutí

V této kapitole byl posluchač seznámen se základními pravidly Booleovy algebry a s jejich využitím při zjednodušování výrazů nebo při úpravě na žádaný tvar. Dále byl posluchač seznámen s problematikou booleovských funkcí, se způsoby jejich zápisu a s metodami minimalizace logických funkcí.

Protože logické systémy se vyznačují tím, že jejich jednotlivé proměnné nabývají pouze dvou hodnot, je zřejmé, že znalost Booleovy algebry je nutná pro studium všech následujících kapitol, které se věnují kombinačním a sekvenčním logickým obvodům.

3 LOGICKÉ KOMBINAČNÍ OBVODY

ČAS POTŘEBNÝ KE STUDIU

Předpokládaný čas k prostudování kapitoly Kombinační obvody je **15 hodin**.

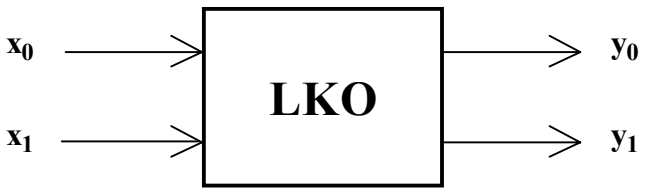
RYCHLÝ NÁHLED DO PROBLEMATIKY KAPITOLY LOGICKÉ KOMBINAČNÍ OBVODY

Logický kombinační obvod je takový logický obvod, jehož vektor výstupních proměnných závisí pouze na logických hodnotách vektoru vstupních proměnných, tzn. určité kombinaci vstupních proměnných odpovídá pouze jedna kombinace výstupních proměnných. Výstupní proměnné tedy nezávisí na stavu obvodu, pokud neuvažujeme přechodový režim – viz. obr. 3-1.

Kompletní návrh kombinačních obvodů předpokládá:

- z obecných požadavků na chování obvodu určit počet vstupních a výstupních proměnných a vytvořit pravdivostní tabulku,
- z pravdivostní tabulky určit logické funkce popisující chování obvodu, provést případnou minimalizaci se zřetelem na soubor logických členů pro vytvoření struktury obvodu,
- v případě potřeby vyšetřit minimalizované výrazy v přechodových režimech a eliminovat případný vznik hazardů,
- z navržených výrazů realizovat logický kombinační obvod,
- v případě nutnosti analyzovat navržený obvod a ověřit splnění požadavků.

V předchozích kapitolách byly probrány první dva body a platnost uvedených metod z hlediska logických systémů je zcela obecná. Jsou proto platné jak pro realizace pomocí mechanických prvků, tak i elektrických, elektronických a integrovaných obvodů.



Obr. 3-1: Princip činnosti logického kombinačního obvodu.

CÍLE KAPITOLY LOGICKÉ KOMBINAČNÍ OBVODY

- Budete umět navrhnout logický kombinační obvod pomocí hradel NAND, NOR, AND-OR-INVERT nebo pomocí paměťového prvku,
- získáte přehled o základních obvodech TTL,
- budete schopni zvolit počet vstupních a výstupních proměnných potřebných k popisu chování konkrétního logického obvodu,
- budete schopni sestavit booleovské funkce popisující chování kombinačního obvodu a realizovat libovolný kombinační obvod pomocí hradel a pomocí paměťových prvků,
- budete schopni odstranit statický a dynamický hazard.

KLÍČOVÁ SLOVA KAPITOLY LOGICKÉ KOMBINAČNÍ OBVODY

Logický kombinační obvod, integrovaný obvod, obvody TTL, AND, NOR, AND-OR-INVERT, paměť ROM, statický hazard, dynamický hazard, souběhový hazard

3.1 Návrh logických kombinačních obvodů

Z pohledu praxe se realizace rozdělují na realizace pomocí:

- mechanických prvků,
- pneumatických prvků,
- elektromagnetických relé,
- tranzistorů a diod
- integrovaných obvodů.

Základní technické parametry těchto skupin jsou charakterizované rychlostí vykonání jedné operace a potřebného příkonu pro tuto operaci, viz. obr. 3-2.

Vzhledem k určení předmětu se budeme dále zabývat pouze realizací pomocí integrovaných obvodů. Při elektronické realizaci se logické hodnoty vyjadřují pomocí dvou hodnot napětí, kterým se přiřazuje označení *L* a *H*. Úroveň *L* je přiřazena nižšímu napětí a úroveň *H* napětí vyššímu. Přiřazení těchto úrovní logickým hodnotám je obecně možné dvojím způsobem a jednotlivé způsoby se nazývají pozitivní nebo negativní logika. U pozitivní logiky je hodnota log I přiřazena úrovni *H* a hodnota log 0 úrovni *L*. U negativní logiky je tomu opačně, hodnotě log I je přiřazena úroveň *L* a hodnotě log 0 úroveň *H*. Protože se v praxi většinou používá pozitivní logika, bude v dalším výkladu uvažováno pouze s touto logikou.

skupina:	doba	příkon
mechanické prvky	10 s	10 kW
pneumatické prvky	10 ms	1W
elektromagnetické relé	100 ms	do 1 W
tranzistory a diody	10 ns	10 mW
integrované obvody	10 ns	1 mW

Obr. 3-2: Porovnání energetické náročnosti různých logických prvků

3.2 Číslicové integrované obvody

Integrované logické obvody mají různé modifikace. Některé z nich si uveďme:

- obvody TTL (Tranzistor Tranzistor logic) - klasické, standardní obvody,
- obvody LTTL (Low-power Tranzistor Tranzistor logic) - obvody s malou spotřebou,
- obvody HTTL (High-speed Tranzistor Tranzistor logic) - obvody s vysokou rychlostí spínání,
- obvody STTL (Schottkyho Tranzistor Tranzistor logic) - obvody se Schottkyho diodami,
- obvody LSTTL (Low-power Schottky Tranzistor Tranzistor logic) - kombinované obvody; jak je zřejmé z názvu, kombinace je vytvořena z obvodů s malou spotřebou a z obvodů se Schottkyho diodami.

Vztah mezi vstupními a výstupními napěťovými úrovněmi je dán převodní charakteristikou. Jsou definovány napěťové úrovně, které odpovídají jednotlivým logickým stavům. Logické 1 na výstupu odpovídá napětí 2,4 V až 5 V, na vstupu 2 V až 5 V. Logické 0 na výstupu odpovídá napětí 0 V až 0,4 V a na vstupu 0 V až 0,8 V. Je zřejmý rozdíl mezi vstupními a výstupními napětími odpovídajícími logickým hodnotám. Toto je z důvodů šumové imunity, resp. statické odolnosti proti rušení, které může mít vliv na velikost napětí.

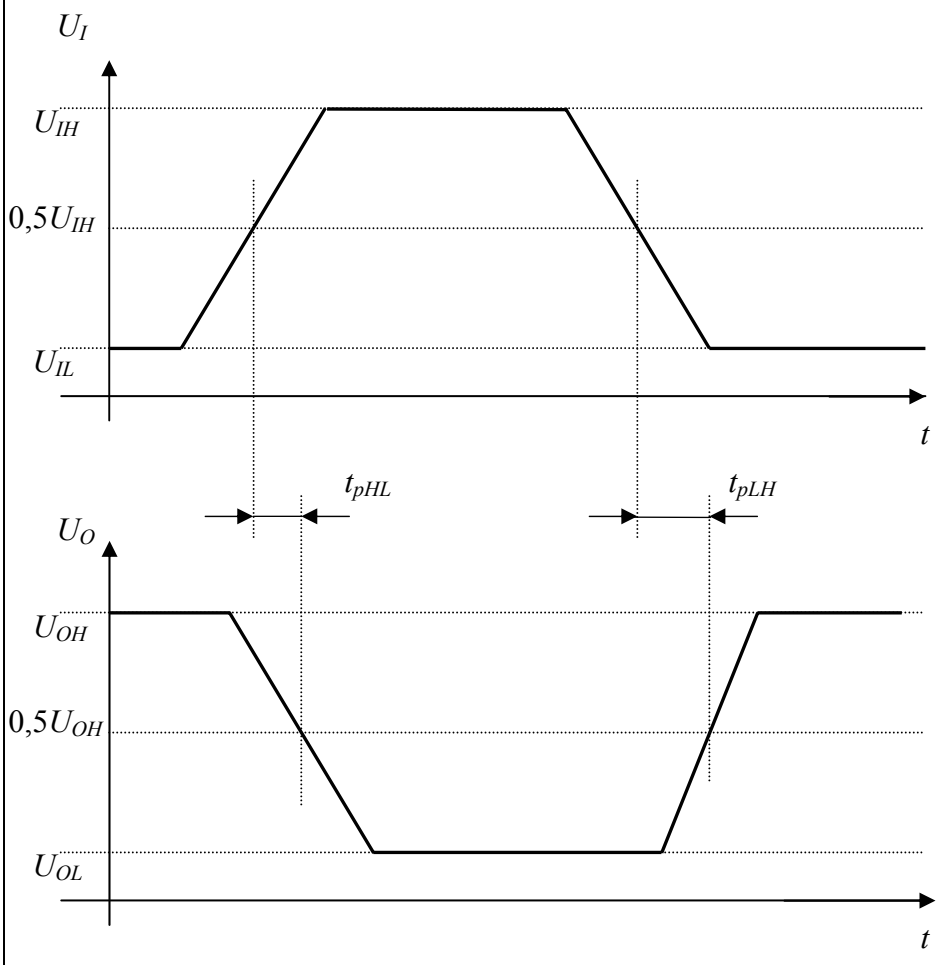
Další důležitý statický parametr hradla je jeho logický zisk *N*, který udává kolik vstupů hradel lze připojit k jednomu výstupu hradla. TTL logika má různé modifikace, viz. tabulka na obr. 3-3, a proto logický zisk je uváděn vždy pro každou modifikaci. V případě, že při realizaci je použito různých modifikací, je zapotřebí počet připojitelných vstupů k jednomu výstupu odvodit od proudových parametrů hradel.

Dynamické parametry obvodů, viz. obr. 3-4, jsou charakterizovány dobou zpoždění hradla, a to při přechodu z úrovně *L* na *H* dobou t_{pLH} a při přechodu z úrovně *H* na *L* dobou t_{pHL}. Charakteristické hodnoty jsou $t_{pHL} < 15$ ns a $t_{pLH} < 22$ ns. Prakticky se udává střední hodnota t_p, pro kterou platí:

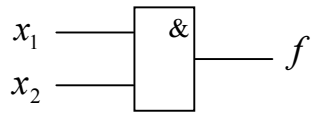
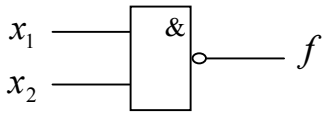
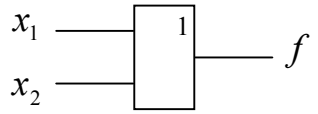
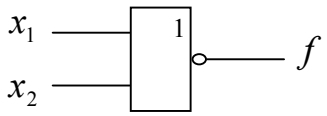
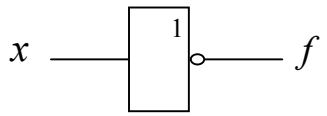
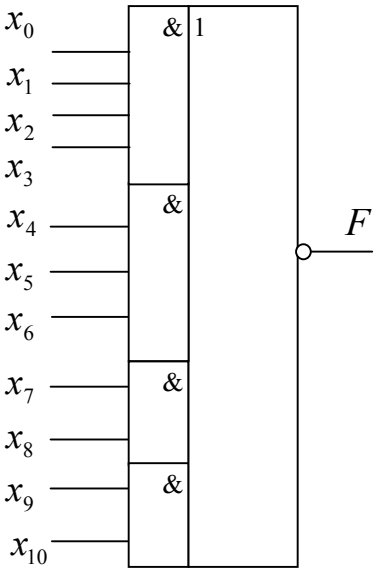
$$t_p = \frac{t_{pHL} + t_{pLH}}{2} \qquad (3\text{-}1)$$

TYP OBVODU	I_{OH} [mA]	I_{OH} [μA]	I_{IL} [mA]	t_P [ns]	N	P [mW]	f_{max} [MHz]
TTL 74xx	16	400	40	12	10	10	25
LTTL 74Lxx	3	100	10	33	1	1	5
HTTL 74Hxx	20	500	50	6	10	20	50
STTL 74Sxx	20	100	50	3	10	20	125
LSTTL 74LSxx	20	100	10	10	10	15	100

Obr. 3-3: Některé parametry obvodů TTL



Obr. 3-4: Dynamické parametry hradla

Název	Funkce	Značka
AND	$f = x_1 \cdot x_2$	
NAND	$f = \overline{x_1 \cdot x_2}$	
OR	$f = x_1 + x_2$	
NOR	$f = \overline{x_1 + x_2}$	
NOT	$f = \overline{x}$	
AND-OR-INVERT	$f = \overline{x_0 \cdot x_1 \cdot x_2 \cdot x_3 + x_4 \cdot x_5 \cdot x_6 + x_7 \cdot x_8 + x_9 \cdot x_{10}}$	

Obr. 3-5: Schématické značky logických členů

3.3 Realizace logických kombinačních obvodů pomocí vícevstupových hradel NAND

Základem pro realizaci kombinačního obvodu je booleovský výraz. Postup při realizaci pomocí vícevstupových hradel NAND je následující:

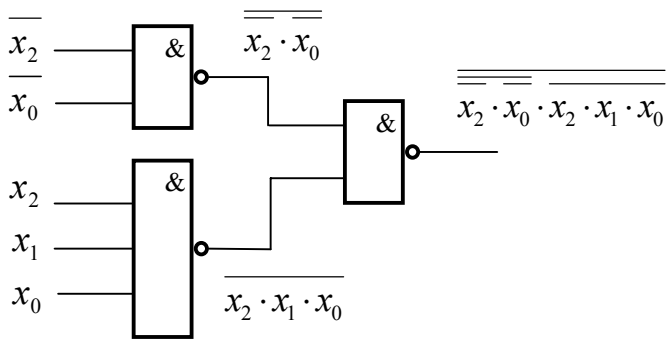
- vycházíme z minimálního součtového tvaru,
- výraz 2x negujeme a pomocí De Morganových zákonů upravíme tak, aby obsahoval jen operaci logický součin.

ŘEŠENÝ PŘÍKLAD

Pomocí vícevstupových hradel NAND realizujte funkci $F_{10} = \overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_1 \cdot x_0$.

Řešení příkladu

$$F_{10} = \overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_1 \cdot x_0 = \overline{\overline{\overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_1 \cdot x_0}} = \overline{\overline{\overline{x_2} \cdot \overline{x_0}} \cdot \overline{x_2 \cdot x_1 \cdot x_0}}$$



*

3.4 Realizace pomocí dvouvstupových hradel NAND

Postup při realizaci pomocí dvouvstupových hradel NAND:

- vycházíme z minimálního součtového tvaru,
- výraz 2x negujeme a pomocí De Morganových zákonů upravíme tak, aby obsahoval jen operaci logický součin,
- dílčí součiny obsahující více než dva členy dále upravíme dvojitou negací.

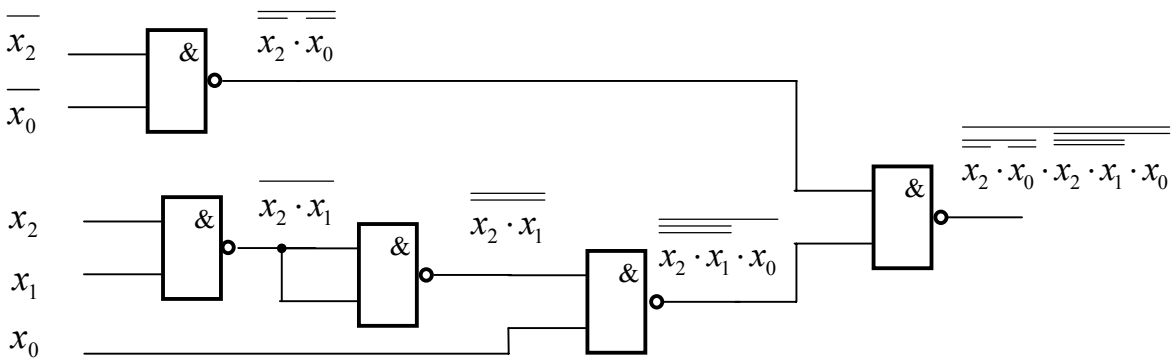
ŘEŠENÝ PŘÍKLAD

Pomocí dvouvstupových hradel NAND realizujte funkci $F_{10} = \overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_1 \cdot x_0$.

Řešení příkladu

$$F_{10} = \overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_1 \cdot x_0 = \overline{\overline{\overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_1 \cdot x_0}} = \overline{\overline{\overline{x_2} \cdot \overline{x_0}} \cdot \overline{x_2 \cdot x_1 \cdot x_0}} =$$

$$= \overline{\overline{\overline{x_2} \cdot \overline{x_0}} \cdot \overline{\overline{x_2 \cdot x_1 \cdot x_0}}}$$



*

3.5 Realizace pomocí dvouvstupových hradel NOR

Postup při realizaci pomocí dvouvstupových hradel NOR:

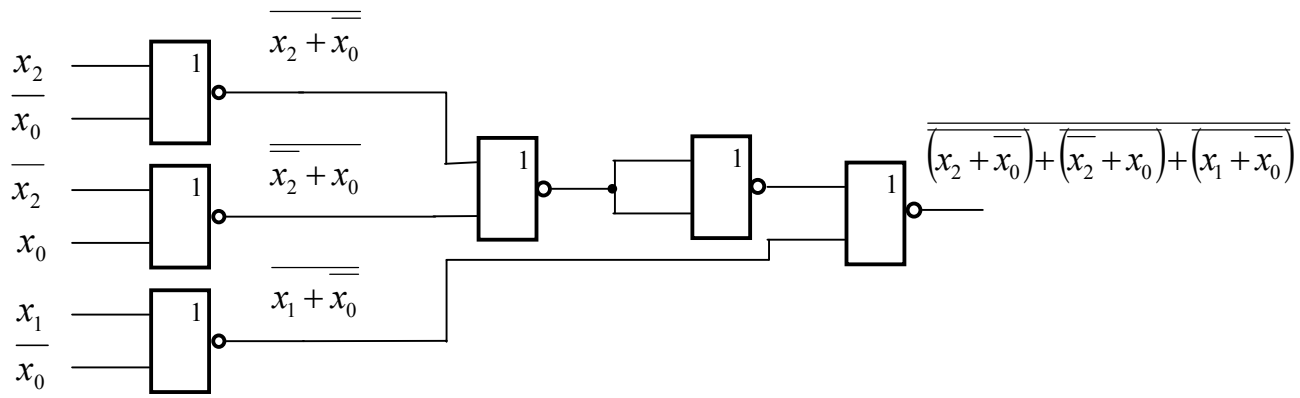
- vycházíme z minimálního součinového tvaru,
- výraz 2x negujeme a pomocí De Morganových zákonů upravíme tak, aby obsahoval jen operaci logický součet,
- dílčí součty obsahující více než dva členy dále upravíme dvojitou negací.

ŘEŠENÝ PŘÍKLAD

Pomocí dvouvstupových hradel NOR realizujte funkci $F_{10} = (x_2 + \overline{x_0}) \cdot (\overline{x_2} + x_0) \cdot (x_1 + \overline{x_0})$.

Řešení příkladu

$$F_{10} = \left(x_2 + \overline{x_0}\right) \cdot \left(\overline{x_2} + x_0\right) \cdot \left(x_1 + \overline{x_0}\right) = \overline{\overline{\left(x_2 + \overline{x_0}\right) \cdot \left(\overline{x_2} + x_0\right) \cdot \left(x_1 + \overline{x_0}\right)}} =$$
$$= \overline{\overline{\left(x_2 + \overline{x_0}\right)} + \overline{\left(\overline{x_2} + x_0\right)} + \overline{\left(x_1 + \overline{x_0}\right)}}$$



*

3.6 Realizace pomocí hradla AND-OR-INVERT

Postup při realizaci pomocí dvouvstupových hradel NOR: …

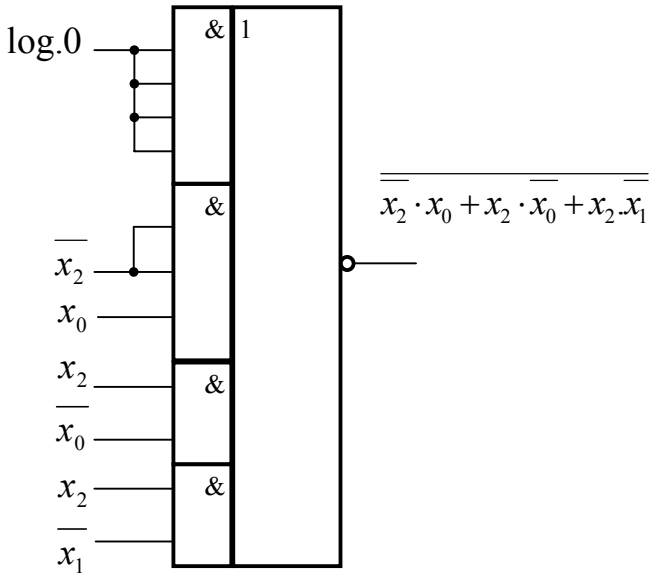
- vycházíme z minimálního součinového tvaru,
- výraz upravíme pomocí De Morganových zákonů tak, aby představoval negaci součtu dílčích součinů.

ŘEŠENÝ PŘÍKLAD

Pomocí hradla AND-OR-INVERT realizujte funkci $F_{10} = \overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_1 \cdot x_0$.

Řešení příkladu

$$F_{10} = \left(x_2 + \overline{x_0}\right) \cdot \left(\overline{x_2} + x_0\right) \cdot \left(\overline{x_2} + x_1\right) = \overline{\overline{x_2}.x_0} \cdot \overline{x_2 \cdot \overline{x_0}} \cdot \overline{x_2 \cdot \overline{x_1}} = \overline{\overline{x_2} \cdot x_0 + x_2 \cdot \overline{x_0} + x_2.\overline{x_1}}$$



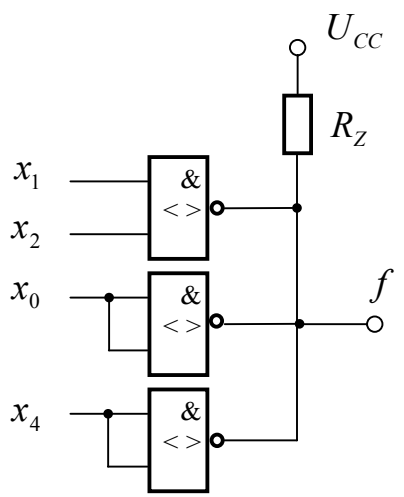
*

3.7 Realizace pomocí hradel NAND s otevřeným kolektorem

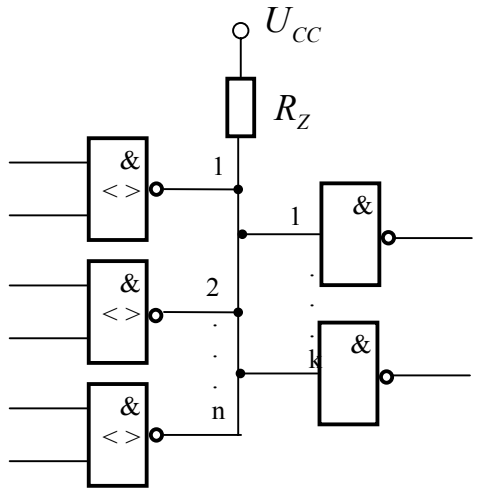
Hradlo s otevřeným kolektorem představuje typ hradla, u kterého není … implementován kolektorový rezistor výstupního tranzistoru. Takovéto hradlo potom umožňuje paralelní spojení výstupů do bodu, do něhož musí být připojen i společný kolektorový rezistor pro všechna paralelně spojená hradla. Realizaci výrazu *(3-2)* odpovídá kombinační síť na obr. 3-6.

$$F = \overline{x_1 \cdot x_2 + x_0 + x_4} \qquad (3\text{-}2)$$

Hodnota rezistoru R_Z závisí na počtu paralelně spojených výstupů hradel a počtu připojených vstupů hradel dalšího stupně. Pro zapojení hradel a volbu hodnoty R_Z platí vztahy z obr. 3-7.



Obr. 3-6: Realizace pomocí hradel s otevřeným kolektorem



$$R_{Z\min} = \frac{U_{CC\max} - U_{OL}}{I_{OL} - k \cdot I_{IL}}$$

$$R_{Z\max} = \frac{U_{CC\min} - U_{OH}}{n \cdot I_{OH} + k \cdot I_{IH}}$$

Obr. 3-7: Zapojení pro výpočet kolektorového odporu

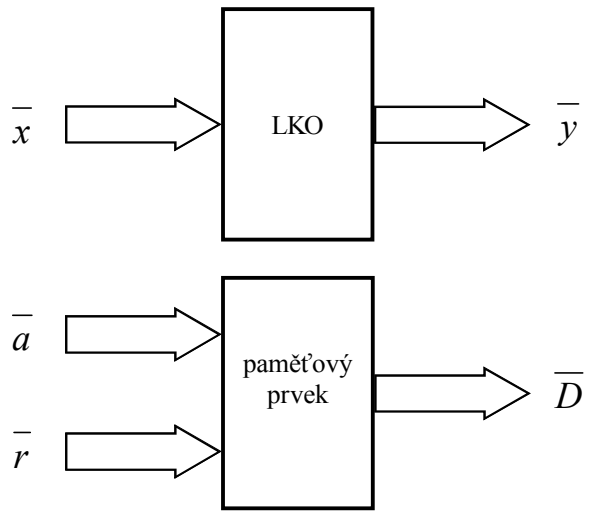
3.8 Realizace kombinačních obvodů pomocí paměťových prvků

Jestliže vycházíme z náhradního blokového schéma kombinačního obvodu, které obsahuje vektor vstupních proměnných $\overline{x}$ a vektor výstupních proměnných $\overline{y}$, a porovnáváme ho s blokovým schématem paměti (ať již typ RAM nebo ROM) na obr. 3-8, můžeme vektor adres $\overline{a}$ přiřadit vektoru $\overline{x}$ a vektor dat $\overline{D}$ vektoru $\overline{y}$. Vstupy charakterizující režim paměti zatím ponecháme bez povšimnutí.

Z organizace paměti potom vyplývá počet adresových vstupů a velikost datového slova. Každé kombinaci logických hodnot na adresových signálech odpovídá v paměti jedna paměťová buňka o délce slova $\overline{D}$, kterou lze libovolně naprogramovat. Například paměť firmy Texas Instrument 27S19 má organizaci 5 x 8, což znamená, že obsahuje 32 adresových kombinací (adresové signály a_4, a_3, a_2, a_1, a_0) a délka slova je 8 bitů ($D_7, D_6, \ldots D_0$). Touto pamětí jsme tedy schopni realizovat 8 různých funkcí, každou pro 5 vstupních proměnných.

Samotné naprogramování pak znamená přenesení pravdivostní tabulky do paměťových buněk. V případě, že počet vstupních proměnných je menší než počet adresových signálů, je nutné nevyužité adresové signály ošetřit. Způsobu přiřazení adresových signálů vektoru $\overline{x}$ odpovídá v úplné pravdivostní tabulce vždy určitá, konkrétní buňka.

Vektorem vstupních proměnných $\overline{r}$ se nastavuje režim, např. čtení, zápis, programování, atd., a to přivedením odpovídající trvalé logické hodnoty pro každou proměnnou v závislosti na typu paměti.



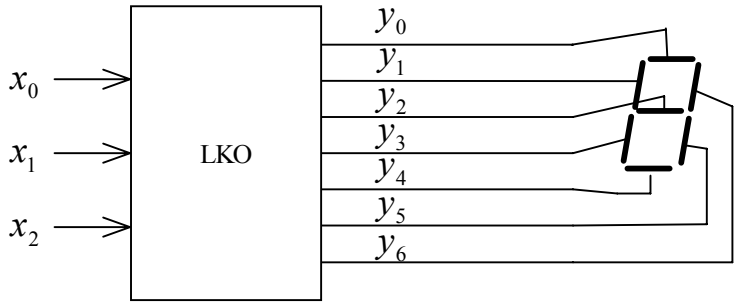
Obr. 3-8: Porovnání kombinačního obvodu a paměťového prvku

ŘEŠENÝ PŘÍKLAD

Pomocí paměti 27S19 realizujte převodník z binárního kódu na kód sedmisegmentu.

Řešení příkladu

Pro zadání všech oktálních číslic potřebujeme 3 vstupní signály x_2, x_1, x_0 a pro zobrazení na sedmisegmentu 7 výstupních signálů $y_6, y_5, \ldots y_0$. Realizujeme tedy kombinační obvod dle následujícího schématu:



Pro paměť zvolíme přiřazení:

$a_0 = x_0$, $a_1 = x_1$, $a_2 = x_2$, $a_3 = 0$, $a_4 = 0$

$D_0 = y_0$, $D_1 = y_1$, $D_2 = y_2$, $D_3 = y_3$, $D_4 = y_4$, $D_5 = y_5$, $D_6 = y_6$

Je zřejmé, že každá paměťová buňka bude obsahovat přímo logické hodnoty výstupních funkcí. Potom kombinační tabulka bude:

a_4	a_3	a_2	a_1	a_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	X	1	1	1	1	0	1	1
0	0	0	0	1	X	1	1	0	0	0	0	0
0	0	0	1	0	X	1	0	1	1	1	0	1
0	0	0	1	1	X	1	1	1	0	1	0	1
0	0	1	0	0	X	0	1	0	0	1	1	0
0	0	1	0	1	X	0	1	1	0	1	1	1
0	0	1	1	0	X	0	1	1	1	1	1	0
0	0	1	1	1	X	1	1	0	0	0	0	1

*

ÚLOHA K ŘEŠENÍ 3-1

Vícevstupovými hradly NAND realizujte funkci $F = \overline{x_3} \cdot \overline{x_2} + \overline{x_3} \cdot \overline{x_1} + x_2 \cdot x_1 \cdot \overline{x_0} + x_3 \cdot x_2 \cdot x_0$

ÚLOHA K ŘEŠENÍ 3-2

Dvouvstupovými hradly NAND realizujte funkci $F(x_2, x_1, x_0) = 10000101$

ÚLOHA K ŘEŠENÍ 3-3

Dvouvstupovými hradly NOR realizujte funkci $F(x_2, x_1, x_0) = (235)_8$

ÚLOHA K ŘEŠENÍ 3-4

Hradlem AND-OR-INVERT realizujte funkci $F(x_2, x_1, x_0) = (235)_8$

KONTROLNÍ OTÁZKY 1

a) Jaký zápis funkce je vhodné použít při návrhu logického kombinačního obvodu realizovaného pomocí hradel NAND?

b) Jaký zápis funkce je vhodné použít při návrhu logického kombinačního obvodu realizovaného pomocí hradel NOR?

ŘEŠENÝ PŘÍKLAD – NÁPOJOVÝ AUTOMAT

V nápojovém automatu jsou umístěny tři nádoby obsahující vodu, malinový sirup a citrónový sirup. Tlačítka na přístroji ovládají automat tak, aby si zákazník vybral čistou vodu, malinovou limonádu nebo citrónovou limonádu. Vodu je možné získat zadarmo, limonádu vydá automat pouze po vhození mince.

Stisknutím kteréhokoliv z tlačítek a vhozením mince se zahájí časově omezený rozhodovací proces. Jestliže tento proces je ukončen dříve, než zákazník učinil platnou volbu, vhozená mince vypadne zpět Mince se vrátí též v případě nesprávné obsluhy.

a) Napište logické výrazy pro řízení automatu a funkci návratu mince v závislosti na stisknutých tlačítkách. Neberte v úvahu zpoždění rozhodovacího procesu.

b) Minimalizujte logické funkce.

c) Nakreslete schéma obvodu, který realizuje tyto funkce.

Řešení příkladu

a) **Použité proměnné a funkce**

Volba proměnných a funkcí

Pro řešení příkladu si zavedeme 4 vstupní proměnné v, m, c, p, které nám budou označovat stavy tlačítek (výběr volby z automatu).

Funkce řídící elektromagnety V, M a C a funkce řídící návrat mince P nám budou popisovat činnost automatu, tedy budou to výstupní funkce.

Výsledné funkce V, M, C, P závisí na proměnných v, m, c, p :

$V = f_1(v,m,c,p)$

$M = f_2(v,m,c,p)$

$C = f_3(v,m,c,p)$

$P = f_4(v,m,c,p)$

Funkce V, M, C a P jsou dvouhodnotové, tzn., že magnet přitáhne nebo nepřitáhne, mince je nebo není vrácena. Proměnné v, m, c, p jsou také dvouhodnotové - tlačítka jsou nebo nejsou stlačena, mince je nebo není vhozena.

Zvolíme tyto konvence:

Zvolená konvence

V=M=C=0: ventily jsou zavřené; P=0: mince je přijata
V=M=C=1: ventily jsou otevřeny; P=1: mince je vrácena
v =m =c =0: tlačítko není stlačeno; p=0: mince není vhozena
v =m =c =1: tlačítko stlačeno; p=1: mince je vhozena

Pravdivostní tabulka popisuje stav každé z výstupních funkcí V, M, C a P pro 16 možných kombinací čtyř vstupních proměnných v,m,c,p. Pro každý ze 16 případů určíme, které elektromagnety mají být vybuzeny (ventily otevřeny) a zda má být mince vrácena:

Pravdivostní tabulka

n	v	m	c	p	V	M	C	P
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	0	0
3	0	0	1	1	1	0	1	0
4	0	1	0	0	0	0	0	0
5	0	1	0	1	1	1	0	0
6	0	1	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	1	0	0	1
10	1	0	1	0	0	0	0	0
11	1	0	1	1	1	0	1	0
12	1	1	0	0	0	0	0	0
13	1	1	0	1	1	1	0	0
14	1	1	1	0	0	0	0	0
15	1	1	1	1	0	0	0	1

Pravdivostní tabulka funkcí V,M,C a P

Jednotlivé kombinace z pravdivostní tabulky

Kombinace $n = 0$

nebyl zadán žádný příkaz, tj. $v = m = c = p = 0$, není tedy třeba provádět žádnou činnost, $V = M = C = P = 0$.

Kombinace $n = 1$

je vhozena mince, ale nenásledovala žádná volba nápoje. Po skončení předepsaného časového intervalu, který začíná vhozením mince, stroj minci vrátí, ale nevybudí žádný elektromagnet;
$V = M = C = 0$, $P = 1$.

Kombinace $n = 2$

Zákazník stiskl tlačítko citrónové limonády, ale nevhodil v předepsané době žádnou minci. Neprovádí se žádná činnost;
$V = M = C = P = 0$.

Kombinace $n = 3$

Je vhozena mince a zákazník stlačil tlačítko citrónové limonády. Vzniknou tedy dva povely:
ventil citrónového sirupu $C = 1$ a ventil vody $V = 1$, $C = P = 0$.

Kombinace $n = 4$

Je stisknuto pouze tlačítko malinové limonády, ale není vhozena mince;
$V = M = C = P = 0$.

Kombinace $n = 5$

Kombinace podobného typu jako kombinace 3;
$V = M = 1$, $C = P = 0$.

Kombinace $n = 6$

Zákazník stiskl tlačítka malinové limonády a citrónové limonády, ale nevhodil minci. Servírování koktejlů zadarmo není v plánu;
$V = M = C = P = 0$.

Kombinace $n = 7$

Zákazník stiskl tlačítka malinové a citrónové limonády a vhodil minci. Bohužel však nelze nabídnout koktejly, i když jsou placené;
$V = M = C = 0$, $P = 1$.

Kombinace $n = 8$

Zákazník stiskne pouze tlačítko voda. Protože voda je zadarmo, stačí tento jediný signál k otevření ventilu vody; $V = 1$, $M = C = P = 0$.

Kombinace $n = 9$

Zákazník, který byl příliš čestný nebo špatně informovaný, stiskne tlačítko vody a vhodí minci. Nalijeme mu sklenici vody ($V = 1$) a vrátíme minci ($P = 1$); $M = C = 0$.

Kombinace $n = 10$

Zákazník stiskl tlačítko vody a tlačítko citrónové limonády bez vhození mince. Žádná činnost se neprovádí. $V = M = C = P = 0$.

Kombinace $n = 11$

Zákazník stiskl tlačítko vody a tlačítko citrónové limonády a vhodil minci. Je mu nabídnuta citrónová limonáda. $V = C = 1$, $M = P = 0$.

Kombinace $n = 12$

Zákazník chce malinovou limonádu zadarmo. Žádná činnost se neprovádí. $V = M = C = P = 0$.

Kombinace $n = 13$

Malinová limonáda řádně zaplacena. $V = M = 1$, $C = P = 0$.

Kombinace $n = 14$

Nerozhodný zákazník, který navíc nezaplatil. Žádná činnost se neprovádí. $V = M = C = P = 0$.

Kombinace $n = 15$

Platící zákazník, který zmáčkne všechno co vidí, takže nelze splnit jeho požadavky. Vrátíme mu minci v naději, že se naučí jak má s přístrojem správně zacházet. $V = M = C = 0$, $P = 1$.

Poznámky k sestavení pravdivostní tabulky:

Mohli bychom také vyžadovat stisknutí tlačítka pro vodu. To se ovšem zdá být zbytečné od okamžiku, kdy volba je jasně provedena a nápoje zaplaceny. Navíc se takovýmto zjednodušením vyhýbáme zbytečné činnosti zákazníka.

V nejasných případech bychom také mohli nabídnout sklenici vody, ale to by jistě nebyla činnost, která by byla výhodná z hlediska majitele automatu, protože ten má zájem na tom, aby uspokojil přání, která jsou vyjádřena jasně a úplně. Analýza problému je obzvláště jednoduchá, jestliže pečlivě prostudujeme všechny možné případy a popíšeme každou situaci pomocí operátorů Booleovy algebry.

Zápis logických funkcí

Zápis logických funkcí

Tyto funkce můžeme odvodit přímo z pravdivostní tabulky.

$$V = \bar{v}\,\bar{m}\,c\,p + \bar{v}\,m\,\bar{c}\,p + v\,\bar{m}\,\bar{c}\,\bar{p} + v\,\bar{m}\,\bar{c}\,p + v\,\bar{m}\,c\,p + v\,m\,\bar{c}\,p$$

$$M = \bar{v}\,m\,\bar{c}\,p + v\,m\,\bar{c}\,p$$

$$C = \bar{v}\,\bar{m}\,c\,p + v\,\bar{m}\,c\,p$$

$$P = \bar{v}\,\bar{m}\,\bar{c}\,p + \bar{v}\,m\,c\,p + v\,\bar{m}\,\bar{c}\,p + v\,m\,c\,p$$

b) **Zjednodušení logických funkcí**

Každou ze získaných funkcí si můžeme dešifrovat a ověřit, do jaké míry odpovídá skutečnosti. Pro minimalizaci výše uvedených logických funkcí použijeme Karnaughovy mapy.

Zjednodušení logické funkce M

M

$c \backslash p$ \ m \ v			
	0_0	0_4	0_{12}	0_8
	0_1	1_5	1_{13}	0_9
	0_3	0_7	0_{15}	0_{11}
	0_2	0_6	0_{14}	0_{12}

Z mapy: $M = m\,\bar{c}\,p$

Nádoba s malinovým sirupem se otevře, jestliže zákazník stiskl tlačítko malinová limonáda (m), nestiskl tlačítko citrónová limonáda (c) a zároveň vhodil minci (p). Tento výraz tedy pokrývá kombinace 5 a 13.

Zjednodušení logické funkce C

C

$c \backslash p$ \ $m\,v$	$\overline{m}\,\overline{v}$	$\overline{m}\,v$	$m\,v$	$m\,\overline{v}$
$\overline{c}\,\overline{p}$	0	0	0	0
$\overline{c}\,p$	0	0	0	0
$c\,\overline{p}$	1_3	0	0	1_{11}
$c\,p$	0	0	0	0

Z mapy: $C = \overline{m}\,c\,p$:

Nádoba s citrónovým sirupem se otevře, jestliže zákazník stiskl tlačítko citrónová limonáda (c), nestiskl tlačítko malinová limonáda (m) a vhodil minci (p). Tento výraz tedy pokrývá kombinace 3 a 11.

Zjednodušení logické funkce V

V

$c \backslash p$ \ $m\,v$	$\overline{m}\,\overline{v}$	$\overline{m}\,v$	$m\,v$	$m\,\overline{v}$
$\overline{c}\,\overline{p}$	0	0	0	1_8
$\overline{c}\,p$	0	1_5	1_{13}	1_9
$c\,\overline{p}$	1_3	0	0	1_{11}
$c\,p$	0	0	0	0

Jestliže si všimneme, že funkce V závisí na funkcích M a C, můžeme z mapy odvodit: $V = v\overline{m}\,\overline{c} + M + C$

Nádoba s vodou se otevře:

- jestliže zákazník stiskne tlačítko voda (v) a nestiskne ani tlačítko malinová limonáda (m), ani tlačítko citrónová limonáda (c),
- jestliže se otevřela nádoba s malinovým sirupem (M),
- jestliže se otevřela nádoba s citrónovým sirupem (C).

Tento výraz pokrývá výroky odpovídající kombinacím 3,5,8,9,11 a 13. Je zřejmě nemožné takovýto výraz odvodit přímo.

Zjednodušení logické funkce P

P

$c \backslash p$ \ m	$\overline{v}$			v
	0	0	0	0
	1_1	0	0	1_9
	0	1_7	1_{15}	0
	0	0	0	0

Z mapy: $P = (\overline{m}\,\overline{c} + m\,c)\,p$

Automat vrátí minci, jestliže tato mince byla vhozena a současně byla ovládací tlačítka použita nesprávně (byla vyžádána současně malinová a citrónová limonáda nebo nebyla vybrána ani malinová, ani citrónová limonáda). Tento výraz pokrývá výroky odpovídající kombinacím 1,7,9 a 15. Jeho sestavení přímo bez předchozí analýzy a syntézy by bylo velmi obtížné.

Minimalizované funkce

Výsledné tvary minimalizovaných funkcí:

$M = m\,\overline{c}\,p$

$C = \overline{m}\,c\,p$

$V = v\,\overline{m}\,\overline{c} + M + C$

$P = (\overline{m}\,\overline{c} + m\,c)\,p$

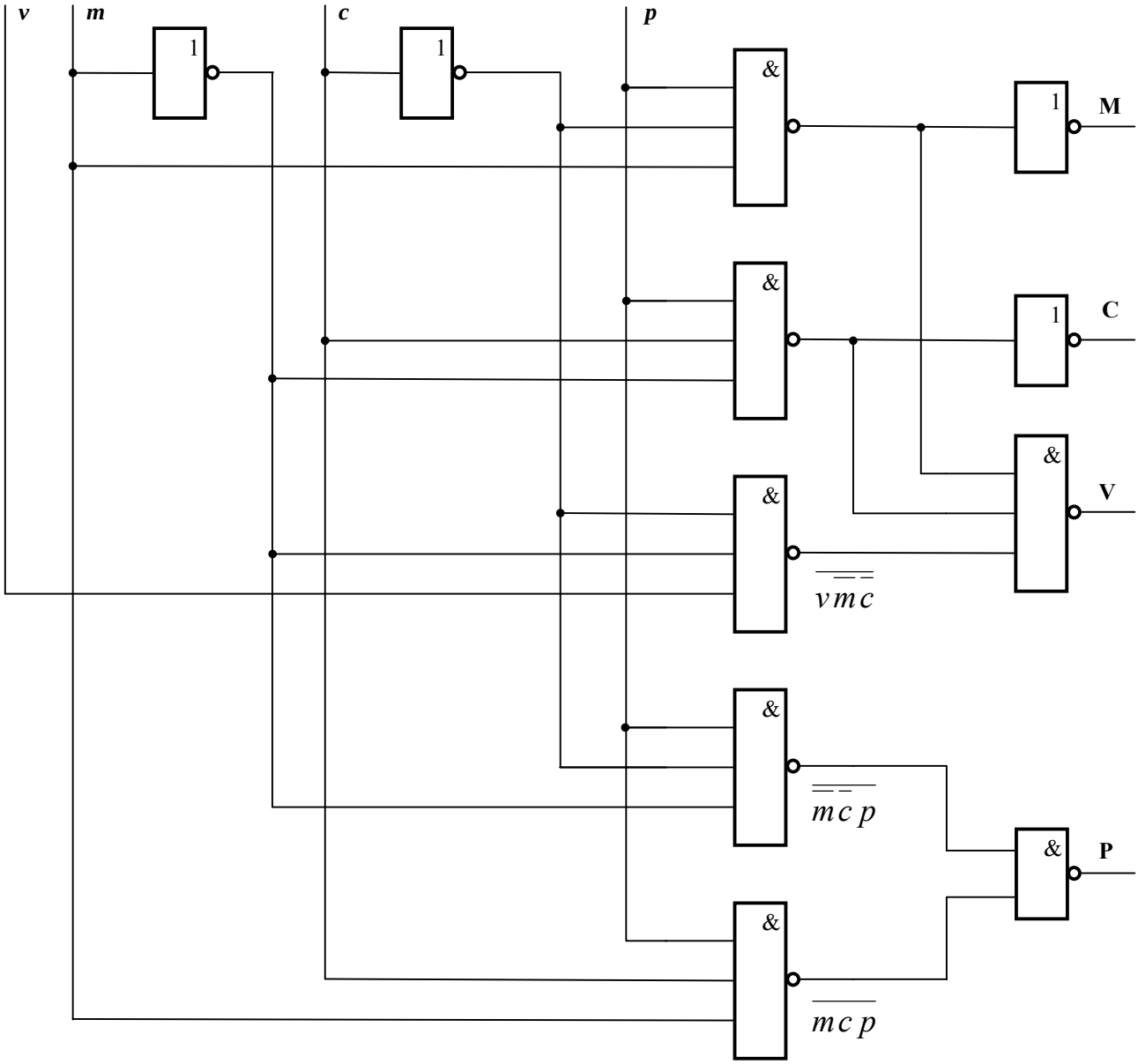
Hodnocení navrhované metody

Hodnocení navrhované metody

Navrhovaná metoda má hlavní výhodu v tom, že je systematická, takže nenechává bez povšimnutí žádný případ, žádnou kombinaci. Použitím Karnaughových map umožňuje zjednodušení funkcí. Na tomto příkladu vidíme, že v některých případech je výhodné odvodit závislost některých funkcí na jiných funkcích.

c) **Odpovídající schéma zapojení** Schéma zapojení

Víme, že pro stejnou funkci lze odvodit různé výrazy tím, že vrcholy pokryjeme různými implikanty. Ze stejného výrazu můžeme navíc odvodit různá schémata, podle toho, jaké použijeme součástky. Pro názornost je použito schéma se základními logickými členy:



*

3.9 Hazardní stavy

Rozdělení hazardních stavů

Logické obvody, které realizují určité logické funkce, mohou vlivem zpoždění signálu t_p v jednotlivých logických členech po přechodnou dobu vykazovat na výstupech jiné hodnoty, než odpovídá modelovaným funkcím. Takovouto situaci označujeme jako hazard. V logických kombinačních obvodech rozlišujeme tři typy hazardů: statický, dynamický a souběhový.

Statický hazard

Statický hazard vykazuje ten obvod, u kterého při přechodu mezi dvěma sousedními stavy vstupních proměnných (stavy, které se liší v hodnotě jedné proměnné) se stejnou logickou hodnotou výstupní proměnné (log 0 nebo log I) dochází na přechodnou dobu ke změně předepsané výstupní hodnoty.

Předpokládejme, že dvě vstupní n-tice se liší ve vstupní proměnné x_i , potom funkční hodnotu vyjádřenou součtovou nebo součinovou formou můžeme psát jako:

$$y = \overline{x_i} \cdot f(a) + x_i \cdot f(b) \qquad (3\text{-}3)$$

$$y = (\overline{x_i} + f(a)) \cdot (x_i + f(b)) \qquad (3\text{-}4)$$

kde $f(a)$ a $f(b)$ představuje logickou hodnotu výstupní proměnné závislé pouze na zbývajících vstupních proměnných.

Statický hazard (součtová forma)

Jestliže v rovnici *(3-3)* pro součtovou formu je $f(a) = f(b) = 1$, potom pro dva vstupní stavy, lišící se pouze v proměnné x_i musí platit:

$$y = \overline{x_i} + x_i = 1 \qquad (3\text{-}5)$$

Rovnice *(3-5)* je jedním ze základních postulátů booleovské algebry. Protože však booleovská algebra nezohledňuje praktickou realizaci logické sítě (zpoždění jednotlivých hradel) může v krátkém časovém okamžiku rovnice *(3-5)* ztratit platnost a tudíž platí:

$$y = \overline{x_i} + x_i = 0 \qquad (3\text{-}6)$$

Statický hazard (součinová forma)

Obdobně pro součinovou formu, za předpokladu $f(a) = f(b) = 0$, vznikne statický hazard jestliže platí:

$$y = \overline{x_i} \cdot x_i = 1 \qquad (3\text{-}7)$$

Příklad na statický hazard

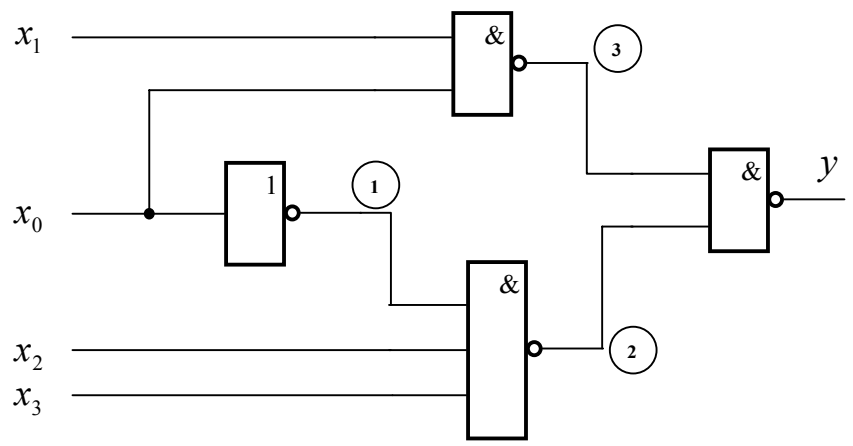
Mějme obvod, který realizuje logickou funkci $y = x_1 \cdot x_0 + x_2 \cdot \overline{x_0} \cdot x_3$ pomocí logických členů NAND, z nichž každý má zpoždění t_p, viz. obr.3-9.

Použijeme-li již odvozené vztahy, potom ke statickému hazardu v realizovaném obvodu musí docházet při změně vstupní proměnné x_0 při vstupních proměnných $x_1 = x_2 = x_3 = 1$. Z obr. 3-10 je zřejmé, jak ke statickému hazardu v log I vlivem konečného zpoždění v logických členech dochází.

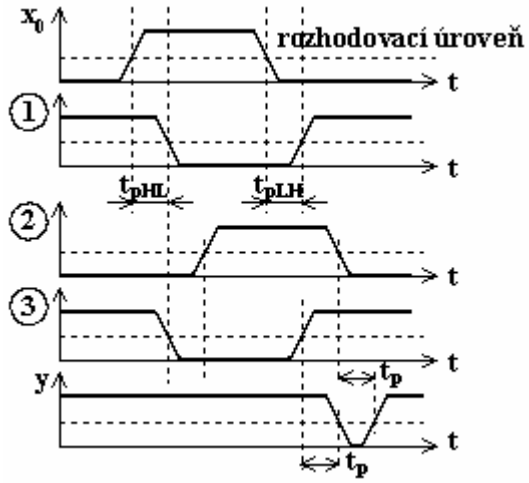
Statický hazard lze stanovit nejen použitím rovnice *(3-6)* nebo *(3-7)*, ale také přímo z Karnaughovy mapy. Jestliže zapíšeme funkci f do mapy, viz. obr. 3-11, vidíme, že statický hazard může vzniknout tehdy, jestliže sousední jednotková (nebo nulová) políčka jsou pokryta různými výrazy. Minterm funkce $x_3 \cdot x_2 \cdot x_1 \cdot x_0$ je pokryt výrazem $x_1 \cdot x_0$ a minterm $x_3 \cdot x_2 \cdot x_1 \cdot \overline{x_0} \cdot$ výrazem $x_3 \cdot x_2 \cdot \overline{x_1} \cdot$. Odtud zjistíme, že ke statickému hazardu bude docházet při změně té proměnné, ve které se obě sousední políčka liší, tj. proměnné x_0 při $x_3 = x_2 = x_1 = 1$. Jeho maximální délku lze odhadnout z časových parametrů použitých obvodů, ale skutečná šířka se bude měnit s teplotou, napájením, zatížením, atd.

Způsob odstranění hazardních stavů je zřejmý z předcházejícího textu. Hazard bude potlačen, jestliže kritická změna proměnné, která je příčinou hazardu, bude pokryta mintermem složek $\overline{x_i} \cdot f_i(0), x_i \cdot f_i(1)$ nebo $(\overline{x_i} + f_i(0)), (x_i + f_i(1))$ nebo výrazy, které tato políčka pokrývají.

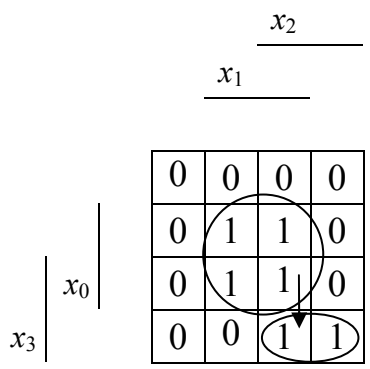
Z toho plyne, že odstraňování hazardů nutně vede k neminimálním logickým výrazům a tudíž i k složitějším realizacím obvodu. Hazardy lze vylučovat přímo při odvozování minimálních forem funkce z mapy tím, že všechny přechody, které mohou způsobit hazard pokryjeme dalšími implikanty funkce, které by normálně nemusely být vybrány.



Obr. 3-9: Logická síť



Obr. 3-10: Časové průběhy

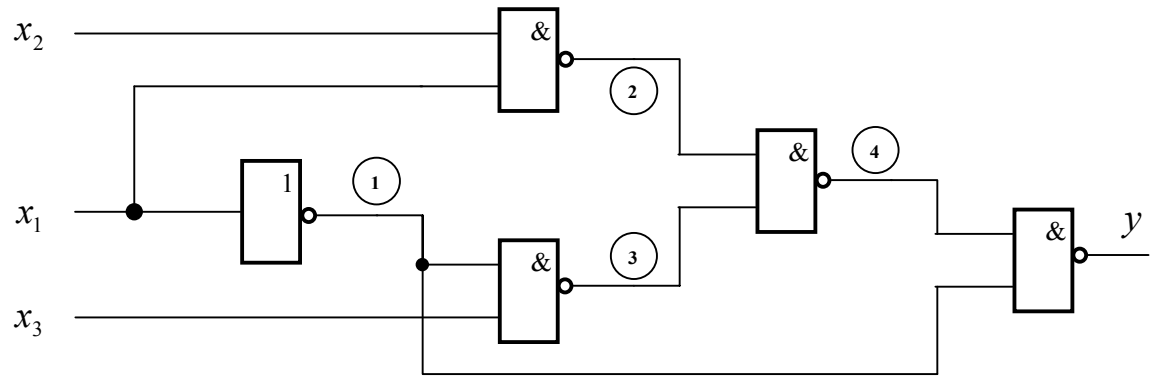


Obr. 3-11: Vyznačení místa statického hazardu

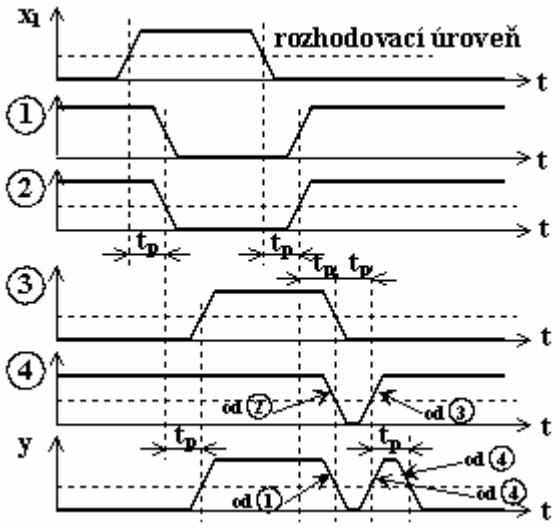
Dynamický hazard je stav obvodu, kdy výstupní proměnná při přechodu z 0→1 nebo 1→0 projde posloupností stavů 0→1→0→1 nebo 1→0→1→0. Jinak řečeno místo skokové změny na výstupu obvodu se objevují zákmity dané strukturou obvodu. V obvodech s logickými členy se může vyskytnout dynamický hazard tehdy, jestliže tutéž proměnnou zavádíme do obvodu přímo a v komplementu v různých stupních obvodu, např. na obr. 3-12.

Dynamický hazard

Dynamický hazard se může vyskytnout pouze ve více než dvoustupňových obvodech a je způsoben statickým hazardem v obvodu. Na obr. 3-13 jsou zobrazeny časové průběhy pro změnu vstupní proměnné x_1 při $x_2 = 1$ a $x_3 = 1$ u obvodu z obr. 3-12.



Obr. 3-12: Zapojení kombinačního obvodu

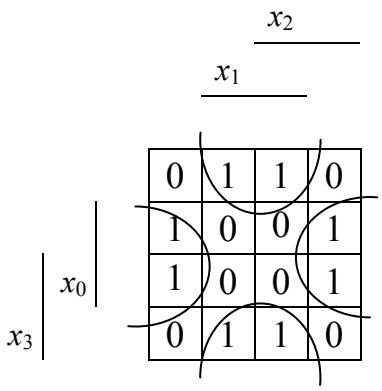


Obr. 3-13: Časové průběhy

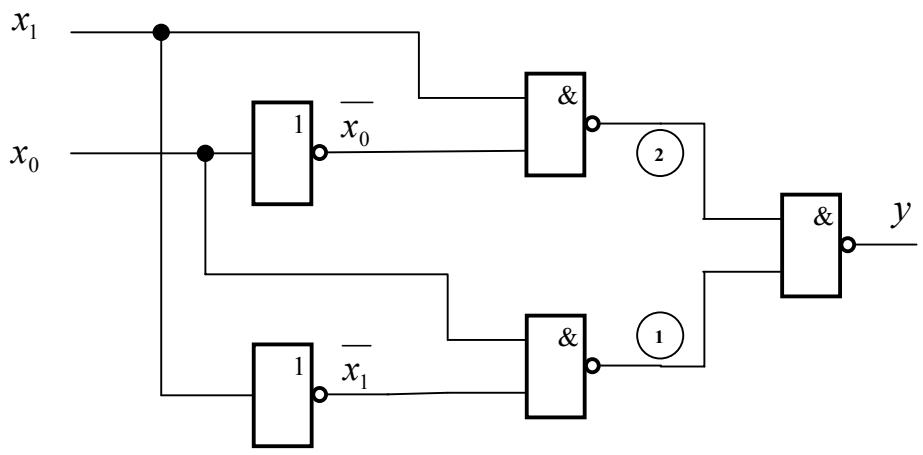
Souběhový hazard je přechodný stav obvodu, který je vyvolán současnou změnou dvou a více vstupních proměnných, při němž výstupní signál na přechodnou dobu nabývá nesprávné hodnoty. Jedná se v podstatě o rozšíření definice statického hazardu. Prostředky potlačení těchto hazardů jsou stejné pro případy, kdy přechod mezi stavy obvodu, při němž vzniká statický hazard, lze pokrýt implikantem modelované funkce.

Souběhový hazard

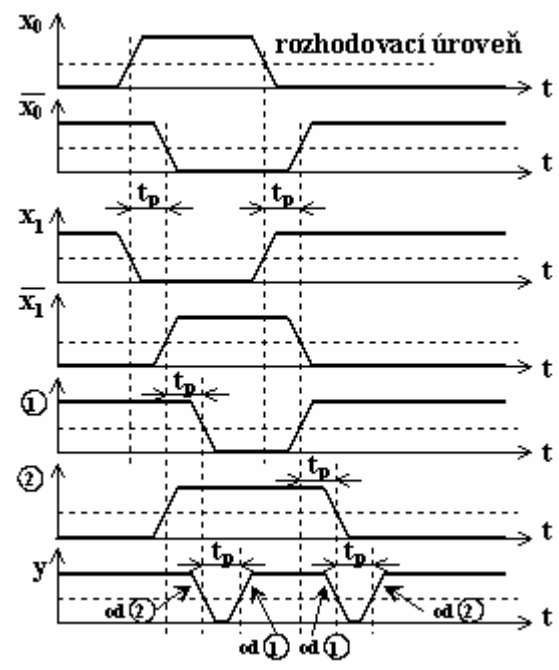
Zvláštním případem souběhového hazardu je hazard, který nelze pokrýt konsensem obr. 3-14. Zapojení obvodu generujícího souběhový hazard je uvedeno na obr. 3-15 a odpovídající časové průběhy na obr. 3-16. V tomto případě je možné hazard potlačit pouze zařazením dodatečných zpožďovacích členů přímo do kombinačního obvodu, obr. 3-17.



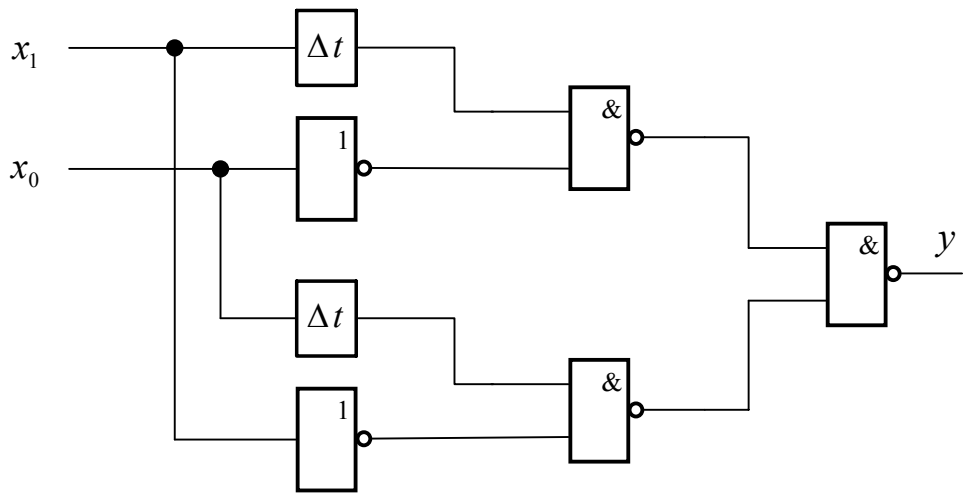
Obr. 3-14: Mapa obvodu se souběhovým hazardem



Obr. 3-15: Zapojení obvodu generujícího souběhový hazard



Obr. 3-16: Časové průběhy v obvodu



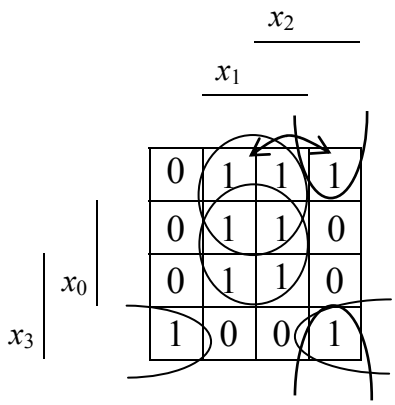
Obr. 3-17: Zapojení obvodu eliminující souběhový hazard

ŘEŠENÝ PŘÍKLAD

Zapište do Karnaughovy mapy funkci F_6 zadanou v minimálním součtovém tvaru $F_6 = x_1 \cdot x_0 + \overline{x_3} \cdot x_1 + x_3 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot \overline{x_0}$. Vyjádřete formu zápisu funkce, která eliminuje vznik statických hazardů v součtovém tvaru.

Řešení příkladu

Karnaughova mapa s vyznačením smyček pro součtový tvar funkce $\boldsymbol{F_6}$ a místa hazardu:



Karnaughova mapa

Zápis funkce, která eliminuje vznik statických hazardů

$$F_6 = x_1 \cdot x_0 + \overline{x_3} \cdot x_1 + x_3 \cdot \overline{x_1} \cdot \overline{x_0} + x_2 \cdot \overline{x_1} \cdot \overline{x_0} + \overline{x_3} \cdot x_2 \cdot \overline{x_0}$$

*

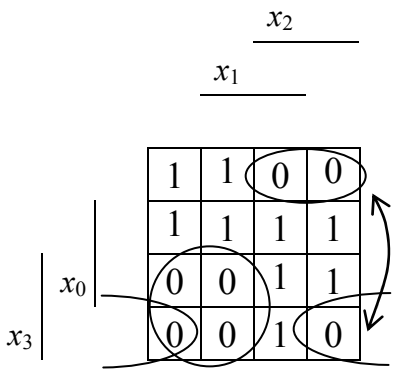
ŘEŠENÝ PŘÍKLAD

Zapište do Karnaughovy mapy funkci F_7 zadanou v minimálním součinovém tvaru $F_7 = (\overline{x_3} + x_2) \cdot (x_3 + \overline{x_2} + x_0) \cdot (\overline{x_3} + x_1 + x_0)$. Vyjádřete formu zápisu funkce, která eliminuje vznik statických hazardů v součinovém tvaru.

Řešení příkladu

Karnaughova mapa s vyznačením smyček pro součinový tvar funkce $\boldsymbol{F_7}$ a místa hazardu:

Karnaughova mapa



Karnaughova mapa

Zápis funkce, která eliminuje vznik statických hazardů

$$F_7 = (\overline{x_3} + x_2) \cdot (x_3 + \overline{x_2} + x_0) \cdot (\overline{x_3} + x_1 + x_0) \cdot (\overline{x_2} + x_1 + x_0)$$

*

ÚLOHA K ŘEŠENÍ 3-5

Vyjádřete v součtovém a součinovém tvaru funkci $F_{11}(x_2,x_1,x_0)$ = =$(235)_8$, minimalizujte ji a vyjádřete i formy, které eliminují vznik statických hazardů.

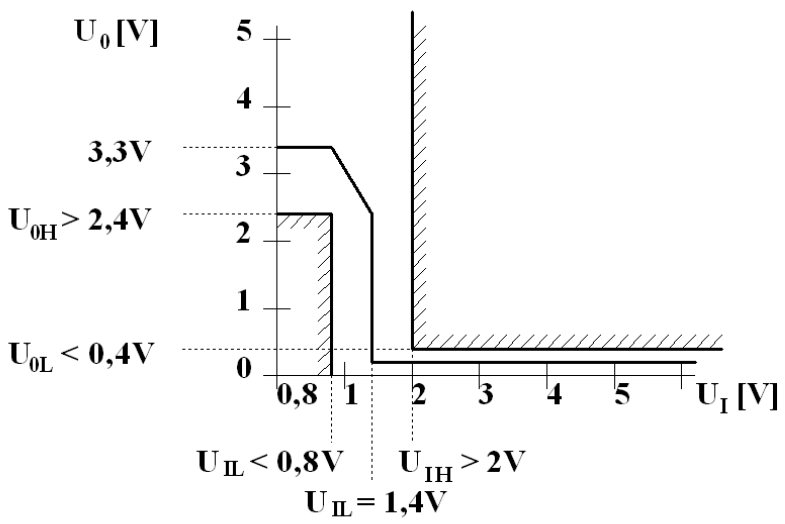
3.10 Ošetření vstupních signálů

Ošetření vstupních signálů

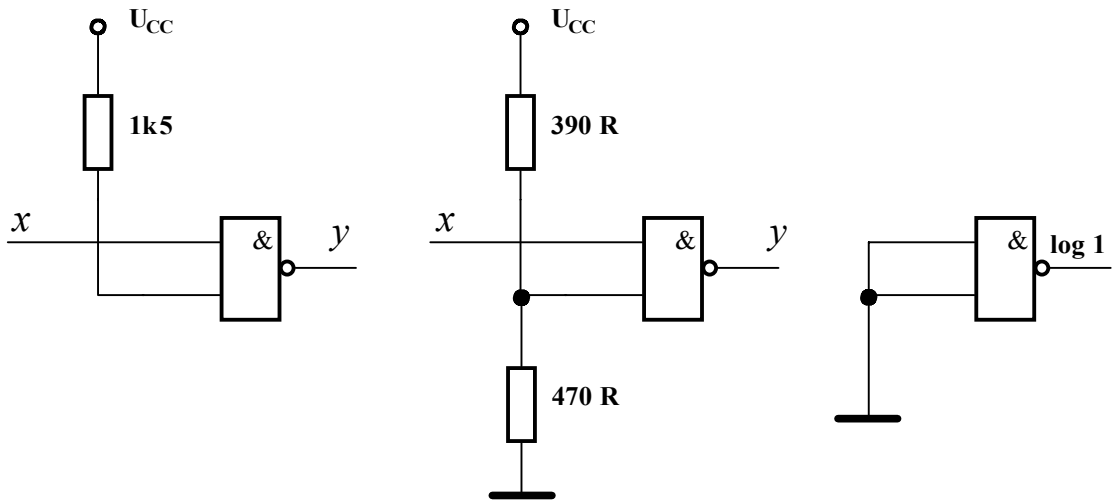
Nevyužití všech vstupů hradla vede k nejednoznačnosti logické hodnoty, viz. přechodová charakteristika hradla na obr. 3-18, a proto dochází k náhodným změnám výstupní proměnné. Z tohoto důvodu je nutné tyto vstupy ošetřit, tj. definovat logickou hodnotu. Způsob ošetření závisí na konkrétním zapojení.

Pro definování logické hodnoty 0 se tento vstup připojí vždy na GND. Pro definování logické hodnoty 1 se použijí varianty podle obr. 3-19. V případě, že budící hradlo signálu *x* má dostatečný zisk, používá se připojení volného vstupu na tento signál.

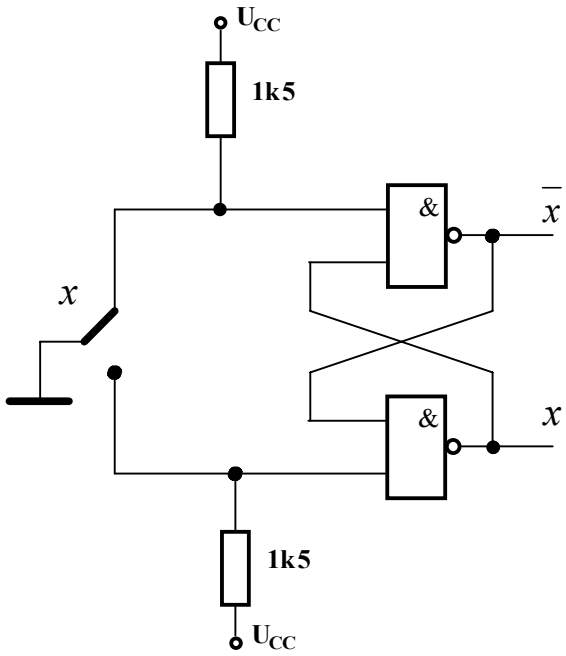
Při praktickém ověřování kombinačních obvodů se velmi často používá ke generování vstupních proměnných manuálně ovládaných tlačítek, které při přepínání zakmitávají, a je proto rovněž nutné jejich ošetření. Nejběžnější způsob je pomocí RS obvodu (kap. 4.1.2 Paměťový člen RS) dle obr. 3-20.



Obr. 3-18: Přechodová charakteristika hradla TTL.



Obr. 3-19: Možnosti ošetření nevyužitých vstupů



Obr. 3-20: Způsob ošetření vstupního signálu

SHRNUTÍ KAPITOLY LOGICKÉ KOMBINAČNÍ OBVODY

Shrnutí

Po prostudování této kapitoly jsme schopni navrhnout i realizovat různými způsoby logický kombinační obvod, tzn. logický obvod, v němž logická hodnota výstupní proměnné závisí na logické hodnotě vstupních proměnných.

PRŮVODCE STUDIEM 4

Dokončete vypracování samostatné práce, tj. bod c). Správnost svého návrhu si ověříte v laboratoři při praktickém zapojení samostatné práce.

4 LOGICKÉ SEKVENČNÍ OBVODY

ČAS POTŘEBNÝ KE STUDIU

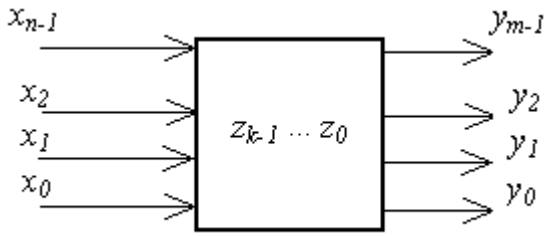
Předpokládaný čas k prostudování kapitoly Logické sekvenční obvody je **50 hodin**.

RYCHLÝ NÁHLED DO PROBLEMATIKY KAPITOLY LOGICKÉ SEKVENČNÍ OBVODY

Rychlý náhled

Princip činnosti sekvenčního obvodu je patrný z obr. 4-1. Do vztahů popisujících chování obvodu vstupují vektory hodnot vnitřních (stavových) proměnných. Logické hodnoty stavových a výstupních proměnných se mění v závislosti na logických hodnotách nebo změnách logických hodnot vstupních proměnných v čase. Logický sekvenční obvod představuje tedy dynamický systém, tj. systém pracující v čase. Vnitřní proměnné se nazývají stavy systému.

Kapitola se zabývá analýzou a návrhem různých typů sekvenčních obvodů. Dále nás seznamuje s činností posuvných registrů, synchronních nebo asynchronních čítačů a s generátory binárních posloupností.



Obr. 4-1: Obecné blokové schéma logického systému

CÍLE KAPITOLY LOGICKÉ SEKVENČNÍ OBVODY

- Budete umět analyzovat zapojení sekvenčních obvodů bez paměťových členů a s paměťovými členy,
- získáte přehled o způsobech realizace paměťových členů RS, RST, D a JK a přehled o jejich chování,
- budete schopni navrhnout a realizovat synchronní sekvenční obvod, posuvný registr, synchronní nebo asynchronní čítač a generátor binárních posloupností.

KLÍČOVÁ SLOVA KAPITOLY LOGICKÉ SEKVENČNÍ OBVODY

Klíčová slova

Analýza obvodu, asynchronní sekvenční obvod, synchronní sekvenční obvod, synchronizační signál, Mealyho typ, Moorův typ, paměťový člen RS, paměťový člen RST, paměťový člen D, paměťový člen JK, vývojový diagram, zpožďovací člen, paměťový registr, posuvný registr, synchronní čítač, asynchronní čítač, dělička číslicového signálu, generátor binární posloupnosti.

PRŮVODCE STUDIEM 5

Druhá samostatná práce je zaměřena na oblast logických sekvenčních obvodů. Problematika logických sekvenčních obvodů je poměrně rozsáhlá a její zvládnutí bude vyžadovat i náročnější studium. Zadání samostatné práce je pro všechny studenty opět společné, ale každý student dostává pro zobrazení zadané jiné stavy automatu. Toto konkrétní zadání si opět vyzvedněte na tutoriálu nebo formou e-mailu. S vypracováním samostatné práce je vhodné začít až po zvládnutí kapitoly 4.2.

SAMOSTATNÁ PRÁCE 2

Navrhněte binární synchronní automat na základě zadaných stavů s možností čtení vpřed a vzad a se zobrazováním stavů v hexadecimálním tvaru. Automat realizujte z paměťových členů D a JK.

Budeme-li vycházet z obecného blokového schématu logického systému na obr. 4-1, pak lze logické sekvenční obvody charakterizovat jako obvody, jejichž výstupní hodnota závisí na historii hodnot vstupního vektoru $\bar{x} = [x_{n-1}, \ldots, x_1, x_0]$. Dále závisí na vnitřních veličinách daného logického systému, které lze charakterizovat vektorem $\bar{z} = [z_{k-1}, \ldots, z_1, z_0]$. Výstupní vektor je obvykle ve tvaru $\bar{y} = [y_{m-1}, \ldots, y_j, \ldots, y_1, y_0]$.

Na základě těchto skutečností lze pro obecnou výstupní proměnnou y_j použít následující popis:

$$y_j^i = f_j\left(\bar{x}^i, \bar{x}^{i-1}, \ldots \bar{x}^0, \bar{z}^0\right) \qquad (4\text{-}1)$$

kde parametr i představuje diskrétní bod odpovídající konkrétnímu času.

Lze-li předpokládat, že historii logických hodnot vstupních proměnných jsme schopni vyjádřit pomocí vnitřních veličin logického systému, lze poté použít následující tvar popisu:

$$y_j^i = f_j\left(\bar{x}^i, \bar{z}^i\right) = f_j\left(x_{n-1}^i, \ldots, x_1^i, x_0^i, z_{k-1}^i, \ldots, z_1^i, z_0^i\right) \qquad (4\text{-}2)$$

Z rovnice *(4-2)* plyne důležitá skutečnost, a to že vlastní obvod musí realizovat i zobrazení obecné libovolné funkce pro vnitřní proměnnou.

Obecnou vnitřní proměnnou z_h lze následně popsat:

$$z_h^{i+1} = g_h\left(\bar{x}^i, \bar{z}^i\right) = g_h\left(x_{n-1}^i, \ldots, x_1^i, x_0^i, z_{k-1}^i, \ldots, z_1^i, z_0^i,\right) \qquad (4\text{-}3)$$

Z rovnice *(4-3)* dále vyplývají následující skutečnosti. Na základě vstupních a vnitřních proměnných v diskrétním bodě i stanovíme logické hodnoty vnitřních proměnných pro následující diskrétní bod i+1. Funkce g_h poté popisuje závislost následujících stavů vnitřních proměnných, nazýváme ji funkcí přechodů. Rovnice *(4-2)* a *(4-3)* představují zobrazení realizované kombinačním obvodem dle obr. 4-2, kde vektor $\bar{y}^i$ představuje vektor obecných výstupních proměnných $\bar{y}^i = [y_{m-1}^i, \ldots, y_1^i, y_0^i,]$.

Správnost předcházející úvahy lze dokázat úpravou rovnice *(4-3)* pro diskrétní čas i, kdy platí:

$$z_h^i = g_h\left(\overline{x}^{i-1}, \overline{z}^{i-1}\right) \qquad (4\text{-}4)$$

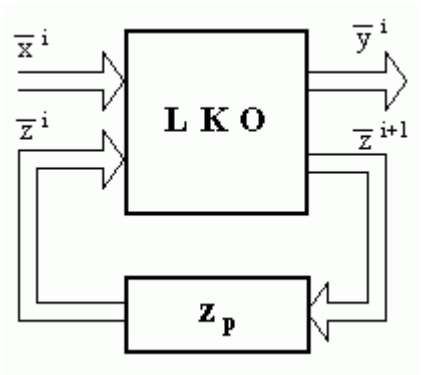
Postupným dosazováním rovnice *(4-4)* do rovnice *(4-2)* dostaneme výraz:

$$y_j^i = f_j\left(\overline{x}^i, g\left(\overline{x}^{i-1}, \overline{z}^{i-1}\right)\right) = f_j\left(\overline{x}^i, g\left(\overline{x}^{i-1}, g\left(\overline{x}^{i-2}, \overline{z}^{i-2}\right)\right)\right) = \qquad (4\text{-}5)$$

$$= f_j\left(\overline{x}^i, g\left(\overline{x}^{i-1}, g\left(\overline{x}^{i-2}, \ldots g\left(\overline{x}^0, \overline{z}^0\right)\right)\right)\right)$$

kde hodnoty $\overline{x}^0$ a $\overline{z}^0$ jsou počáteční hodnoty vektorů. Vektor vnitřních proměnných představuje počáteční stav logického sekvenčního systému.

Základní zapojení bloků logického sekvenčního obvodu je uvedeno na obr. 4-2, kde blok označený z_p realizuje zpoždění mezi jednotlivými diskrétními body a LKO je logický kombinační obvod.



Obr. 4-2: Blokové schéma logického sekvenčního obvodu

Zpožďovací člen z_p můžeme realizovat: Zpožďovací člen

- zpožděním v logických členech,
- zpožďovacími linkami,
- paměťovými členy.

Podle vlastností funkcí výstupů se mohou sekvenční obvody dělit na:

Mealyho a Moorův typ sekvenčního obvodu

- obvody **Mealyho** typu, kde výstupy jsou funkcemi vstupních a vnitřních proměnných, podle rovnice *(4-2)*
- obvody **Moorova** typu, kde funkce výstupů jsou pouze funkcí vnitřních proměnných; rovnice *(4-2)* pro Moorův typ potom přechází na tvar:

$$y_j^i = f_j\left(z_j^i\right) \qquad (4\text{-}6)$$

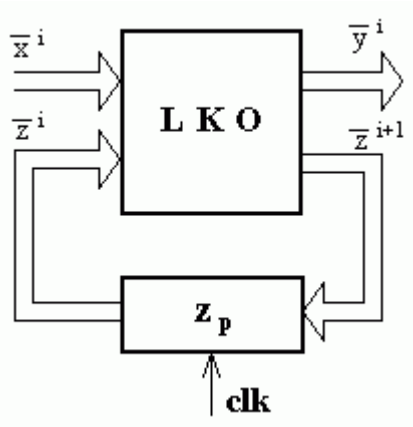
Na základě vlastností použitého typu zpožďovacího členu se mohou sekvenční obvody dále dělit na obvody:

Synchronní a asynchronní obvody

- **synchronní -** ke změnám vnitřních proměnných dochází téměř současně,
- **asynchronní -** ke změnám vnitřních proměnných dochází v závislosti na tom, jak se šíří podnět obvodem.

V synchronních sekvenčních obvodech se používají ve zpožďovací funkci výhradně paměťové členy (někdy označené též jako klopné obvody), které jsou řízeny zdrojem hodinových impulsů. Tyto impulsy synchronizují činnost všech paměťových členů, viz. obr. 4-3. Časový interval mezi hodinovými impulsy se volí tak, aby v obvodu před příchodem následujícího hodinového impulsu odezněly přechodné jevy. Z tohoto důvodu se také neuplatní hazardní stavy, kterými může být zatížena kombinační část systému. Neuplatní se ani hazardní stavy typické pro asynchronní sekvenční obvody. Jsou možné současné změny vstupních proměnných za předpokladu, že k jejich změně nedochází v době trvání hodinového impulsu nebo v okolí náběžné nebo sestupné hrany hodinového impulsu. Tyto okolnosti však závisí na použitých paměťových členech. U těchto jsou v integrované formě podmínky pro správnou činnost a způsob synchronizace určeny výrobcem (katalog).

Synchronní sekvenční obvod



Obr. 4-3: Blokové schéma synchronního sekvenčního obvodu

Asynchronní sekvenční obvod

Pro realizaci asynchronního sekvenčního obvodu lze použít shodné základní logické členy jako pro kombinační obvody. Potřebné paměťové vlastnosti zajistíme zavedením vhodných zpětných vazeb. Zpožďovací člen asynchronního sekvenčního obvodu lze realizovat paměťovými členy nebo jen zpožděním v logických členech.

Použití paměťových členů pro paměťové funkce má řadu výhod. Podstatnou výhodou je relativně jednoduchý návrh. Tímto návrhem je současně automaticky vyloučena možnost vzniku statických hazardů. U asynchronních obvodů obvykle předpokládáme, že zpoždění v logických členech se soustředí do zpětnovazebních smyček. Ačkoliv předpokládáme ve zpětnovazebních smyčkách stejné zpoždění Δt nelze u reálného obvodu tento předpoklad splnit.

Předpoklad shodného zpoždění umožňuje vyloučit rozměr času z logických výrazů a provádět řešení obvodu podobnými metodami jako u kombinačních obvodů. Nemožnost splnit předpoklad shodného zpoždění znamená, že nelze počítat se současnou změnou dvou nebo více vnitřních proměnných. V případech, ve kterých by mělo dojít k současné změně vnitřních proměnných, je nutné provést analýzu všech hazardních stavů, které se v důsledku nedosažení současné změny mohou objevit.

Způsoby řízení asynchronních obvodů

V závislosti na době trvání změny logické hodnoty vstupních proměnných $\overline{x}^i$ rozeznáváme dva způsoby řízení asynchronních obvodů:

- **základní způsob řízení** - po změně vektoru $\overline{x}^i$ smí jeho další změna následovat až po dosažení nového stabilního stavu obvodu,

- **impulsní řízení** - k řízení se používá impuls na jednom ze vstupů; u těchto obvodů se obvykle předpokládá, že vstupní signály se nepřekrývají a k možnosti dalšího vstupního impulsu dochází až po ustálení odezvy na předchozí impuls;
  každý generovaný impuls způsobí jen jednu změnu vnitřního stavu obvodu - z tohoto důvodu musí být aktivní šířka vstupního impulsu kratší než doba odezvy obvodu a současně musí být dostatečně dlouhá na to, aby tuto odezvu vyvolala.

Každý logický sekvenční obvod je úplně specifikován šesti veličinami:

$$LSO = \left\lfloor \overline{x}, \overline{y}, \overline{z}, f, g, \overline{z}^0 \right\rfloor \qquad (4\text{-}7)$$

kde $\overline{x}, \overline{y}$ a $\overline{z}$ jsou vektory vstupních, výstupních a vnitřních proměnných, f a g jsou funkce výstupů a přechodů a $\overline{z}^0$ je počáteční vektor vnitřních proměnných (počáteční vnitřní stav).

Při analýze a návrhu sekvenčního logického obvodu se můžete setkat s následujícími pojmy:

Pojmy

Vnitřní stav obvodu může být vyjádřen kombinací hodnot vnitřních proměnných (při analýze nebo konečné fázi návrhu obvodu) nebo obecným symbolem (při návrhu obvodu).

Vnitřní stav

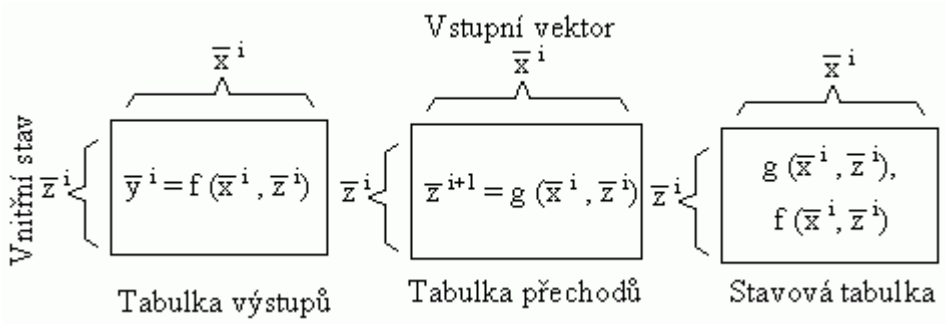
Stabilní stav asynchronního obvodu je takový stav, ve kterém obvod při konstantním vstupním vektoru $\overline{x}^i$ může setrvávat neomezeně dlouhou dobu. Matematicky jej lze vyjádřit rovnicemi *(4-8),* v tabulkách jej označujeme kroužkem nebo zvýrazníme šedě.

Stabilní stav

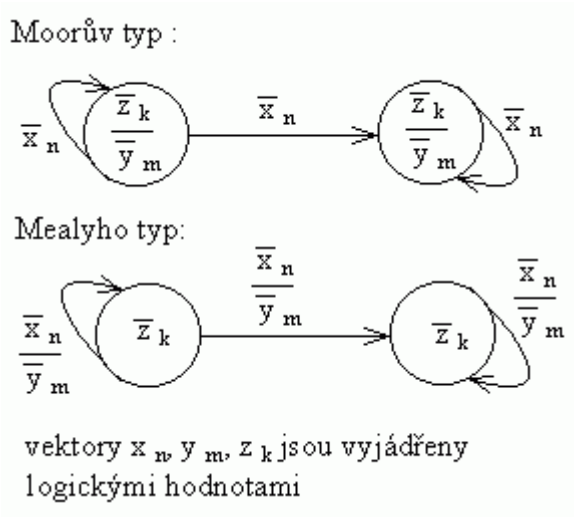
$$\overline{x}^{i+1} = \overline{x}^i, \qquad \overline{z}^{i+1} = \overline{z}^i \qquad (4\text{-}8)$$

Funkce přechodů a výstupů mohou být reprezentovány :

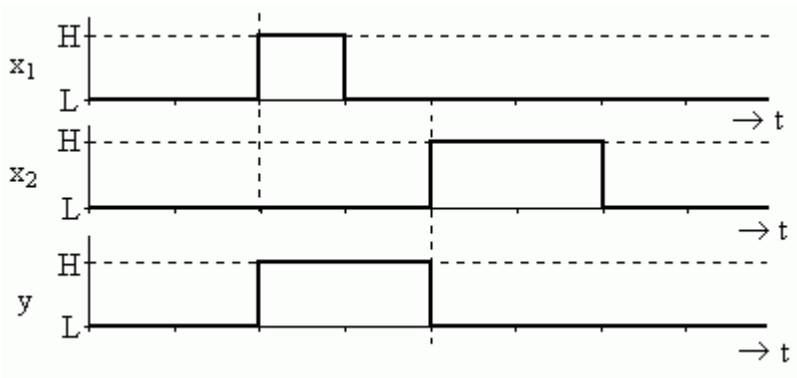
- **Algebraickým výrazem** - využití při analýze a v závěrečné fázi návrhu sekvenčního obvodu při přechodu od jeho struktury ke stavové tabulce (případně obráceně). Například $Q^{i+1} = S + \overline{R} \cdot Q^i$.
- **Tabelárním vyjádřením** viz. obr.4-4. - k označení sloupců a řádků se velmi často používají n-tice logických hodnot daných vektorů.
- **Grafickým vyjádřením** viz. obr.4-5 - tabulky přechodů a výstupů lze převést na orientovaný graf skládající se z uzlů a přechodů.
- **Časovým diagramem** vstupních a výstupních proměnných obvodu viz. obr.4-6 - do tohoto grafu lze také včlenit do jisté míry i vnitřní stavy.
- **Vývojovým (programovým) diagramem** - tento způsob zápisu se používá při popisu chování složitých sekvenčních soustav (řadiče, procesory PC), kde vystupuje velký počet proměnných a klasické prostředky reprezentace jsou obtížně sestavitelné a později i značně nepřehledné. Vývojové diagramy obsahují tři základní typy prvků, které jsou uvedeny na obr.4-7.
- **Programem činnosti** LSO - k popisu chování složitých sekvenčních soustav někteří autoři zavádí programovací jazyk vhodný k popisu činnosti těchto soustav. V jazyce vystupují instrukce typu např. zápis n-bitové slabiky do registru, zvětšení/zmenšení obsahu registru o jedničku při splnění definované podmínky, atd. Chování obvodu potom popisuje program vytvořený z instrukcí tohoto jazyka.



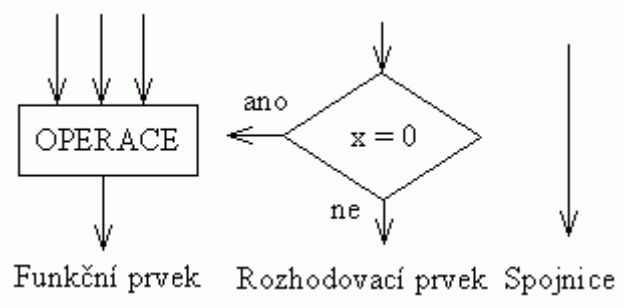
Obr. 4-4: Vyjádření funkcí pomocí tabulek



Obr. 4-5: Příklady orientovaného grafu



Obr. 4-6: Časové průběhy proměnných



Obr. 4-7: Označení prvků pro vývojové diagramy

KONTROLNÍ OTÁZKY 2

a) Jakým způsobem lze realizovat zpožďovací člen?

b) Jak lze rozdělit na základě použitého typu zpožďovacího členu sekvenční obvody?

c) Jaké rozeznáváme způsoby řízení asynchronních obvodů?

d) Za jaké podmínky může asynchronní obvod setrvat v daném stavu?

e) Čím lze popsat funkce přechodů a výstupů (6 druhů)?

4.1 Analýza logických sekvenčních obvodů

Cílem analýzy sekvenčního obvodu je odvození chování známé struktury, což představuje odvození funkcí výstupů f a funkcí přechodů g a jejich reprezentace stavovou tabulkou nebo stavovým diagramem. Postup se člení podle struktury obvodu na:

- analýzu sekvenčních obvodů bez paměťových členů
- analýzu sekvenčních obvodů s paměťovými členy

4.1.1 Analýza sekvenčních obvodů bez paměťového členů

Do této skupiny logických sekvenčních obvodů patří obvody asynchronní. Logická síť je vytvořena z logických členů, přičemž obsahuje zpětné vazby. Příklad takovéhoto obvodu je na obr. 4-8. Pro dosažení modelu z obr. 4-8, vyčleníme v prvním přiblížení zpoždění z jednotlivých logických členů a soustředíme jej ve formě zpožďovacích členů do zpětnovazebních smyček. Obvod je pak tvořen logickými členy bez zpoždění a zpožďovacími členy, které realizují paměťovou funkci, obr. 4-9.

Analýza obvodu se skládá z těchto kroků:

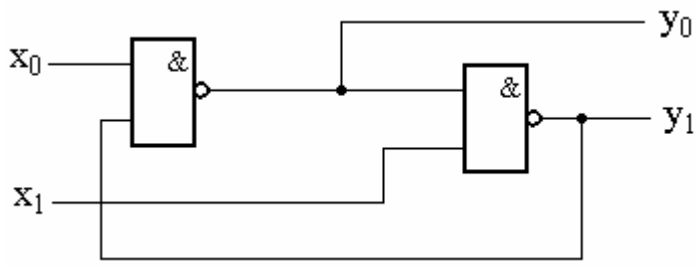
- Určení množiny vnitřních stavů $\bar{z}$ (stanovení nemusí být jednoznačné – analýzu to však neovlivní). V obvodu nalezneme všechny zpětnovazební smyčky a výstupy logických členů, které jsou součástí zpětnovazebních smyček, označíme za vnitřní proměnné. Vnitřní proměnné volíme tak, aby jejich celkový počet byl minimální. Za výstup, kde jsme zvolili vnitřní proměnnou z_k, zařadíme zpožďovací člen.
- Přerušení zpětnovazebních smyček. Na vstupu každého zpožďovacího členu definujeme proměnnou z_k^{i+1} a na jeho výstupu proměnnou z_k^i. Tímto způsobem je možné na jednom vodiči definovat dva různé stavy a analyzovat tak přechodové děje v obvodu.
- Rozpojením všech zpětnovazebních smyček se ze sekvenčního obvodu stal obvod pseudokombinační, u kterého snadno stanovíme algebraické výrazy pro funkci přechodů a výstupů.
- Získané funkce přechodů a výstupů zapíšeme do stavové tabulky, určíme stabilní stavy a chování obvodu

Pro demonstraci postupu analýzy asynchronního sekvenčního obvodu použijeme zapojení uvedené na obr. 4-8. Je zřejmé, že v obvodu je jedna zpětnovazební smyčka, přičemž dle způsobu kreslení můžeme uvažovat smyčku tvořenou výstupním signálem y_1, resp. y_0. Označením vnitřních proměnných, přerušením zpětné vazby a zapojením zpětnovazebního členu dostaneme zapojení podle obr. 4-9. Z obr. 4-9 lze potom snadno psát rovnici přechodů a výstupů:

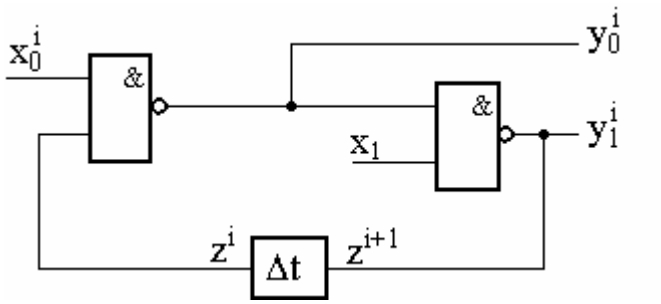
$$z^{i+1} = g(x_0^i, x_1^i, z^i) = \overline{(\overline{x_0^i \cdot z^i}) \cdot x_1^i} = \overline{x_1^i} + x_0^i \cdot z^i \qquad (4\text{-}9)$$

$$y_0^i = f_1(x_0^i, x_1^i, z^i) = \overline{x_0^i \cdot z^i} = \overline{x_0^i} + \overline{z^i} \qquad (4\text{-}10)$$

$$y_1^i = f_2(x_0^i, x_1^i, z^i) = z^{i+1} \qquad (4\text{-}11)$$



Obr. 4-8: Asynchronní logický sekvenční obvod



Obr. 4-9: Asynchronní sekvenční obvod s vyjádřenou paměťovou funkcí

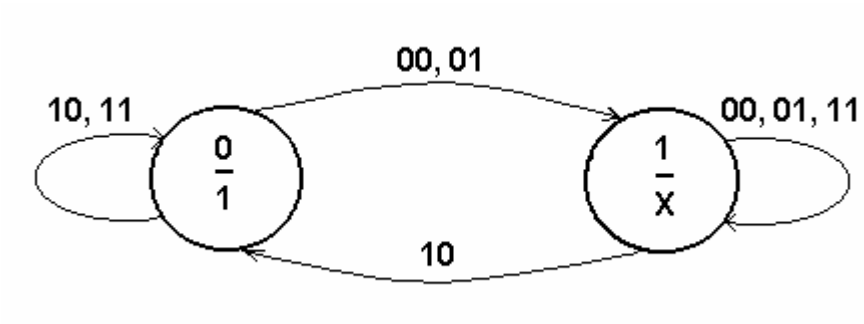
$y_1^i = z^{i+1}$

z^i \ $x_1 x_0$	00	01	10	11
0	1	1	0	0
1	1	1	0	1

y_0^i

z^i \ $x_1 x_0$	00	01	10	11
0	1	1	1	1
1	1	0	1	0

Obr. 4-10: Tabulka přechodů a výstupů



Obr. 4-11: Orientovaný graf

Zapsáním rovnice *(4-9)* do tabulky přechodů a rovnice *(4-10)* do tabulky výstupů dostaneme obr. 4-10.

Pokud do stavové tabulky vyznačíme stabilní stavy obvodu (tj. stavy, kdy platí $z^{i+1} = z^i$), lze nakreslit graf chování obvodu při změně vstupních proměnných, tzv. orientovaný graf viz. obr. 4-11.

Jestliže předpokládáme základní stabilní stav obvodu $z^i = 0$, do stavu 1 se obvod dostane změnou proměnné x_1 na hodnotu logická 0. Ze stavu $z^i = 1$ do stavu $z^{i+1} = 0$ musí být splněna podmínka $x_1 = 1$ a $x_0 = 0$.

4.1.2 Analýza sekvenčních obvodů s paměťovými členy

Paměťové členy jsou logické sekvenční obvody, které mají dva různé stavy a využívají se jako paměť hodnoty logické proměnné. Paměťové členy se využívají k realizaci čítačů, registrů atd., a lze je podle jejich vlastností rozdělit na

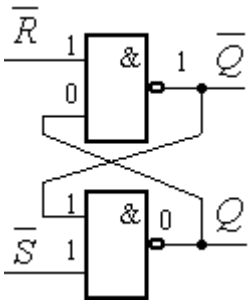
- asynchronní řízení,
- synchronní řízení:
  - hladinovým signálem,
  - vzestupnou hranou synchronizačního signálu,
  - sestupnou hranou synchronizačního signálu.

PAMĚŤOVÝ ČLEN RS

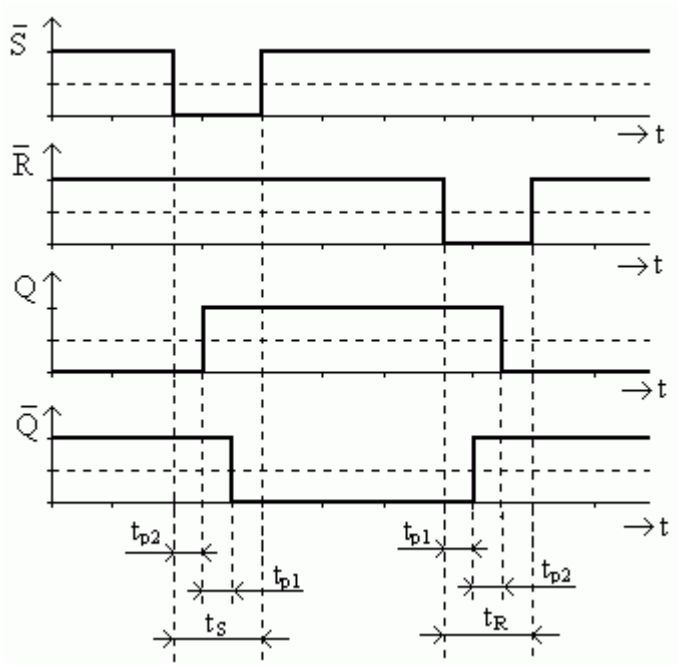
Paměťový člen RS představuje asynchronní obvod řízený dvěma vstupními signály. Řídícími signály jsou vstupy *R* (Reset) a *S* (Set). K praktické realizaci se používá obvodů typu NAND nebo typu NOR.

Člen RS realizovaný pomocí obvodů NAND

Paměťový člen RS realizovaný z obvodů NAND. Zapojení obvodu je uvedeno na obr. 4-12. Protože vstupní signály *R* a *S* jsou aktivní v logické 0 jsou ve schématu uvedeny jako negované. Časové průběhy popisující chování obvodu vychází ze zakresleného výchozího stavu. Činnost obvodu je zřejmá z uvedeného průběhu na obr. 4-13. Z obr. 4-13 je vidět i potřebná šířka impulsu na vstupech $\overline{R}$ a $\overline{S}$. Impuls musí mít větší šířku, než je doba určená součtem zpoždění hradel. Pro šířku impulsu tedy platí, $t_S \geq (t_{p2} + t_{p1})$ a analogicky $t_R \geq (t_{p1} + t_{p2})$. Jestliže jsou zpoždění obou hradel shodná, pak platí $t_S \geq 2.t_p$ a $t_R \geq 2.t_p$.



Obr. 4-12: Paměťový člen RS sestavený z hradel typu NAND



Obr. 4-13: Časové průběhy paměťového členu RS

$\overline{R^i}$	$\overline{S^i}$	Q^{i+1}
0	0	Zakázaný stav
0	1	0
1	0	1
1	1	Q^i

Obr. 4-14: Modifikovaná tabulka výstupů (pravdivostní tabulka)

Modifikovanou tabulku výstupů obvodu (často nazývána pravdivostní tabulka) lze odvodit ze zapojení uvedeného na obr. 4-12. Předpokládáme-li, že výstup Q bude totožný se stavovou proměnnou z^{i+1}, tedy $Q = z^{i+1} = Q^{i+1}$, můžeme sestavit rovnici (nazýváme ji charakteristická rovnice):

$$Q^{i+1} = \overline{\overline{S^i} \cdot \left(\overline{\overline{R^i} \cdot Q^i}\right)} = S^i + \overline{R^i} \cdot Q^i \qquad (4\text{-}12)$$

Stav $\overline{R^i} = 0$ a současně $\overline{S^i} = 0$ je označován jako zakázaný stav, to i přes skutečnost, že po dosazení do rovnice *(4-12)* platí $Q^{i+1} = 1 + 0 = 1$. V tomto případě je však problémem porušení vazby negací mezi přímým výstupem Q a negovaným výstupem $\overline{Q}$. Protože platí nesmyslný stav, kdy $Q = \overline{Q} = 1$.

Z tabulky na obr. 4-14 nebo z rovnice *(4-12)* lze potom odvodit tabulku na obr. 4-15 uvádějící podmínky přechodu mezi jednotlivými možnými stavy.

$Q^i \rightarrow Q^{i+1}$	$\overline{R^i}$	$\overline{S^i}$
$0 \rightarrow 0$	X	1
$0 \rightarrow 1$	1	0
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	1	X

Obr. 4-15: Podmínky přechodu mezi jednotlivými stavy

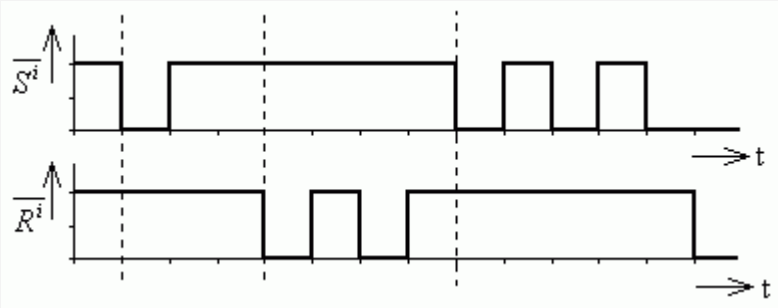
PRŮVODCE STUDIEM 6

Můžete se také setkat s následujícími pojmy:

- $Q^{i+1} = X$ označeno Zakázaný stav
- $Q^{i+1} = 1$ označeno Jednotkový stav
- $Q^{i+1} = 0$ označeno Nulový stav
- $Q^{i+1} = Q^i$ označeno Pamatovací stav

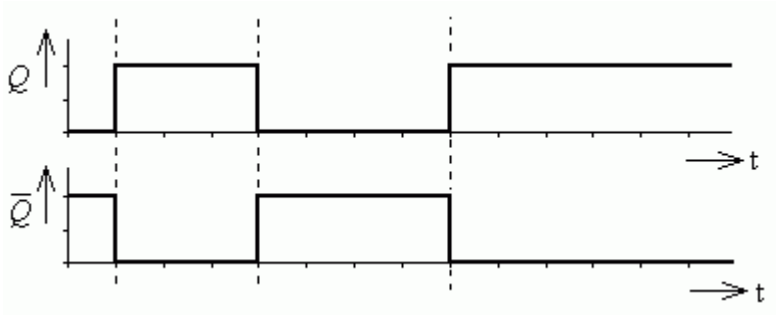
ŘEŠENÝ PŘÍKLAD

Zakreslete do časového diagramu chování paměťového členu RS realizovaného pomocí hradel NAND. Jsou zadány časové průběhy vstupních signálů:



Řešení příkladu

Grafické řešení příkladu. Průběhy výstupních proměnných.



*

Paměťový člen RS realizovaný z obvodů NOR. Paměťový člen RS realizovaný z obvodů NOR je opět asynchronně řízený obvod. Řízení je opět provedeno pomocí vstupních proměnných *R* a *S*. Oproti realizaci s členy NAND jsou však signály aktivní v hodnotě logická 1. Schéma zapojení realizace tohoto členu je na obr. 4-16. Časové průběhy popisující činnost obvodu jsou zobrazeny na obr. 4-17. Z obr. 4-17 je patrná potřebná šířka setovacího, resp. resetovacího impulsu. Pro šířku impulsu při shodné hodnotě zpoždění jednotlivých hradel platí

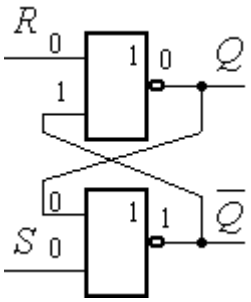
$t_R \geq 2.t_p$ a $t_S \geq 2.t_p$.

Člen RS realizovaný pomocí obvodů NOR

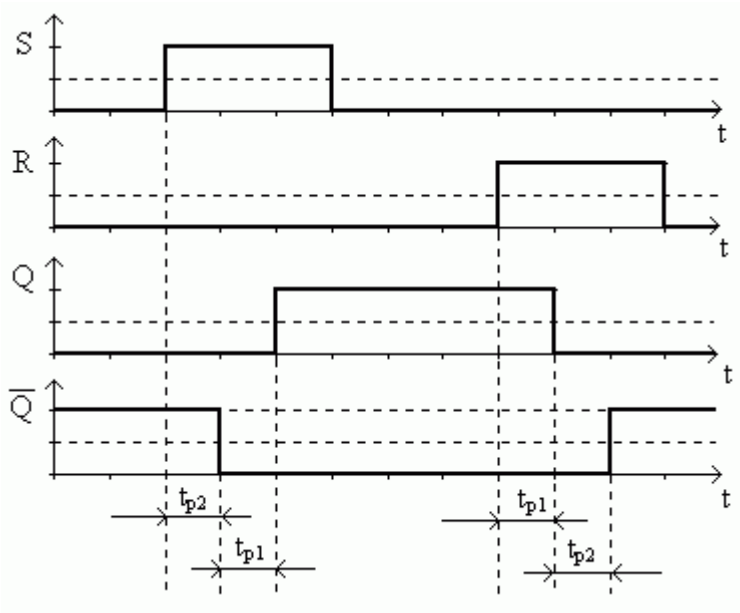
Při přerušení vazby z výstupu hradla označeného 1 a při přechodu do oblasti diskrétního času *i*, můžeme psát charakteristickou rovnici ve tvaru:

$$Q^{i+1} = \overline{R^i + \overline{\left(S^i + Q^i\right)}} = \overline{R^i} \cdot \left(S^i + Q^i\right) \qquad (4\text{-}13)$$

Z rovnice *(4-13)* lze vytvořit stavovou tabulku na obr. 4-18. Ze stavové tabulky jsou po vyznačení stabilních stavů zřejmé i tabulky popisující chování obvodu na obr. 4-19 a obr. 4-20.



Obr. 4-16: Paměťový člen RS sestavený z hradel typu NOR



Obr. 4-17: Časové průběhy paměťového členu RS s obvody NOR

Q^{i+1} \ $R^i S^i$	00	01	10	11
Q^i 0	0	1	0	0
1	1	1	0	0

Obr. 4-18: Stavová tabulka RS obvodu

R^i	S^i	Q^{i+1}
0	0	Q^i
0	1	1
1	0	0
1	1	Zakázaný stav

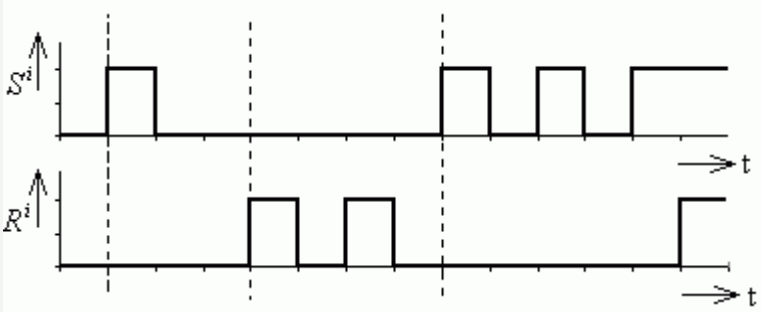
Obr. 4-19: Modifikovaná tabulka výstupů

$Q^i \rightarrow Q^{i+1}$	R^i	S^i
$0 \rightarrow 0$	X	0
$0 \rightarrow 1$	0	1
$1 \rightarrow 0$	1	0
$1 \rightarrow 1$	0	X

Obr. 4-20: Podmínky přechodu mezi jednotlivými stavy

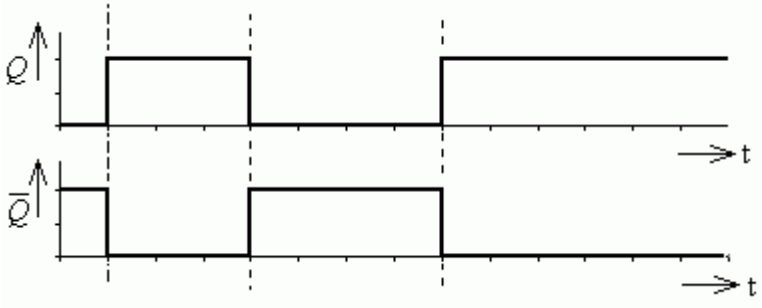
ŘEŠENÝ PŘÍKLAD

Zakreslete do časového diagramu chování paměťového členu RS realizovaného pomocí hradel NOR. Jsou zadány časové průběhy vstupních signálů:



Řešení příkladu

Grafické řešení příkladu. Průběhy výstupních proměnných.



*

HLADINOVĚ ŘÍZENÉ PAMĚŤOVÉ ČLENY

Tato skupina obvodů představuje paměťové členy, které jsou synchronně řízené hladinovým signálem. Logická hodnota výstupních proměnných je dána nejen logickými hodnotami vstupních proměnných, ale i logickou hodnotou synchronizačního signálu. V důsledku toho může ke změně logické hodnoty výstupních proměnných dojít pouze v okamžiku, když je synchronizační signál aktivní. Pokud synchronizační signál není aktivní, pak nemůže dojít ke změně vektoru výstupních proměnných a obvod setrvá ve stabilním stavu.

Hladinově řízené paměťové členy

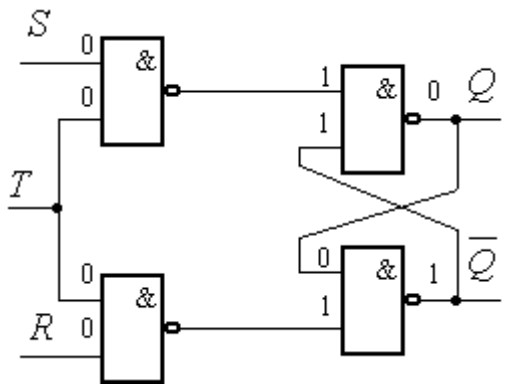
PAMĚŤOVÝ ČLEN RST

Základ hladinově řízeného paměťového členu RST je tvořen paměťovým členem RS. Pro realizaci členu RS se používají realizace pomocí hradel NAND nebo NOR. Synchronizační signál se označuje symbolem *T*.

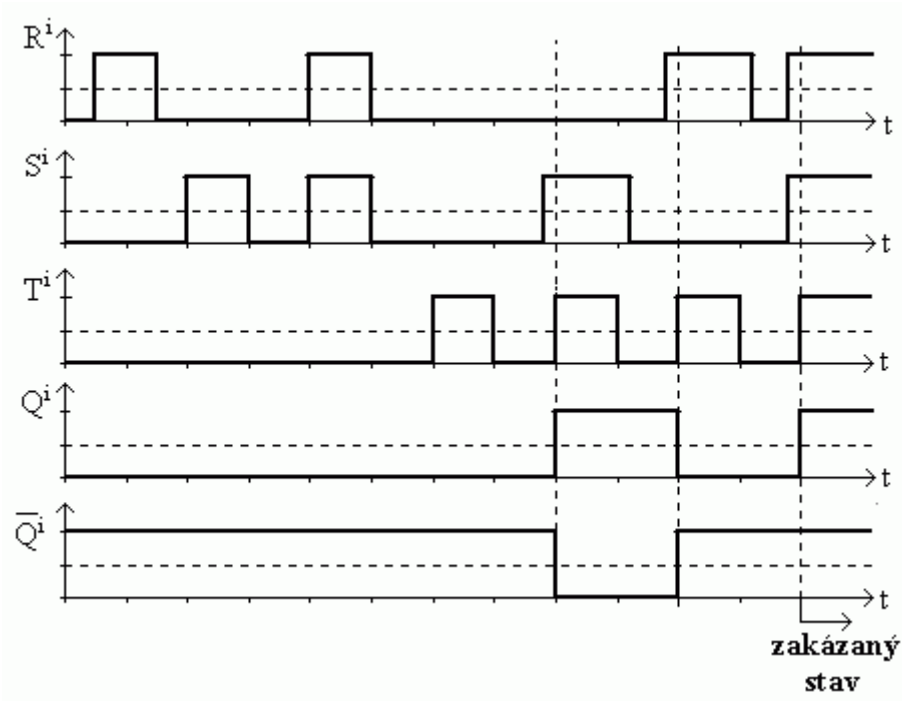
Člen RST

Paměťový člen RST realizovaný pomocí hradel NAND. Schéma zapojení je na obr. 4-21. Na obrázku je zároveň vyznačen výchozí stav pomocí konkrétních logických hodnot.

Člen RST realizovaný pomocí obvodů NAND



Obr. 4-21: Paměťový člen RST sestavený z hradel typu NAND



Obr. 4-22: Časové průběhy členu RST z hradel NAND

Pro analýzu tohoto paměťového členu přerušíme zpětnou vazbu z výstupu Q, v důsledku přerušení lze psát:

$$Q^{i+1} = \overline{\overline{S^i \cdot T^i} \cdot \overline{\left(Q^i \cdot \overline{\left(R^i \cdot T^i\right)}\right)}} \qquad (4\text{-}14)$$

Po úpravě dostáváme charakteristickou rovnici ve tvaru:

$$Q^{i+1} = S^i \cdot T^i + Q^i \cdot \left(\overline{R^i} + \overline{T^i}\right) \qquad (4\text{-}15)$$

Časové průběhy činnosti obvodu jsou uvedeny na obr. 4-22. Znázornění zanedbává možná zpoždění hradel.

Je zřejmé, že pro $R^i = S^i = T^i =$ log I bude platit $Q^i = \overline{Q^i} = \log I$. Opět se tedy jedná o zakázaný stav obvodu RS. Stavová tabulka členu RST je uvedena na obr. 4-23. Ze stavové tabulky vyplývá, že ke změně stavu dochází výhradně při splnění podmínky $T = 1$. Při akceptování této skutečnosti lze tabulku přechodů mezi stavy sestavit bez této proměnné (obr. 4-24 a obr. 4-25).

Q^{i+1} \ $R^i S^i T^i$	000	001	010	011	100	101	110	111
Q^i 0	0	0	0	1	0	0	0	1
1	1	1	1	1	1	0	1	1

Obr. 4-23: Stavová tabulka členu RST z hradel NOR

R^i	S^i	Q^{i+1}
0	0	Q^i
0	1	1
1	0	0
1	1	Zakázaný stav

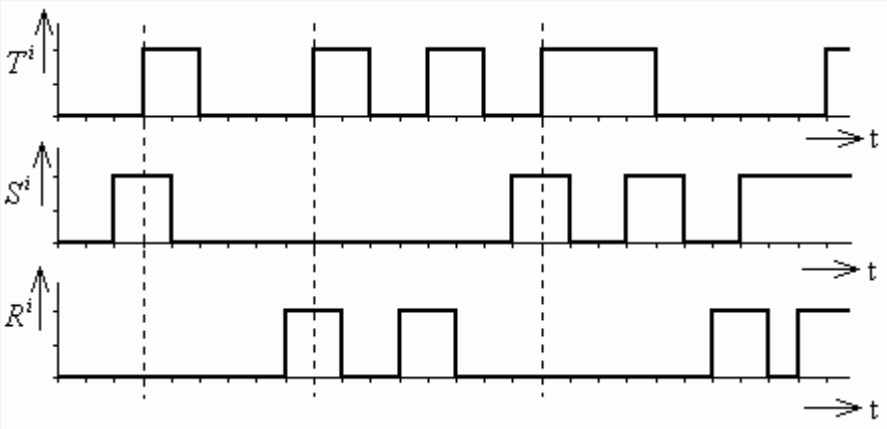
Obr. 4-24: Modifikovaná tabulka výstupů

$Q^i \rightarrow Q^{i+1}$	R^i	S^i
$0 \rightarrow 0$	X	0
$0 \rightarrow 1$	0	1
$1 \rightarrow 0$	1	0
$1 \rightarrow 1$	0	X

Obr. 4-25: Podmínky přechodu mezi jednotlivými stavy

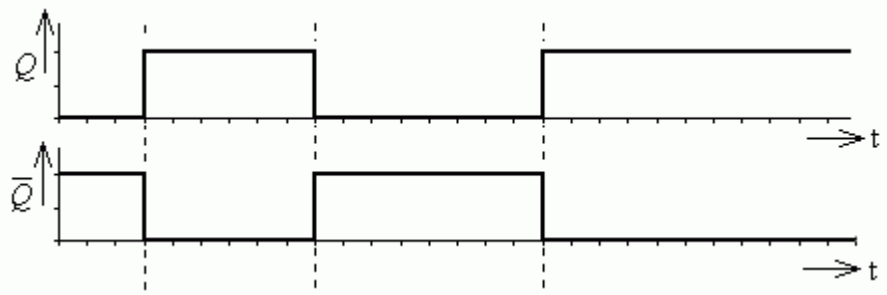
ŘEŠENÝ PŘÍKLAD

Zakreslete do časového diagramu chování paměťového členu RST sestaveného z hradel typu NAND. Jsou zadány časové průběhy vstupních signálů:



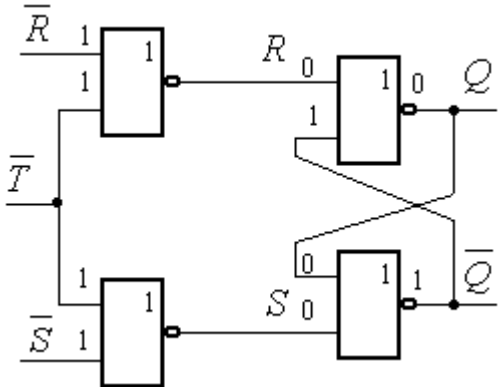
Řešení příkladu

Grafické řešení příkladu. Průběhy výstupních proměnných.

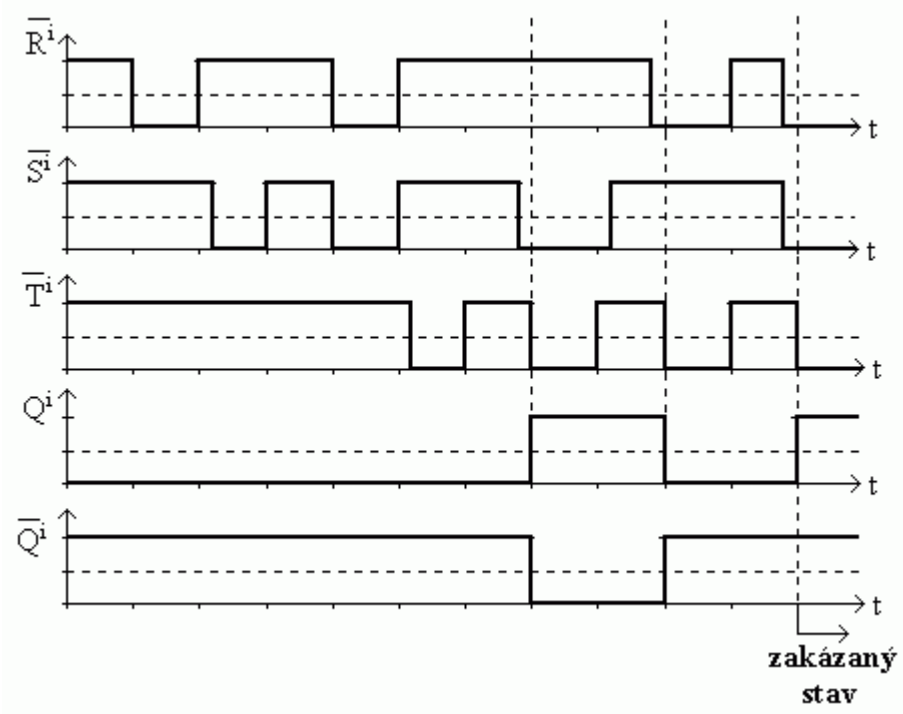


Člen RST realizovaný z hradel NOR. Schéma zapojení členu RST realizovaného z hradel NOR je uvedeno na obr. 4-26, také je zde vyznačen základní stav obvodu. S ohledem na pravdivostní tabulku paměťového členu RS sestaveného z hradel NOR je nutné uvažovat s negovanými vstupními proměnnými. Časové průběhy jsou potom uvedeny na obr. 4-27.

Člen RST realizovaný pomocí hradel NOR



Obr. 4-26: Zapojení obvodu RST ze členů typu NOR



Obr. 4-27: Časové průběhy členu RST z hradel NOR

Po přerušení zpětné vazby z výstupu Q lze psát:

$$Q^{i+1} = \overline{\overline{\overline{R^i} + \overline{T^i}} + \overline{\overline{\overline{S^i} + \overline{T^i}} + Q^i}} \qquad (4\text{-}16)$$

Po úpravě dostaneme charakteristickou rovnici ve tvaru:

$$Q^{i+1} = \overline{R^i} \cdot S^i \cdot T^i + Q^i \cdot \left(\overline{R^i} + \overline{T^i}\right) \qquad (4\text{-}17)$$

Stavovou tabulku na obr. 4-28 vytvoříme na základě výrazu *(4-17)*. Jestliže akceptujeme, že ke změně stavu členu dochází za podmínky $T^i = 0$, platí tabulky přechodů na obr. 4-29 a na obr. 4-30.

Q^{i+1} \ $\overline{R^i}\,\overline{S^i}\,\overline{T^i}$		000	001	010	011	100	101	110	111
Q^i	0	0	0	0	0	1	0	0	0
	1	1	1	0	1	1	1	1	1

Obr. 4-28: Stavová tabulka členu RST z hradel NOR

$\overline{R^i}$	$\overline{S^i}$	Q^{i+1}
0	0	Zakázaný stav
0	1	0
1	0	1
1	1	Q^i

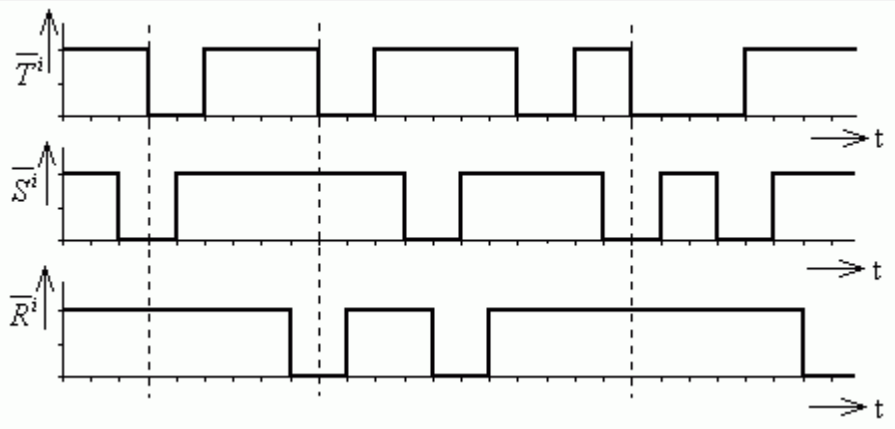
Obr. 4-29: Modifikovaná tabulka výstupů

$Q^i \rightarrow Q^{i+1}$	$\overline{R^i}$	$\overline{S^i}$
$0 \rightarrow 0$	X	1
$0 \rightarrow 1$	1	0
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	1	X

Obr. 4-30: Podmínky přechodu mezi jednotlivými stavy

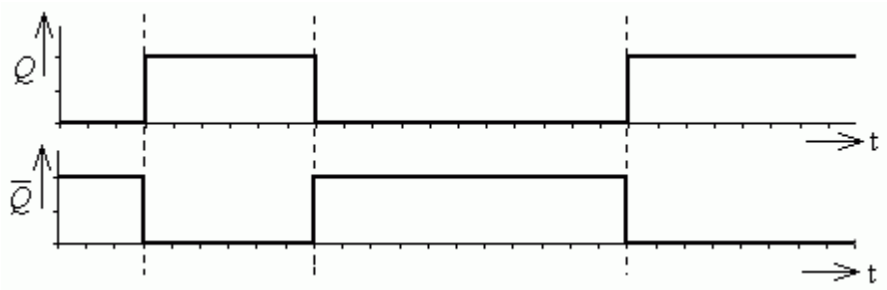
ŘEŠENÝ PŘÍKLAD

Zakreslete do časového diagramu chování paměťového členu RST sestaveného z hradel typu NOR. Jsou zadány časové průběhy vstupních signálů:



Řešení příkladu

Grafické řešení příkladu. Průběhy výstupních proměnných.



*

PAMĚŤOVÝ ČLEN D HLADINOVĚ ŘÍZENÝ

Zapojení paměťového členu D je na obr. 4-31 a jeho základ tvoří paměťový člen RS. Vstupním signálem tohoto členu je signál *D* hradlovaný synchronizačním signálem *T*. Podmínkou *T* = 0 je zajištěn základní stav RS obvodu. K realizaci členu je použito hradel NAND.

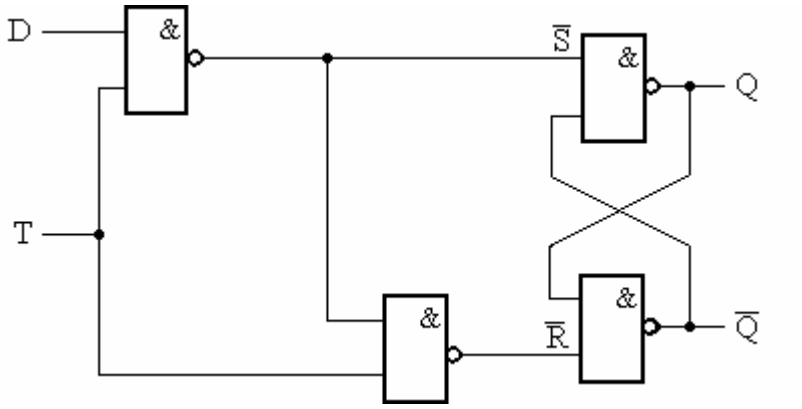
Přerušíme-li zpětnou vazbu z výstupního signálu *Q*, můžeme psát:

$$Q^{i+1} = \overline{\overline{D^i \cdot T^i} \cdot \overline{\overline{T^i \cdot \overline{D^i T^i}} \cdot Q^i}} \qquad (4\text{-}18)$$

a po úpravě dostaneme

$$Q^{i+1} = D^i \cdot T^i + Q^i \cdot \overline{T^i} \qquad (4\text{-}19)$$

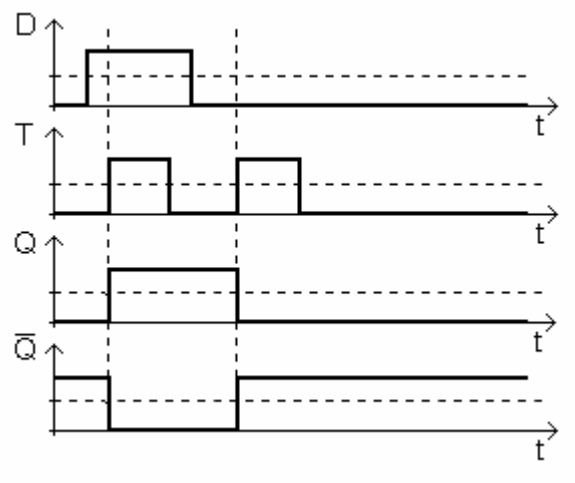
Z rovnice *(4-19)* sestavíme tabulku stavů na obr. 4-32 a časové průběhy na obr. 4-33. Ze stavové tabulky vyplývá, že ke změně stavu dochází při *T* = 1 a logická hodnota výstupní proměnné *Q* kopíruje logické hodnoty vstupní proměnné *D*. Pokud *T* = 0, je zapamatován stav zapsaný v RS obvodu.



Obr. 4-31: Paměťový člen D

Q^{i+1} \ $D^i\ T^i$	00	01	10	11
Q^i 0	0	0	0	1
1	1	0	1	1

Obr. 4-32: Stavová tabulka členu D



Obr. 4-33: Časové průběhy členu D

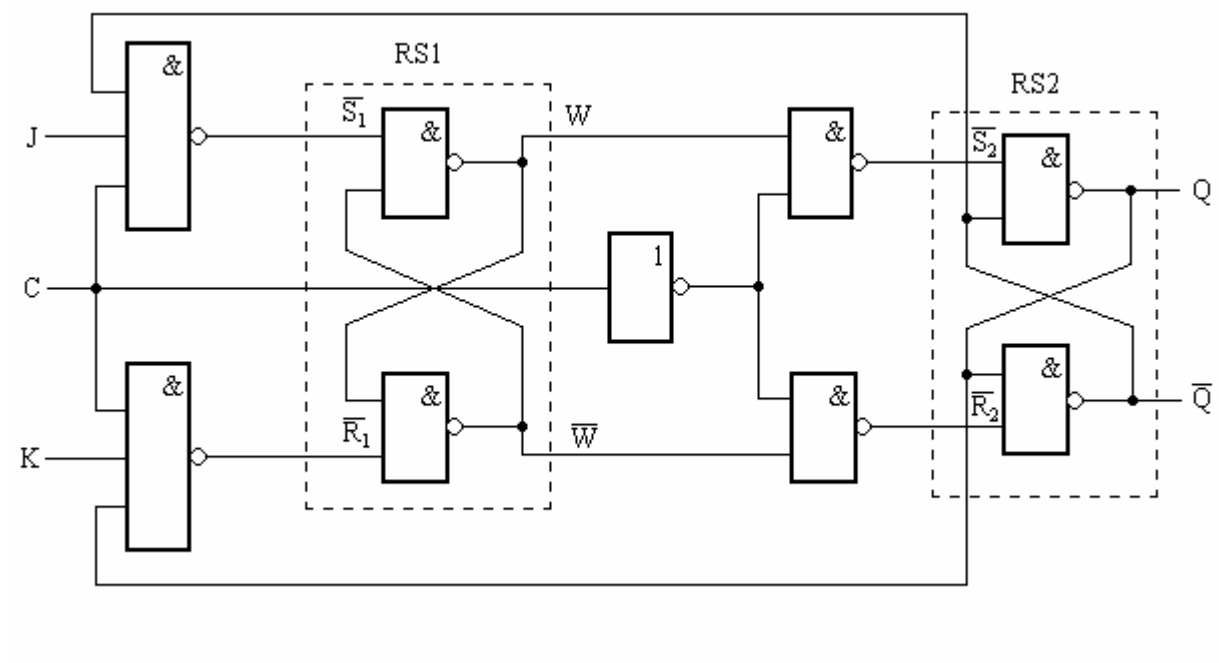
PAMĚŤOVÉ ČLENY SYNCHRONIZOVANÉ HRANOU SIGNÁLU

Paměťové členy synchronizované hranou signálu představují skupinu členů, ve které dochází ke změně logické hodnoty výstupní proměnné v okamžiku změny synchronizačního signálu. Lze říci, že vstupní proměnné jsou vzorkovány synchronizačním signálem, který se u těchto členů většinou značí *C* (clock).

DVOUFÁZOVÝ PAMĚŤOVÝ ČLEN JK

Paměťový člen JK se řídí dvěma synchronními vstupy – J, K v závislosti na hodinových impulsech C (je řízen vzestupnou hranou). Jeho činnost lze rozdělit do dvou fází. Vnitřní struktura tohoto členu je na obr. 4-34.

K realizaci tohoto členu jsou použita hradla NAND a základ tvoří dva hradlové RS obvody se synchronizačním signálem C. RS1 je hradlován přímým signálem C a RS2 signálem negovaným $\overline{C}$. JK proto představuje člen s dvoufázovým přenosem dat, u kterého na náběžnou hranu synchronizačního signálu je zapsána vstupní informace do vnitřního RS obvodu – RS1 a na sestupnou hranu se stav vnitřního RS obvodu přepíše do výstupního RS obvodu – RS2. Název obvodu je odvozen od názvu vstupních proměnných, které jsou J a K. Vnitřní RS obvod má výstupní proměnné označeny W a výstupní RS obvod Q.



Obr. 4-34: Struktura paměťového členu JK

Pro analýzu činnosti obvodu přerušíme zpětné vazby z výstupu Q a W a můžeme pro RS1 psát

$$W^{i+1} = \overline{\overline{\overline{S_1^i} \cdot (\overline{\overline{R_1^i} \cdot W^i})}} = S_1^i + \overline{R_1^i} \cdot W^i \qquad (4\text{-}20)$$

a pro RS2

$$Q^{i+1} = \overline{\overline{\overline{S_2^i} \cdot (\overline{\overline{R_2^i} \cdot Q^i})}} = S_2^i + \overline{R_2^i} \cdot Q^i, \qquad (4\text{-}21)$$

kde proměnné $\overline{S_1^i}, \overline{R_1^i}, \overline{S_2^i}, \overline{R_2^i}$ jsou vstupní proměnné jednotlivých RS členů, pro které platí následující vztahy

$$\overline{S_1^i} = \overline{J^i \cdot C \cdot \overline{Q^i}} = \overline{J^i} + \overline{C} + Q^i \qquad (4\text{-}22)$$

$$\overline{R_1^i} = \overline{K^i \cdot C \cdot Q^i} = \overline{K^i} + \overline{C} + \overline{Q^i} \qquad (4\text{-}23)$$

$$\overline{S_2^i} = \overline{W^i \cdot \overline{C}} = \overline{W^i} + C \qquad (4\text{-}24)$$

$$\overline{R_2^i} = \overline{\overline{W^i} \cdot \overline{C}} = W^i + C \qquad (4\text{-}25)$$

Dosazením výrazů *(4-22)* a *(4-23)* do výrazu *(4-20)* dostaneme

$$W^{i+1} = J^i \cdot C \cdot \overline{Q^i} + (\overline{K^i} + \overline{C} + \overline{Q^i}) \cdot W^i \qquad (4\text{-}26)$$

a dosazením výrazů *(4-24)* a *(4-25)* do výrazu *(4-21)* dostaneme

$$Q^{i+1} = W^i \cdot \overline{C} + (W^i + C) \cdot Q^i \qquad (4\text{-}27)$$

Z výrazů *(4-26)* a *(4-27)* je možné sestavit stavovou tabulku členu JK, která je uvedena na obr. 4-35.

Když vycházíme z předpokladu, že $W^i = Q^i =$ log 0, lze z tabulky stavů odvodit funkci členu JK. Za podmínky $J =$ log I a změny hodinového signálu z $L \rightarrow H$ překlopí RS1 a člen přejde do stavu 10 a následně samovolně do stabilního stavu, kdy platí $W^i = 1$ a $Q^i = 0$. Na sestupnou hranu hodinového signálu, změna $H \rightarrow L$ bez závislosti na vstupních proměnných JK překlopí RS2 a člen přejde do stavu 11 a posléze do stabilního stavu $W^i = 1$ a $Q^i = 1$.V tomto diskrétním čase se nový stav objeví na výstupech členu JK. Analogicky se za podmínky $K = 1$ uskutečňuje přechod ze stavu 11 do stavu 00.

$W^{i+1}\,Q^{i+1}$ \ $J^i\,K^i\,C^i$	000	100	110	010	011	111	101	001
$W^i\,Q^i$ 00	00	00	00	00	00	10	10	00
10	11	11	11	11	10	10	10	10
11	11	11	11	11	01	01	11	11
01	00	00	00	00	01	01	01	01

Obr. 4-35: Stavová tabulka členu JK

Rozborem výrazů *(4-26)* a *(4-27)* lze dospět ke stejnému závěru, a to tímto způsobem. Prvním členem výrazu *(4-26)* se uskutečňuje přechod vnitřní proměnné W^{i+1} do stavu log I a druhým členem do stavu log 0. Za podmínky W^i = log 0 a C = log I, RS1 mění stav na W^{i+1} = log I, je možné psát

$$W^{i+1} = J^i \cdot \overline{Q^i} \qquad (4\text{-}28)$$

a pro změnu na opačný stav, RS1 mění stav na W^{i+1} = log 0, platí

$$W^{i+1} = \overline{K}, \qquad (4\text{-}29)$$

protože $W^i = \log 1$, $\overline{Q^i} = \log 0$ a $\overline{C} = \log 0$.

Analogicky ve výrazu *(4-27)* zajišťuje druhý člen přechod RS2 do stavu Q^{i+1} = log 0, protože Q^i = log I a první člen přechod do stavu Q^{i+1} = log I, protože obvod RS2 mění stav na sestupnou hranu synchronizačního impulsu a tedy $\overline{C} = \log 1$. Dosazením výrazů *(4-28)* a *(4-29)* do výrazu *(4-27)* dostaneme

$$Q^{i+1} = J^i \cdot \overline{Q^i} \cdot \overline{C} + (\overline{K^i} + C) \cdot Q^i \qquad (4\text{-}30)$$

a po akceptování časového průběhu synchronizačního impulsu

$$Q^{i+1} = J^i \cdot \overline{Q^i} + \overline{K^i} \cdot Q^i. \qquad (4\text{-}31)$$

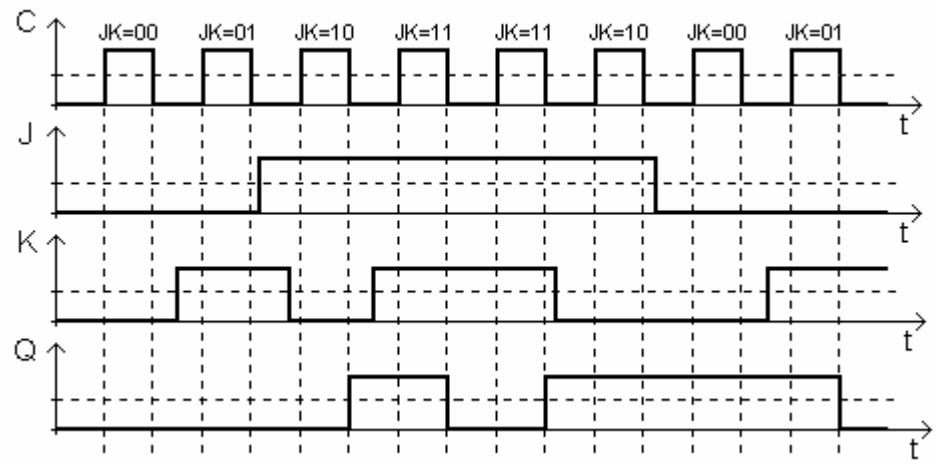
Výraz *(4-31)* představuje booleovskou funkci členu JK, která se využívá k sestavení tabulek na obr. 4-36 a 4-37. Časové průběhy paměťového členu JK jsou na obrázku 4-38.

J	K	Q^{i+1}
0	0	Q^i
0	1	0
1	0	1
1	1	$\overline{Q^i}$

Obr. 4-36: Pravdivostní tabulka paměťového členu JK

$Q^i \rightarrow Q^{i+1}$	J	K
$0 \rightarrow 0$	0	X
$0 \rightarrow 1$	1	X
$1 \rightarrow 0$	X	1
$1 \rightarrow 1$	X	0

Obr. 4-37: Podmínky přechodu mezi jednotlivými stavy



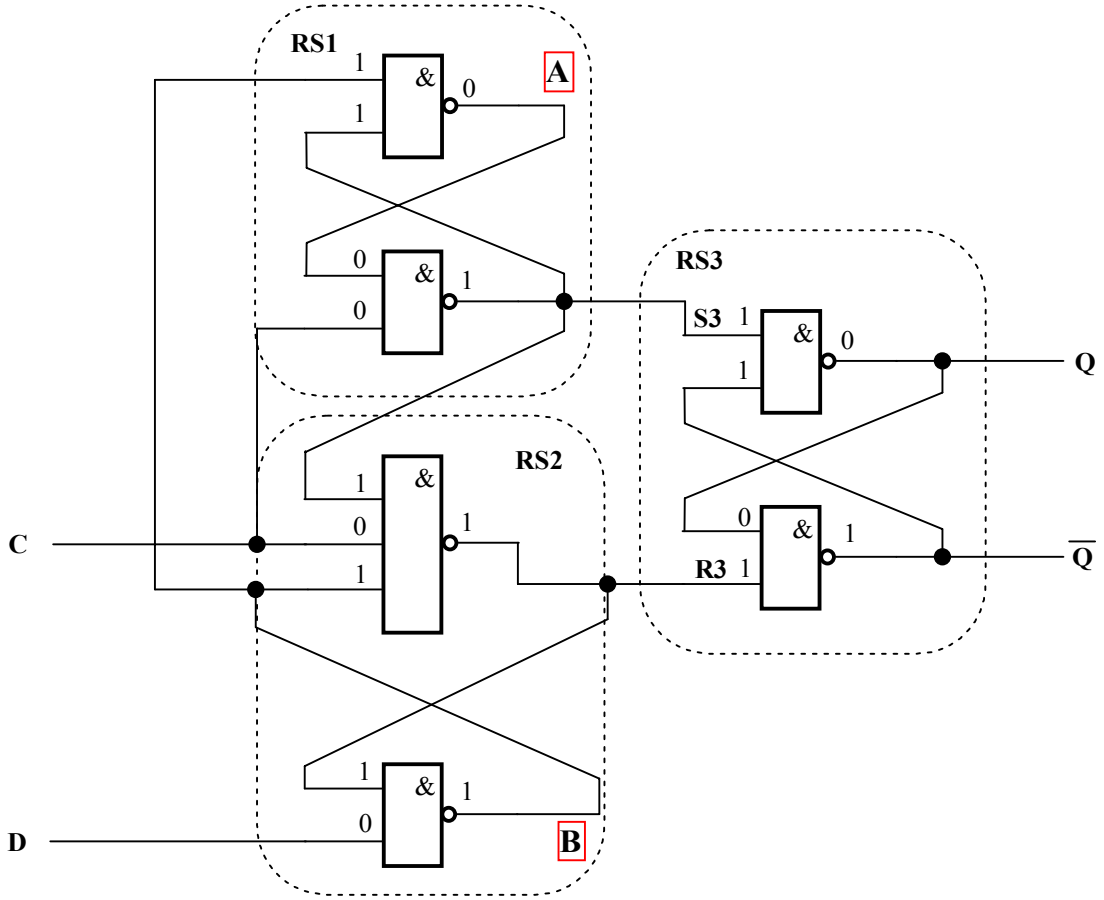
Obr. 4-38: Časové průběhy členu JK

HRANOVĚ ŘÍZENÝ PAMĚŤOVÝ ČLEN D

Jak již sám název napovídá, tyto paměťové členy jsou řízeny hranou hodinového signálu (vzestupnou, resp. sestupnou hranou). U těchto paměťových členů se využívá zpětná vazba, která zabrání průchodu vstupního signálu D strukturou obvodu bezprostředně po vzestupné, resp. sestupné hraně synchronizačního signálu C.

Zapojení paměťového členu D řízeného vzestupnou hranou hodinového signálu je uvedeno na obr. 4-39 zároveň s vyznačením základního stavu.

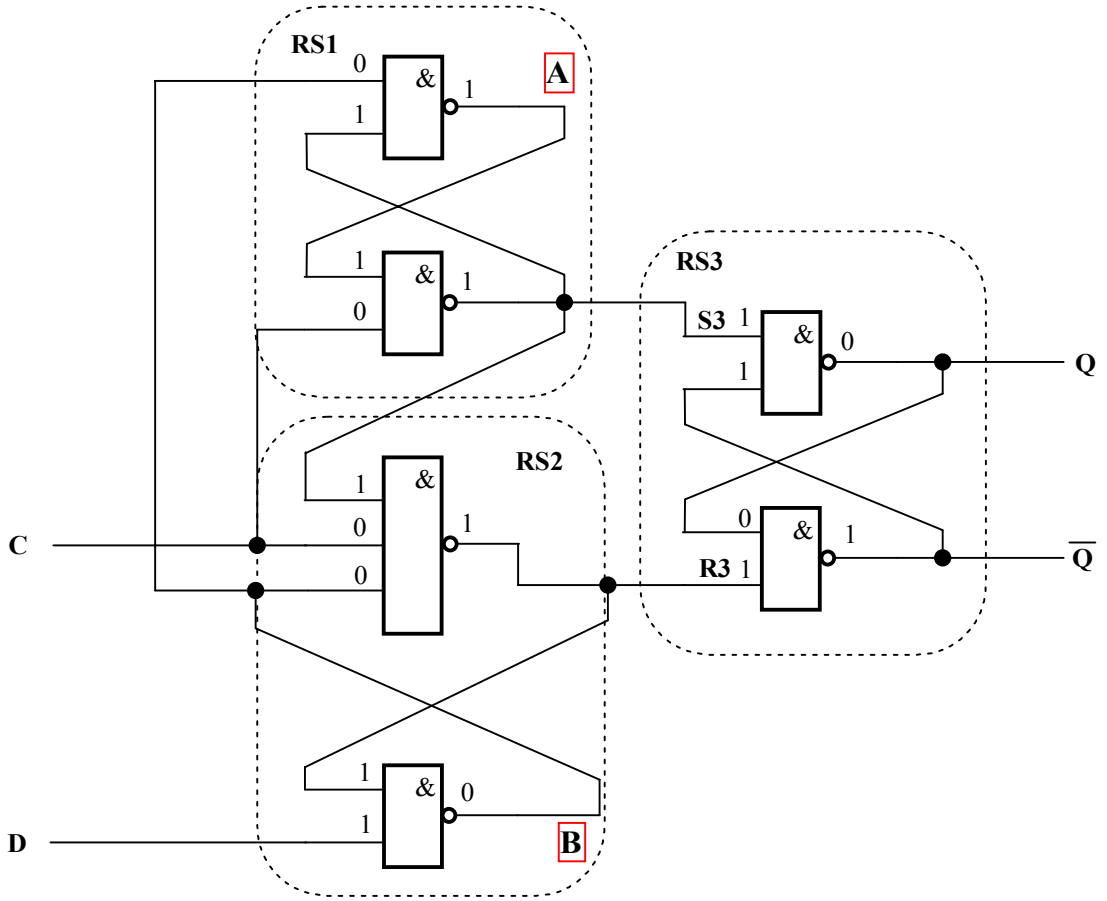
Zapojení je tvořeno třemi paměťovými obvody typu RS sestavených z hradel realizujících funkci NAND.



Obr. 4-39: Zapojení hranově řízeného paměťového členu D s vyznačením základního stavu

Vzhledem k poměrně složitému sestavení výrazů popisujících funkci tohoto paměťového členu, je na následujících obrázcích postupně uveden:

- stav paměťového členu, kdy signál D = log I a C = log 0, obr. 4-40
- stav paměťového členu, kdy signál D = log 0 a C = log I, obr. 4-41
- stav paměťového členu, kdy signál D = log I a C = log I, obr. 4-42

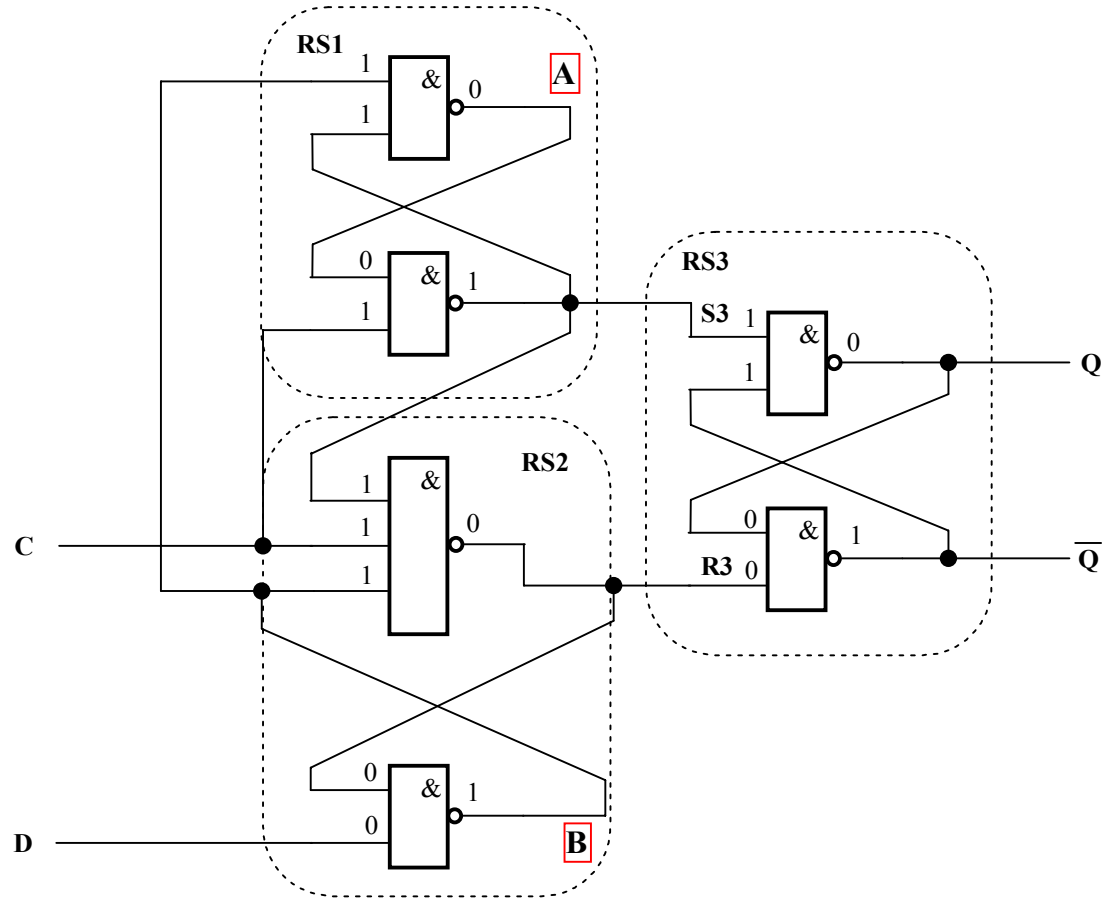


Obr. 4-40: Stav paměťového členu, kdy signál D = log I a C = log 0

ÚKOL K ZAMYŠLENÍ 2

Zaznamenali jste na obr. 4-41, že ke změně signálu *R3* došlo vzestupnou hranou (přechod z log 0 na log I) signálu *C*?

Signál *R3* je nulovací vstup paměťového členu RS3.



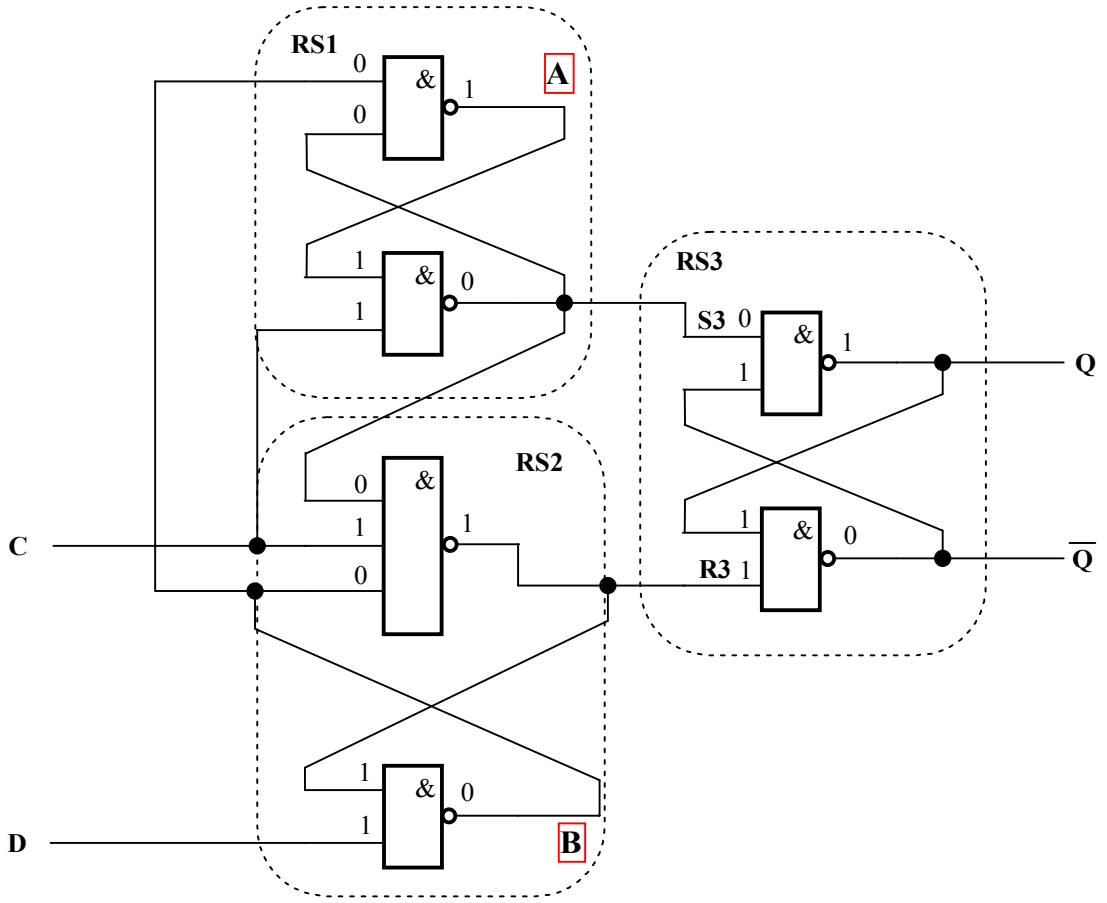
Obr. 4-41: Stav paměťového členu kdy signál *D* = log 0 a *C* = log I

ÚKOL K ZAMYŠLENÍ 3

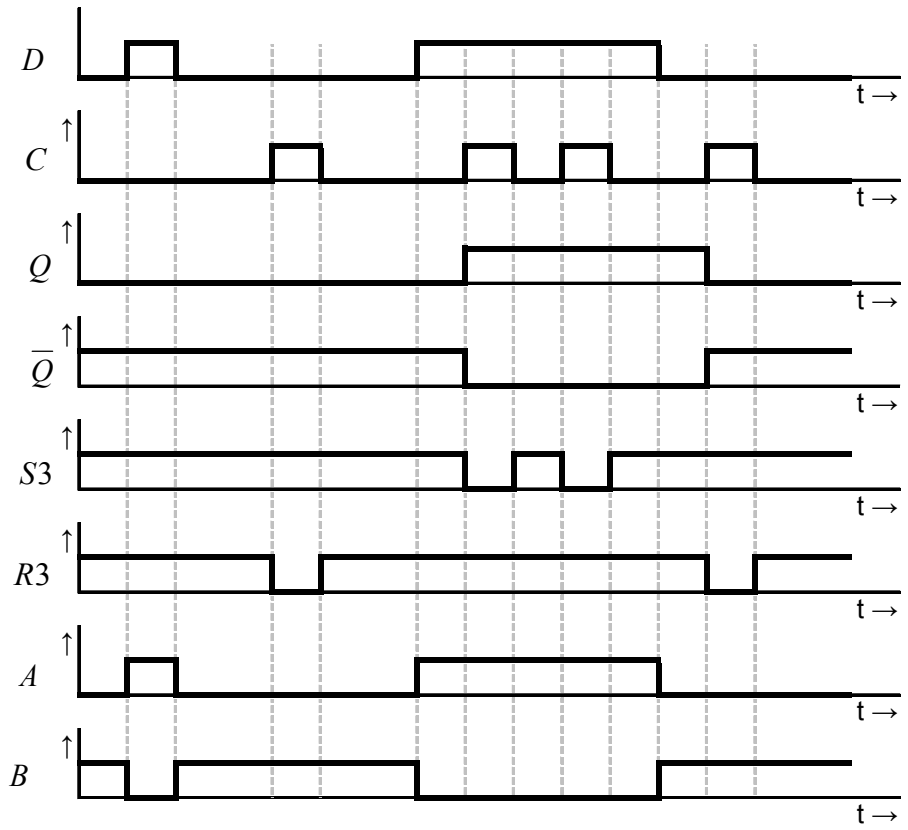
Zaznamenali jste na obr. 4-42, že ke změně signálu *S3* došlo vzestupnou hranou (přechod z log 0 na log I) signálu *C*?

Signál *S3* je nastavovací vstup paměťového členu RS3.

Na závěr je činnost zapojení zobrazena časovým průběhem, viz. obr. 4-43. V těchto časových průbězích neuvažujeme se zpožděním jednotlivých hradel, protože na funkci členu nemají vliv.



Obr. 4-42: Stav paměťového členu kdy signál *D* = log I a *C* = log I



Obr. 4-43: Časový průběh činnosti paměťového členu D řízeného vzestupnou hranou synchronizačního signálu

K ZAPAMATOVÁNÍ 7

Z časových průběhů vyplývá, že ke změně výstupního signálu Q dochází na základě vzestupné hrany synchronizačního impulsu a logická hodnota výstupní proměnné Q je dána logickou hodnotou vstupní proměnné D.

Toto lze vyjádřit následujícím výrazem:

$$Q^{i+1} = D^{i} \qquad (4\text{-}33)$$

INTEGROVANÝ OBVOD MH 7474

Integrovaný obvod MH 7474 je tvořen hranově řízeným D obvodem (obr. 4-39) a asynchronním RS obvodem tvořeným z hradel NAND (obr. 4-12). Funkce tohoto obvodu je dána funkční tabulkou dle obr. 4-44.

Vstupy				Výstupy	
S	R	C	D	Q	$\overline{Q}$
Asynchronní režim					
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1*	1*
Synchronní režim					
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Pamatován stav obvodu	

* nestabilní stav

↑ přechod z log 0 do log I

Obr. 4-44: Pravdivostní tabulka

$Q^i \rightarrow Q^{i+1}$	D
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	0
$1 \rightarrow 1$	1

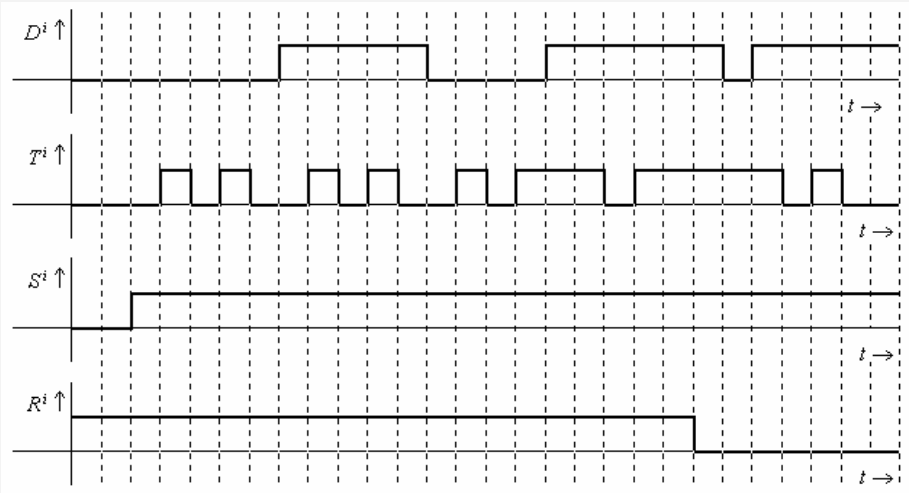
Obr. 4-45: Tabulka přechodů

Z tabulky na obr. 4-44 vidíme, že:

- vstupy *R* a *S* mají vyšší prioritu,
- stav na výstupu je nestabilní v případě, že *R* i *S* jsou aktivní současně,
- výstupní signál *Q* sleduje stav na vstupu D při náběžné hraně hodinového impulsu.

ÚLOHA K ŘEŠENÍ 4-1

Zakreslete do časového diagramu chování paměťového obvodu MH 7474. Jsou zadány časové průběhy vstupních signálů:



INTEGROVANÝ OBVOD MH 74S112

Integrovaný obvod MH 74S112 je tvořen dvojicí obvodů JK vybavených navíc asynchronními vstupy nastavení a nulování. Funkce tohoto obvodu je dána funkční tabulkou dle obr. 4-46.

Vstupy					Výstupy	
S	R	J	K	C	Q^{i+1}	$\overline{Q^{i+1}}$
Asynchronní režim						
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1*	1*
Synchronní režim						
1	1	0	0	↓	Q^i	$\overline{Q^i}$
1	1	1	0	↓	1	0
1	1	0	1	↓	0	1
1	1	1	1	↓	$\overline{Q^i}$	Q^i
1	1	X	X	1	Pamatován stav obvodu	

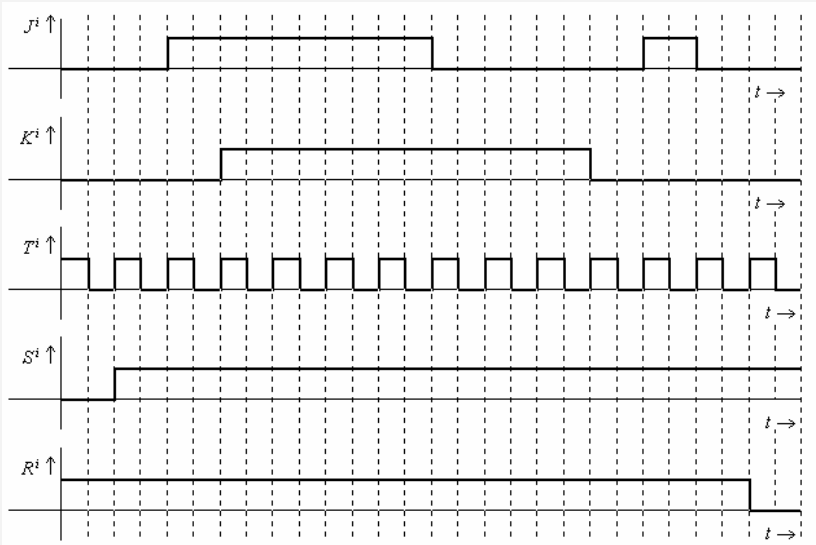
* nestabilní stav

↓ přechod z log I do log 0

Obr. 4-46: Funkční tabulka

ÚLOHA K ŘEŠENÍ 4-2

Zakreslete do časového diagramu chování paměťového obvodu MH74S112. Jsou zadány časové průběhy vstupních signálů:



4.2 Návrh synchronních sekvenčních obvodů

Úvodem je třeba konstatovat, že návrh sekvenčních obvodů je podstatně složitější než návrh kombinačních obvodů. Kapitola je omezena pouze na synchronní obvody, protože tato skupina sekvenčních obvodů eliminuje vznik hazardů a každý asynchronní sekvenční obvod lze převést na obvod synchronní.

Předpokládejme, že máme zadáno chování sekvenčního obvodu tabulkou nebo jinou ekvivalentní formou. Cílem návrhu potom je vytvoření struktury synchronního sekvenčního obvodu, který obvodově realizuje zadané funkční závislosti. V návaznosti na předchozí kapitolu je tedy zapotřebí realizovat zobrazení, které proměnným $\bar{x}^i$ a $\bar{z}^i$ přiřadí výstupní proměnné $\bar{y}^i$ a proměnné $\bar{z}^{i+1}$ následujícího stavu označované jako budící kanál.

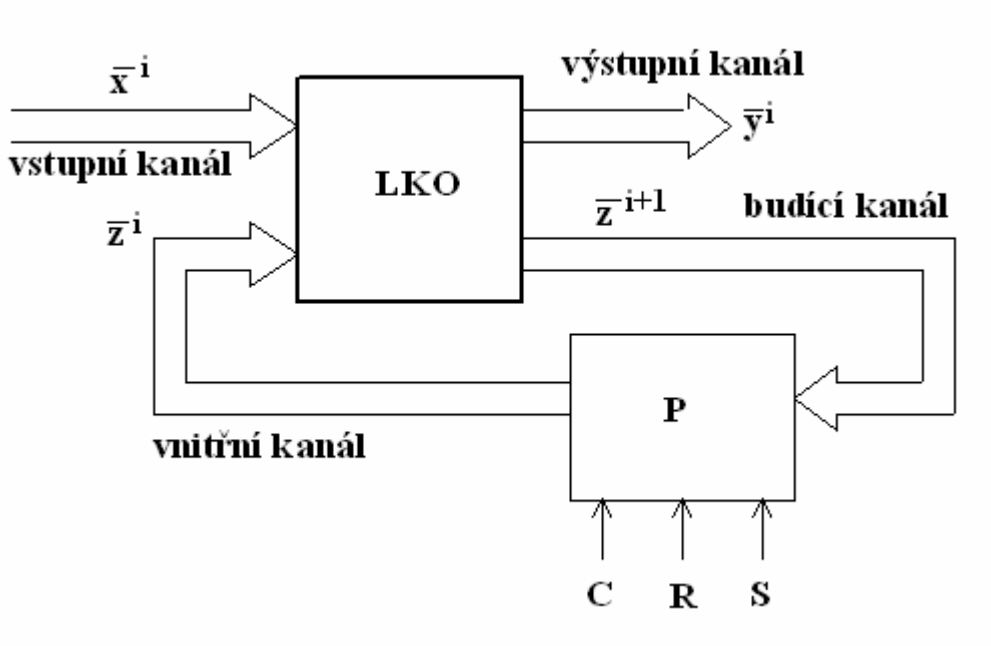
Protože paměťový člen P bývá předem specifikován, resp. zadán, např. člen D, JK, redukuje se celý návrh na blok LKO (logický kombinační obvod), který realizuje zobrazení:

$$\bar{y}^i = f(\bar{x}^i, \bar{z}^i)\ , \qquad \bar{z}^{i+1} = g(\bar{x}^i, \bar{z}^i) \tag{4-34}$$

Proměnné *R* a *S* paměťového členu se využívají k nastavení počátečního stavu $\bar{z}^0$ paměťového členu P. Synchronizační signál *C* definuje jednoznačně diskrétní body, ve kterých se vyhodnocuje nový stav a tedy časový odstup mezi impulsy představuje žádané zpoždění.

Kompletní návrh je možné rozdělit do následujících etap:

- stanovení počtu stavových proměnných,
- zakódování stavů sekvenčního obvodu pomocí stavových proměnných,
- stanovení následujících stavů pro každý stav obvodu a každý vstupní vektor $\bar{x}(t)$,
- stanovení výstupního vektoru $\bar{y}(t)$ pro každý stav obvodu a každý vstupní vektor $\bar{x}(t)$,
- sestavení výstupních booleovských funkcí,
- sestavení budících booleovských funkcí,
- realizace výstupních a budících booleovských funkcí,
- realizace nastavení počátečního stavu sekvenčního obvodu.

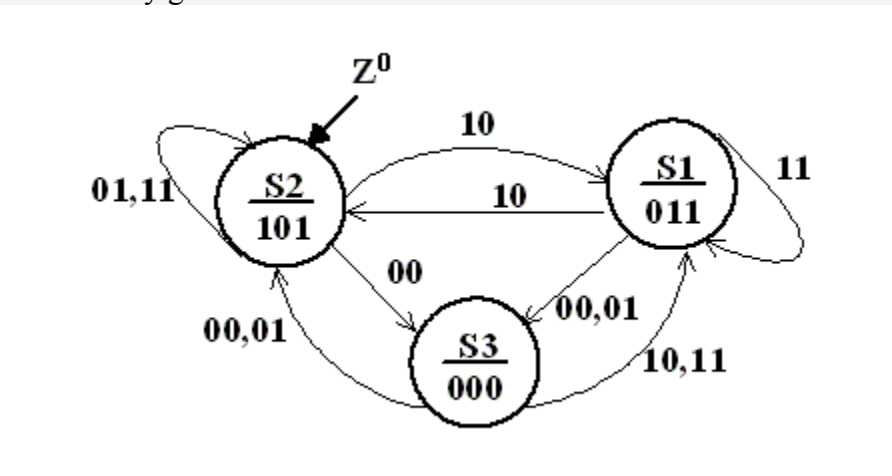


Obr. 4-47: Blokové schéma synchronního sekvenčního obvodu

ŘEŠENÝ PŘÍKLAD

Navrhněte synchronní sekvenční obvod, jehož činnost je dána orientovaným grafem. Výstupní proměnné jsou vázány na stav obvodu, výchozí stav obvodu je stav S2.

Orientovaný graf sekvenčního obvodu:



Řešení příkladu

Ze zadání vyplývá, že se jedná o sekvenční obvod, který má 2 vstupní, 3 výstupní proměnné a 3 stavy.

Stanovení počtu stavových proměnných. K jednoznačnému popisu stavů takovéhoto obvodu je nutný počet stavových proměnných *k*, který je dán zobrazením

$$S \rightarrow (0,1)^k = 2^k ,$$

kde S je počet stavů. Pro jednoznačné zobrazení potom musí platit

$$S \leq 2^k .$$

Odtud

$$k \geq \log_2 S .$$

V našem případě je k=1,585. Počet stavových proměnných proto volíme 2 a označíme je z_0 a z_1. Je zřejmé, že k tomu abychom zakódovali 3 stavy, je zapotřebí 2 proměnných.

Zakódování stavů sekvenčního obvodu spočívá v přiřazení logických hodnot stavových proměnných jednotlivým stavům, přičemž přiřazení musí být jednoznačné. Standardně se používá způsob zobrazení pomocí Karnaughovy mapy, resp. tabulky:

Zakódování stavů obvodu

z_0 \ z_1		
		S1
	S3	S2

S	z_1	z_0
1	1	0
2	1	1
3	0	1

Je třeba poznamenat, že zakódování je zcela libovolné, ale pro přehlednost se často používá kódování binární. Redundantnost zakódování definuje nepoužité kombinace jako volné a stavovým proměnným proto přiřazuje logickou hodnotu X.

Stanovení následujících stavů obvodu vychází z orientovaného grafu. Cílem je přehledně uspořádat všechny podmínky, za kterých sekvenční obvod mění svůj stav včetně uvedení stavu, do kterého přechází, a to nejčastěji formou tzv. tabulky přechodů:

Tabulka přechodů

stav			vstupní proměnné x_1x_0			
S	z_1	z_0	00	01	10	11
1	1	0	01	01	11	10
2	1	1	01	11	10	11
3	0	1	11	11	10	10
	0	0	XX	XX	XX	XX

Je zapotřebí si uvědomit, že tabulka přechodů zahrnuje informace o stavu obvodu ve dvou sousedních diskrétních časových bodech. V levé silně orámované části tabulky přechodů jsou informace v čase i a v pravé silně orámované části v čase $i + 1$.

Stav obvodu je vzhledem k dalším úkonům zapisován pomocí stavových proměnných z_1 a z_0. Například, když je obvod ve stavu S = 2, pro stavové proměnné platí z_1 = log I a z_0 = log I. Když bude vektor vstupních proměnných x_1 = log I a x_0 = log 0, obvod přejde do následujícího stavu S = 1, který je kódován pomocí stavových proměnných Z_1 = log I a Z_0 = log 0. Tento kód je zapsán v pravé silně vyznačené části tabulky jako následující stav v diskrétním bodě $i + 1$. Tímto způsobem je vyplněna celá tabulka s výjimkou nedefinovaného stavu. Pro tento stav budou stavové proměnné nedefinovány.

Stanovení výstupního vektoru vychází z orientovaného grafu a v podstatě představuje sestavení tabulky potřebné pro výpis výstupních booleovských funkcí. Vzhledem k tomu, že výstupní funkce je vyhodnocována vždy v jednom diskrétním bodě, tabulka obsahuje informace pouze pro tento bod.

Tabulka výstupních funkcí

stav			vstupní proměnné x_1x_0			
S	z_1	z_0	00	01	10	11
1	1	0	011	011	011	011
2	1	1	101	101	101	101
3	0	1	000	000	000	000
	0	0	XXX	XXX	XXX	XXX

Sestavení výstupních booleovských funkcí se provádí pro každou výstupní proměnnou samostatně a způsobem shodným s návrhem v kombinačních obvodech. Zvolíme metodu zjednodušování, např. pomocí Karnaughových map. Sestavíme mapy a provedeme výpis funkce v minimálním součtovém tvaru:

Mapy pro výstupní funkce

y_2^i

z_1^i \ z_0^i \ x_1^i x_0^i	X	X	X	X
	0	0	0	0
	1	1	1	1
	0	0	0	0

y_1^i

z_1^i \ z_0^i \ x_1^i x_0^i	X	X	X	X
	0	0	0	0
	0	0	0	0
	1	1	1	1

y_0^i

z_1^i \ z_0^i \ x_1^i x_0^i	X	X	X	X
	0	0	0	0
	1	1	1	1
	1	1	1	1

Výstupní funkce v minimálním součtovém tvaru:

$$y_2 = z_0 \cdot z_1$$

$$y_1 = \overline{z}_0$$

$$y_0 = z_1$$

Sestavení budících funkcí vychází z tabulky přechodů. Funkce opět zapíšeme do Karnaughových map:

Mapy pro budící funkce

$Z_1 = z_1^{i+1}$

z_1^i \ z_0^i \ x_1^i x_0^i	X	X	X	X
	1	1	1	1
	0	1	1	1
	0	0	1	1

$Z_0 = z_0^{i+1}$

z_1^i \ z_0^i \ x_1^i x_0^i	X	X	X	X
	1	1	0	0
	1	1	1	0
	1	1	0	1

Další postup se liší druhem zpožďovacího členu. V praxi se používají nejčastěji dva typy zpožďovacího členu, a to buď paměťový člen typu D (MH 7474) nebo paměťový člen JK (MH 7472 nebo MH 74112).

Sestavení budících funkcí pro paměťový člen typu D. Protože paměťový člen D má pouze jednu vstupní proměnnou, lze budící funkce vypsat přímo z výše uvedených map budících funkcí. Když uvážíme, že kolik je budících funkcí, tolik je zpožďovacích členů, můžeme obecně pro zapojení *k* těchto členů psát

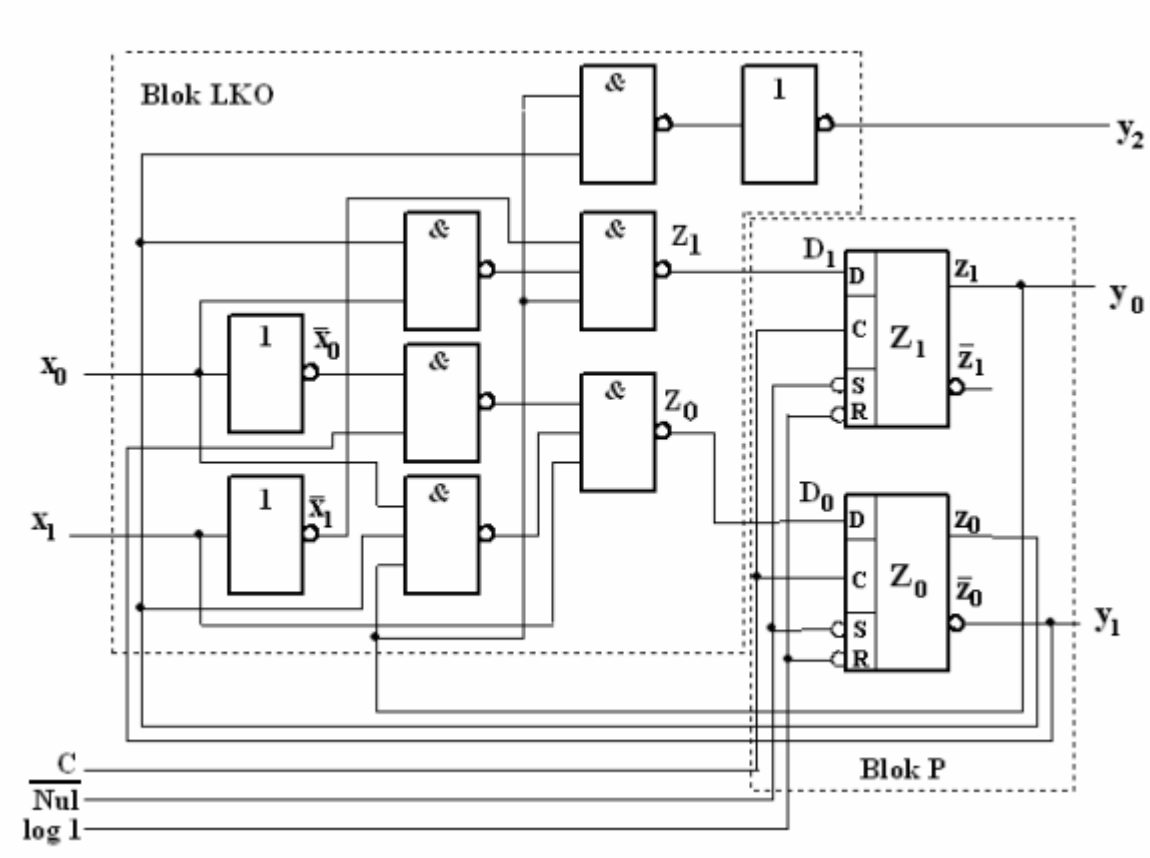
$$D_k = z_k^{i+1} = Z_k$$

V součtovém tvaru mají jednotlivé funkce tvar

$$D_1 = Z_1 = \bar{z}_1 + x_1 + x_0 \cdot z_0$$
$$D_0 = Z_0 = \bar{x}_1 + \bar{x}_0\bar{z}_0 + x_0 z_0 z_1$$

Realizace nastavení počátečního stavu se zajišťuje asynchronním členem RS. Jestliže je zadán jako výchozí stav obvodu stav S2, pro který platí $z_1^i = z_0^i = \log 1$, je nutné na vstupy hradla *S* připojit nulovací signál, kterým se paměťové členy nastaví do požadovaného stavu. Vstupy *R* je nutné ošetřit, v tomto případě připojením na log I.

Síť s paměťovými členy typu D (typ hradla MH 7474):



Sestavení budících funkcí pro paměťový člen typu JK. Paměťový člen JK má 2 vstupní proměnné, a proto je zapotřebí mapy budících funkcí přizpůsobit počtu vstupů. Přizpůsobení se uskuteční pomocí tabulky přechodů mezi stavy členu JK viz. obr. 4-37. Takto upravené mapy, označeny jako mapy budících funkcí pro paměťový člen JK, jsou uvedeny níže. Při jejich vytváření je třeba důsledně dodržovat oddělení diskrétních časových bodů, tj. současný stav budící proměnné a její následující stav. Jestliže tyto mapy rozepíšeme pro jednotlivé vstupní proměnné paměťového členu JK, dostaneme Karnaughovy mapy pro budící proměnnou Z_1 a Z_0.

Mapy budících funkcí pro paměťový člen JK

$J_1, K_1 = Z_1$

z_1^i \ z_0^i \ x_1^i x_0^i				
	X	X	X	X
	1X	1X	1X	1X
	1X	X0	X0	X0
	X1	X1	X0	X0

$J_0, K_0 = Z_0$

z_1^i \ z_0^i \ x_1^i x_0^i				
	X	X	X	X
	X0	X0	X1	X1
	X0	X0	X0	X1
	1X	1X	0X	1X

Mapy pro budící proměnnou Z_1

J_1

z_1^i \ z_0^i \ x_1^i x_0^i				
	X	X	X	X
	1	1	1	1
	1	X	X	X
	X	X	X	X

K_1

z_1^i \ z_0^i \ x_1^i x_0^i				
	X	X	X	X
	X	X	X	X
	X	0	0	0
	1	1	0	0

Mapy pro budící proměnnou Z_0

J_0

z_1^i \ z_0^i \ x_1^i x_0^i				
	X	X	X	X
	X	X	X	X
	X	X	X	X
	1	1	0	1

K_0

z_1^i \ z_0^i \ x_1^i x_0^i				
	X	X	X	X
	0	0	1	1
	0	0	0	1
	X	X	X	X

Výpisem z map dostaneme hledané funkce:

$$J_1 = 1$$

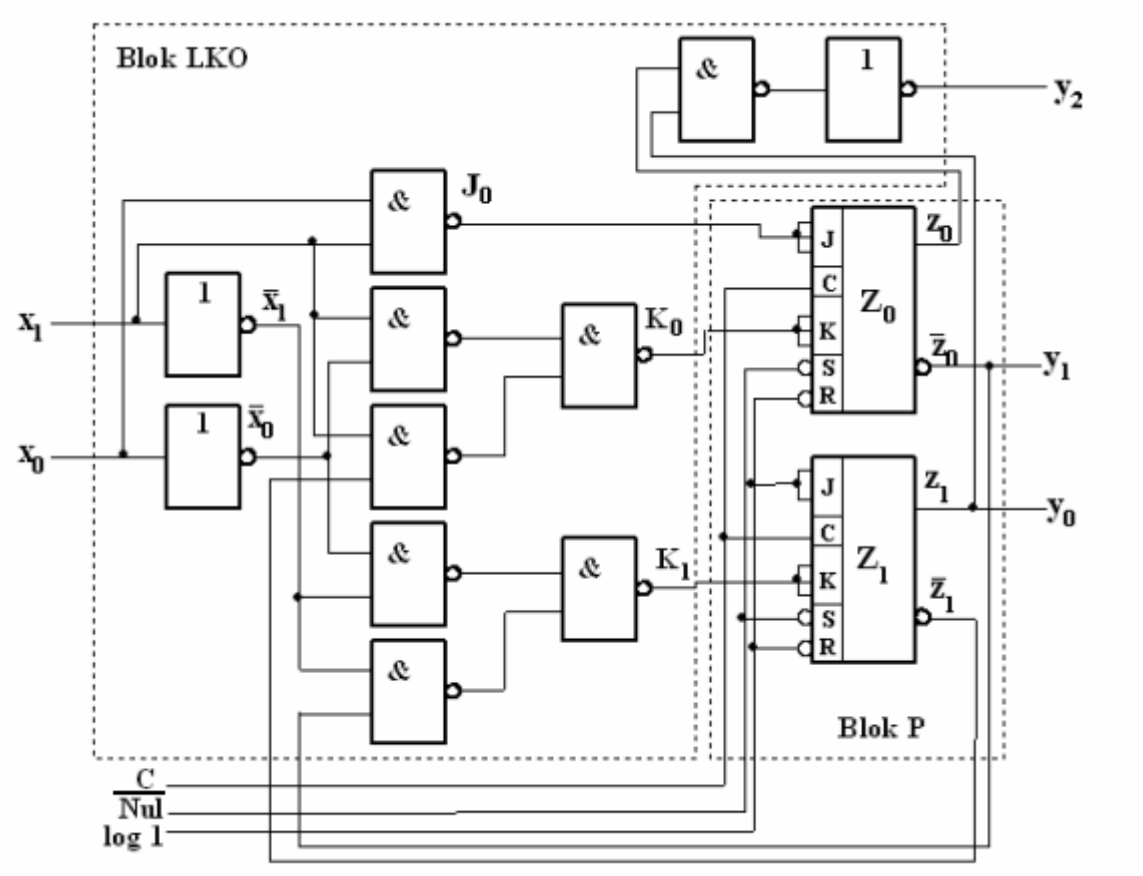
$$K_1 = \bar{x}_0\bar{x}_1 + \bar{x}_1\bar{z}_0$$

$$J_0 = \bar{x}_1 + \bar{x}_0$$

$$K_0 = x_1\bar{z}_1 + \bar{x}_0 x_1$$

Realizace nastavení počátečního stavu se zajišťuje asynchronním členem RS stejným způsobem jako u paměťového členu D.

Síť s paměťovými členy typu JK (typ hradla MH 7472):



Vzhledem k tomu, že u integrovaných obvodů typu MH 7472 je možné přímo na vstupech JK realizovat součin 3 proměnných, je výhodné sestavit z map funkce v součinovém tvaru:

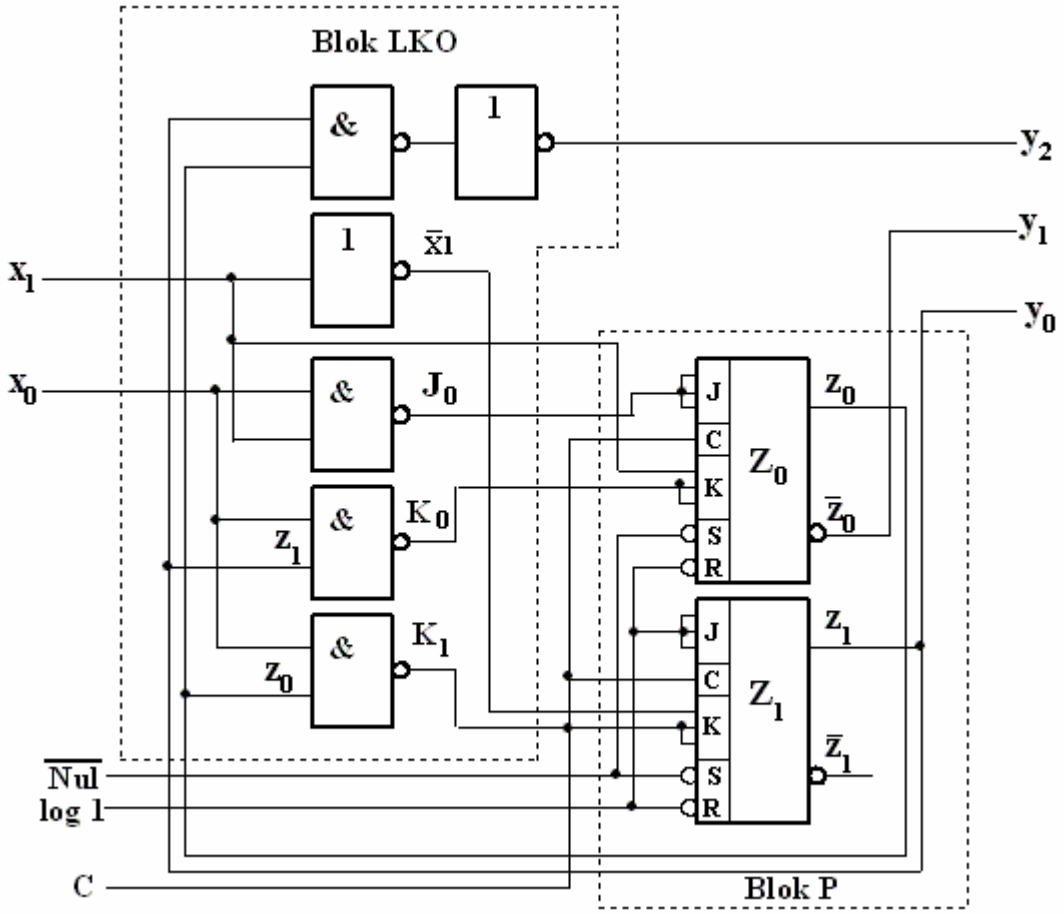
$$J_1 = 1$$

$$K_1 = x_1 \cdot (\bar{x}_0 + \bar{z}_0)$$

$$J_0 = \bar{x}_0 + \bar{x}_1$$

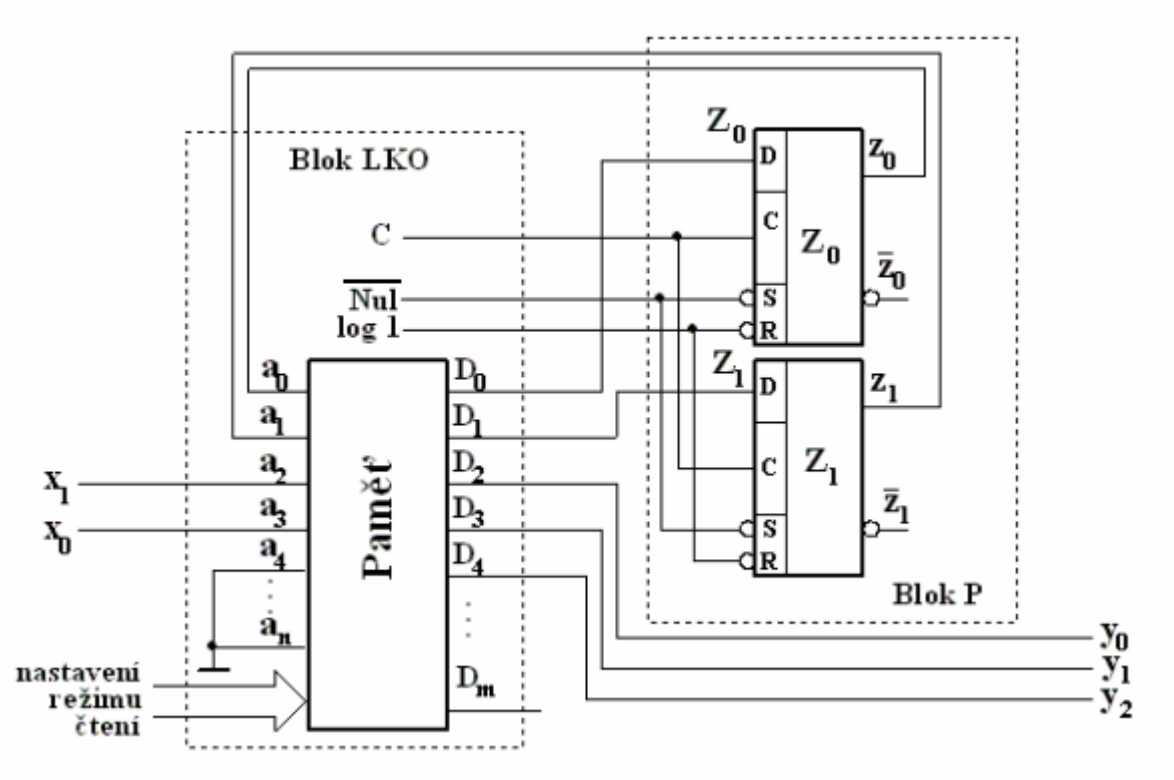
$$K_0 = \bar{x}_1 \cdot (\bar{x}_0 + \bar{z}_1)$$

Síť s paměťovými členy typu JK (typ hradla MH 7472):



Jak již bylo uvedeno, představuje blok LKO logický kombinační obvod, k jehož realizaci může být použito libovolného způsobu, včetně paměťového prvku. Způsob zapojení paměťového prvku je dán přiřazením vstupních a výstupních proměnných. Na tomto přiřazení potom závisí i obsah jednotlivých adres.

Využití paměťového prvku v sekvenčním obvodu:



Pro přiřazení podle předchozího obrázku, bude obsah paměti dán následující pravdivostní tabulkou:

vstupní proměnné				výstupní proměnné				
a_3	a_2	a_1	a_0	D_4	D_3	D_2	D_1	D_0
x_1	x_0	z_1	z_0	y_2	y_1	y_0	Z_1	Z_0
0	0	0	0	X	X	X	X	X
0	0	0	1	0	0	0	1	1
0	0	1	0	0	1	1	0	1
0	0	1	1	1	0	1	0	1
0	1	0	0	X	X	X	X	X
0	1	0	1	0	0	0	1	1
0	1	1	0	0	1	1	0	1
0	1	1	1	1	0	1	1	1
1	0	0	0	X	X	X	X	X
1	0	0	1	0	0	0	1	0
1	0	1	0	0	1	1	1	1
1	0	1	1	1	0	1	1	0
1	1	0	0	X	X	X	X	X
1	1	0	1	0	0	0	1	0
1	1	1	0	0	1	1	1	0
1	1	1	1	1	0	1	1	1

*

ŘEŠENÝ PŘÍKLAD

Navrhněte synchronní čítač modulo 6, krok 5, jehož počáteční stav je 3. Nakreslete schéma zapojení pomocí klopných obvodů typu D.

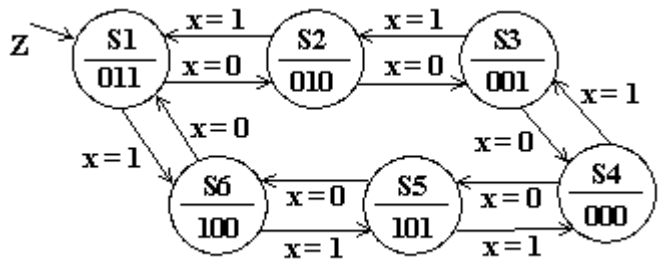
Řešení příkladu

Diagram stavů: počet S = 6,

Počet stavových proměnných: $S \leq 2^k \Rightarrow k = 3$

Zakódování stavů: S1 = 3, S2 = 2, S3 = 1, S4 = 0, S5 = 5, S6 = 4.

Stavový diagram:



Sestavení tabulky přechodů pro paměťový člen D:

stav				x=0			x=1			výstupní		
S	z_2	z_1	z_0	Z_2	Z_1	Z_0	Z_2	Z_1	Z_0	y_2	y_1	y_0
1	0	1	1	0	1	0	1	0	0	0	1	1
2	0	1	0	0	0	1	0	1	1	0	1	0
3	0	0	1	0	0	0	0	1	0	0	0	1
4	0	0	0	1	0	1	0	0	1	0	0	0
5	1	0	1	1	0	0	0	0	0	1	0	1
6	1	0	0	0	1	1	1	0	1	1	0	0

Je zřejmé, že tabulka stavů je shodná s tabulkou výstupů, tzn. že $\bar{z} = \bar{y}$.

Sestavení budících funkcí pro paměťové členy D:

Mapy budících funkcí pro paměťové členy D

Z_2

$x \backslash z_2 \backslash z_1 z_0$	00	01	11	10
00	1	0	0	0
01	0	X	X	1
11	1	X	X	0
10	0	0	1	0

Z_1

$x \backslash z_2 \backslash z_1 z_0$	00	01	11	10
00	0	0	1	0
01	1	X	X	0
11	0	X	X	0
10	0	1	0	1

Z_0

$x \backslash z_2 \backslash z_1 z_0$	00	01	11	10
00	1	1	0	0
01	1	X	X	0
11	1	X	X	0
10	1	1	0	0

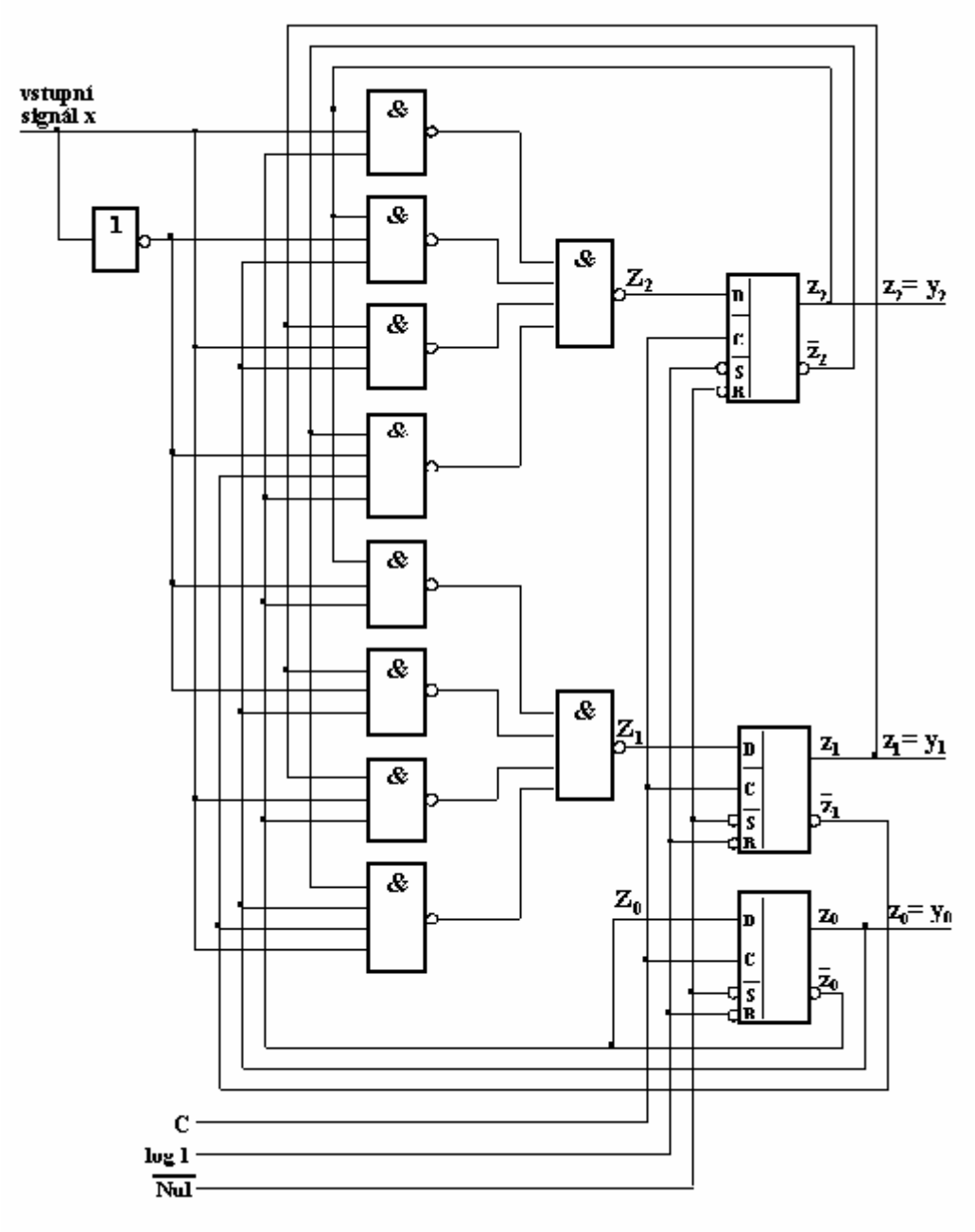
Budící funkce

$$Z_2 = z_2\bar{z}_0x + z_2z_0\bar{x} + z_1z_0x + \bar{z}_2\bar{z}_1\bar{z}_0\bar{x}$$

$$Z_1 = z_2\bar{z}_0\bar{x} + z_1z_0\bar{x} + z_1\bar{z}_0x + \bar{z}_2\bar{z}_1z_0x$$

$$Z_0 = \bar{z}_0$$

Schéma zapojení pomocí D paměťových členů:

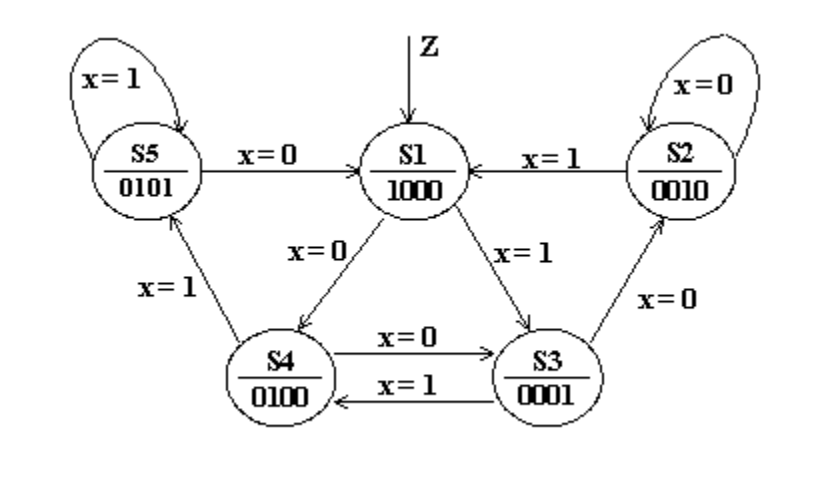


*

ŘEŠENÝ PŘÍKLAD

Navrhněte a realizujte synchronní sekvenční obvod pomocí paměťových členů JK podle uvedeného diagramu stavů. Schéma zapojení navrhněte pomocí obvodů MH 7472.

Stavový diagram:



Řešení příkladu

Počet stavů: S = 5,

Počet stavových proměnných: $S \leq 2^k \Rightarrow k = 3$

Zakódování stavů: S1 = 0, S2 = 2, S3 = 1, S4 = 4, S5 = 5.

Sestavení tabulky přechodů budících funkcí pro návrh s paměťovými členy MH 7472:

stav				$x = 0$			$x = 1$			$x = 0$						$x = 1$						výstupní proměnné			
S	z_2	z_1	z_0	Z_2	Z_1	Z_0	Z_2	Z_1	Z_0	J_2	K_2	J_1	K_1	J_0	K_0	J_2	K_2	J_1	K_1	J_0	K_0	y_3	y_2	y_1	y_0
1	0	0	0	1	0	0	0	0	1	1	X	0	X	0	X	0	X	0	X	1	X	1	0	0	0
2	0	1	0	0	1	0	0	0	0	0	X	X	0	0	X	0	X	X	1	0	X	0	0	1	0
3	0	0	1	0	1	0	1	0	0	0	X	1	X	X	1	1	X	0	X	X	1	0	0	0	1
4	1	0	0	0	0	1	1	0	1	X	1	0	X	1	X	X	0	0	X	1	X	0	1	0	0
5	1	0	1	0	0	0	1	0	1	X	1	0	X	X	1	X	0	0	X	X	0	0	1	0	1

Mapy budících funkcí pro návrh s paměťovými členy MH 7472

J_2

x \ z_0 \ $z_2 z_1$	00	01	11	10
0	1	0	X	X
1	0	X	X	X
0	1	X	X	X
1	0	0	X	X

K_2

x \ z_0 \ $z_2 z_1$	00	01	11	10
0	X	X	X	1
1	X	X	X	1
0	X	X	X	0
1	X	0	X	0

J_1

x \ z_0 \ $z_2 z_1$	00	01	11	10
0	0	X	X	0
1	1	X	X	0
0	0	X	X	0
1	0	X	X	0

K_1

x \ z_0 \ $z_2 z_1$	00	01	11	10
0	X	0	X	X
1	X	X	X	X
0	X	X	X	X
1	X	1	X	X

J_0

x \ z_0 \ $z_2 z_1$	00	01	11	10
0	0	0	X	1
1	X	X	X	X
0	X	X	X	X
1	1	0	X	1

K_0

x \ z_0 \ $z_2 z_1$	00	01	11	10
0	X	X	X	X
1	1	X	X	1
0	1	X	X	0
1	X	X	X	X

Budící funkce

$$J_2 = \bar{z}_1 \cdot (\bar{z}_0 + x) \cdot (z_0 + \bar{x})$$

$$K_2 = \bar{x}$$

$$J_1 = \bar{z}_2 \cdot \bar{x} \cdot z_0$$

$$K_1 = x$$

$$J_0 = \bar{z}_1 \cdot (z_2 + x)$$

$$K_0 = \bar{z}_2 + \bar{x}$$

Mapy výstupních funkcí

y_3

$z_0 \backslash z_2 z_1$	00	01	11	10
0	1	0	X	0
1	0	X	X	0

y_2

$z_0 \backslash z_2 z_1$	00	01	11	10
0	0	0	X	1
1	0	X	X	1

y_1

$z_0 \backslash z_2 z_1$	00	01	11	10
0	0	1	X	0
1	0	X	X	0

y_0

$z_0 \backslash z_2 z_1$	00	01	11	10
0	0	0	X	0
1	1	X	X	1

Výstupní funkce

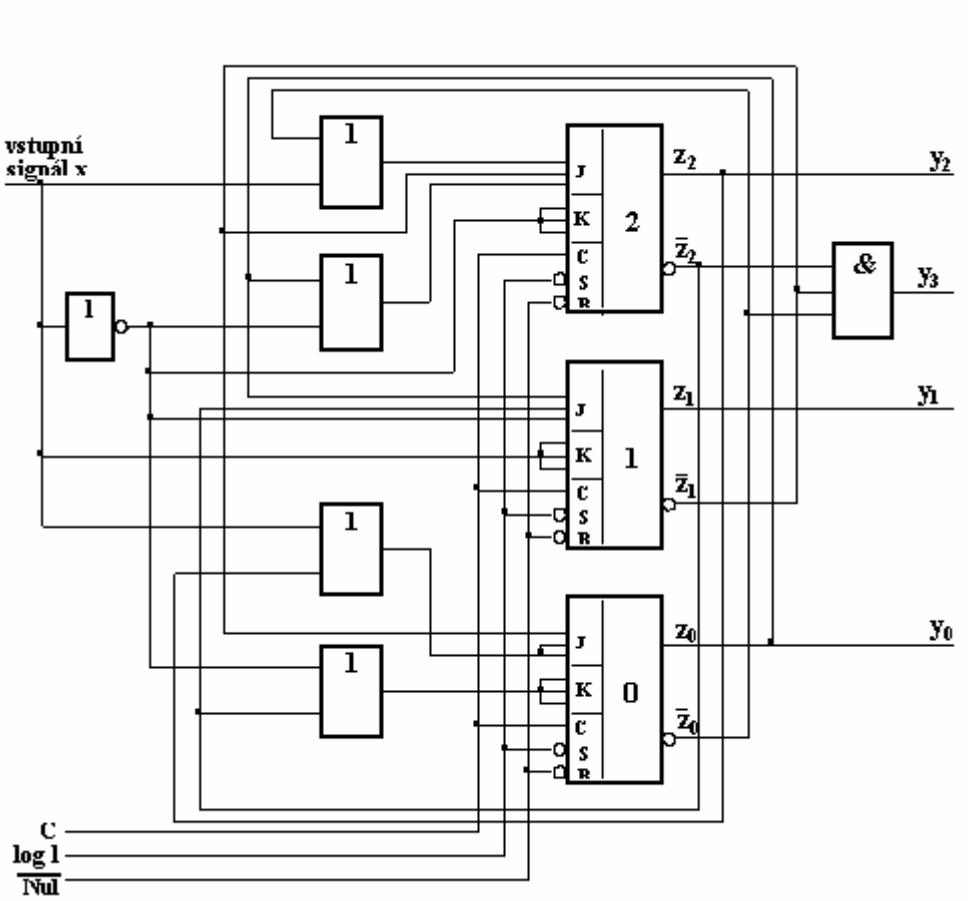
$y_3 = \bar{z}_2 \bar{z}_1 \bar{z}_0$

$y_2 = z_2$

$y_1 = z_1$

$y_0 = z_0$

Schéma zapojení pomocí paměťových členů MH 7472:



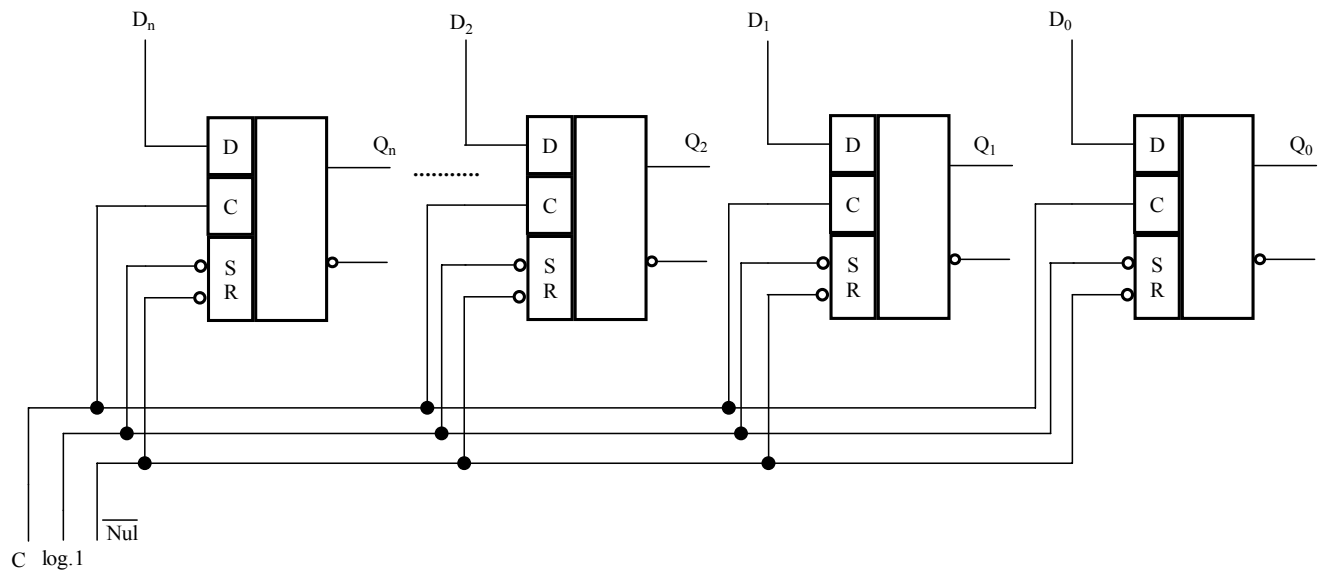
*

4.3 Standardní zapojení logických obvodů

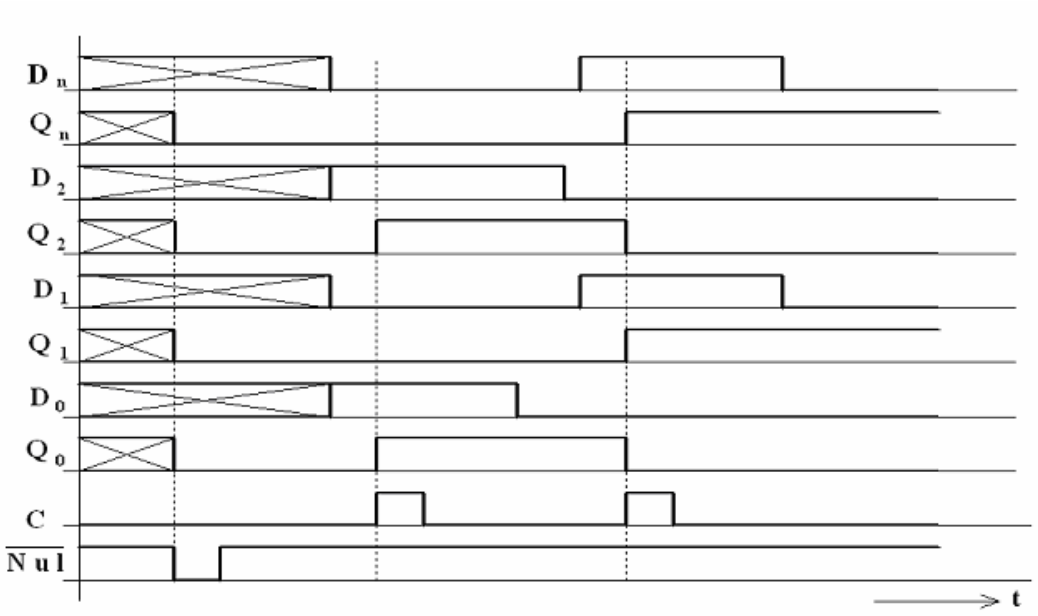
K ZAPAMATOVÁNÍ 8

Uspořádání paměťových členů, které umožňují uchovávat větší počet binárních informací, se nazývá **registr**, viz. obr. 4-48.

Základní stav registru je takový, kdy ve všech paměťových členech je zapsána informace log 0. Toho dosáhneme impulsem na vstupech *R* paměťových členů. Následnými impulsy synchronizačního signálu *C* je zapisována paralelně logická hodnota vstupních proměnných D_0, D_1, D_2,..., D_n do jednotlivých paměťových členů. Časové průběhy registru jsou uvedeny na obr. 4-49.



Obr. 4-48: Zapojení *n*-bitového registru

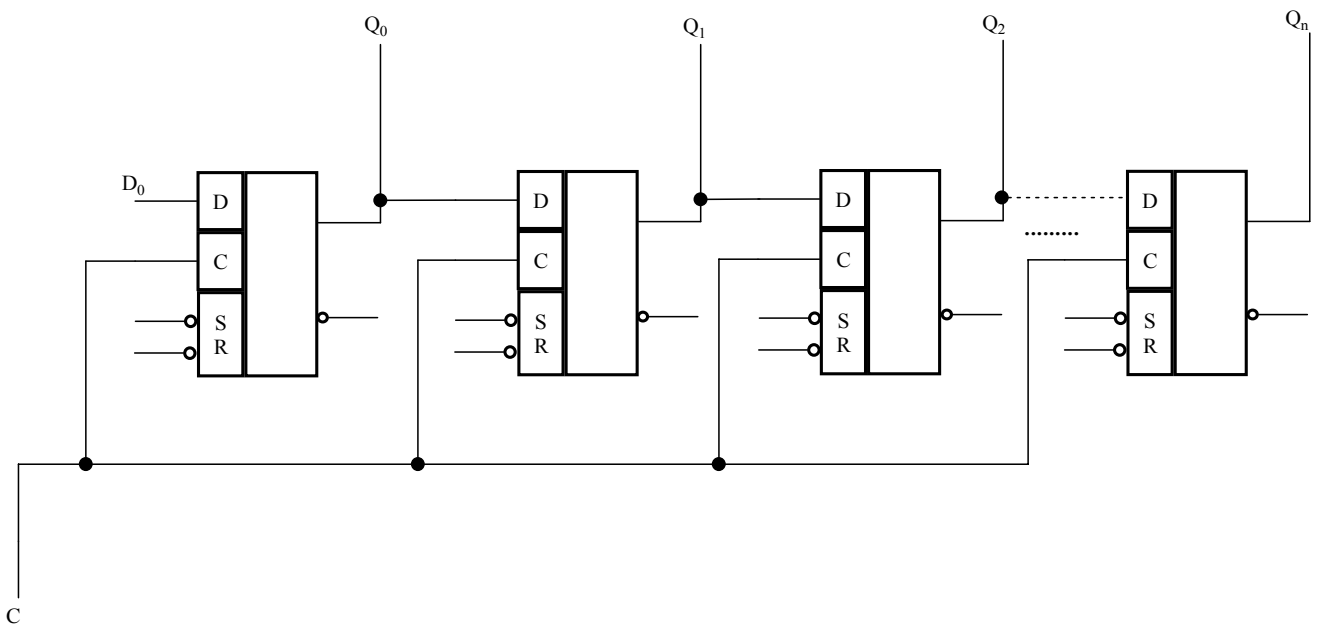


Obr. 4-49: Časové průběhy registru

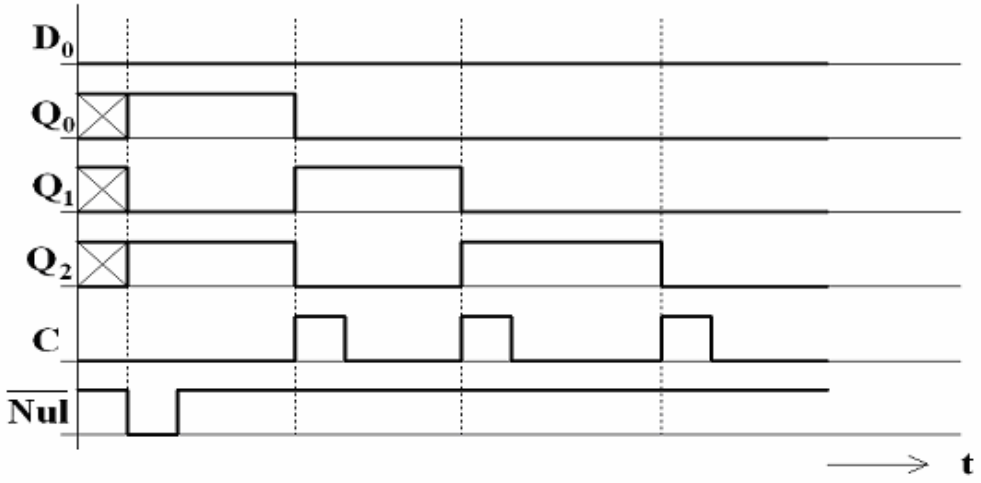
K ZAPAMATOVÁNÍ 9

Zapojení, které zapsanou informaci posouvá v žádaném směru, se nazývá **posuvný registr.**

Zapojení posuvného registru **s posunem vpravo** je na obr. 4-50. Výchozí stav posuvného registru je dán paralelním zápisem informace do paměťových prvků prostřednictvím vstupů *R* a *S*. Má-li být zapsána do daného paměťového členu hodnota log 0, musí být přiveden nulovací signál na vstup *R*, tak jak vidíme ve schématu. Pokud je požadován zápis log I, musel by být přiveden nulovací signál na vstup *S*. Příchodem synchronizačního impulsu se logická hodnota vstupní proměnné posouvá vpravo, směrem k výstupu. Časové průběhy posuvného registru jsou uvedeny na obr. 4-51.



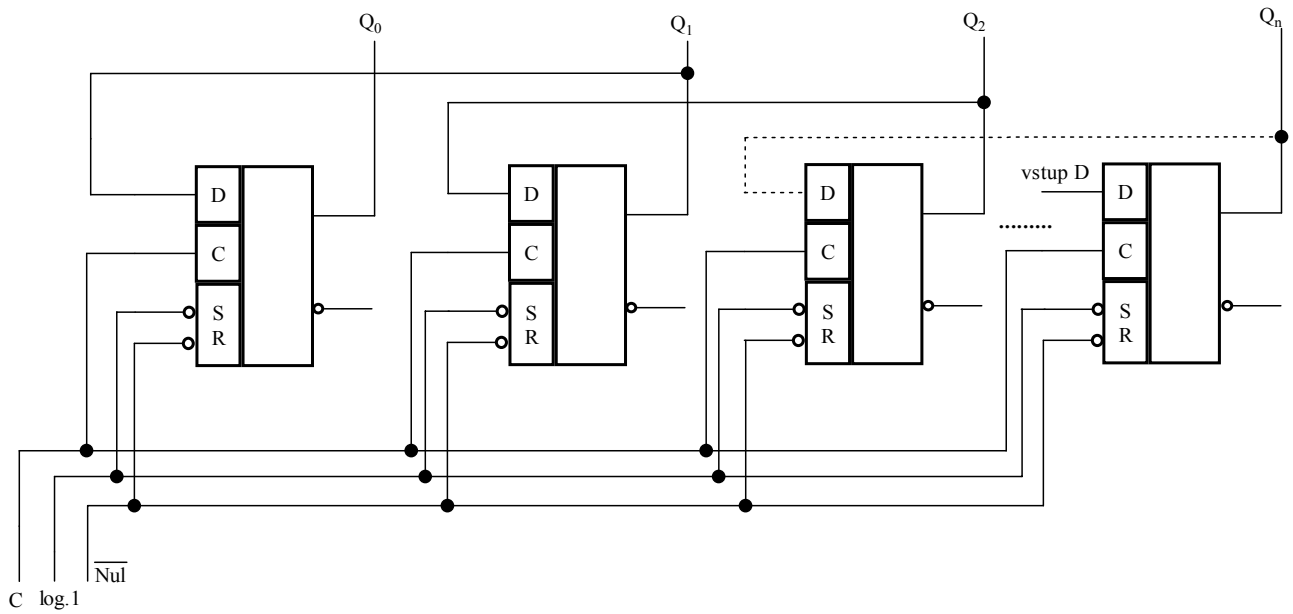
Obr. 4-50: Zapojení posuvného registru s posunem vpravo



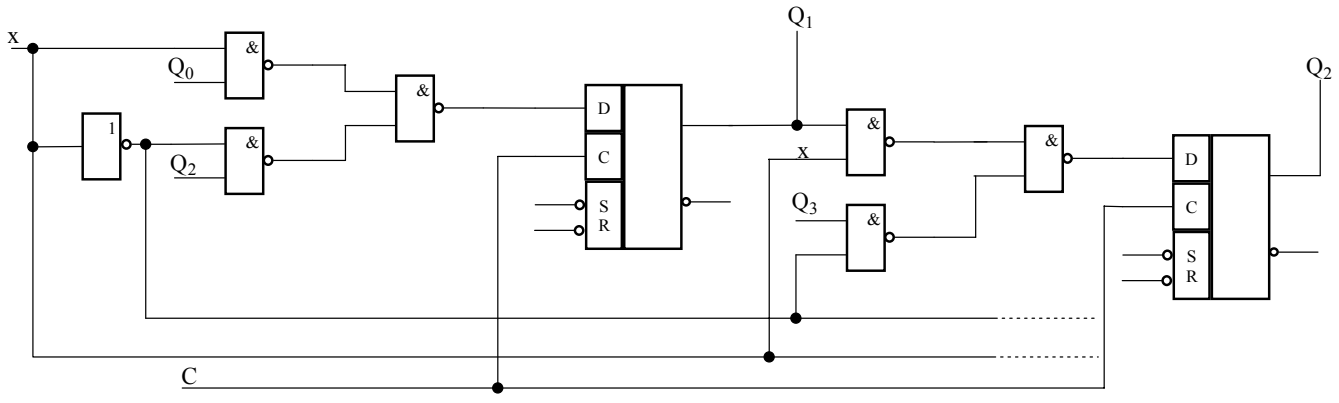
Obr. 4-51: Časové průběhy posuvného registru

Analogická je činnost posuvného registru **směrem vlevo** a **obousměrného** posuvného registru. Směr posunu je u obousměrného registru volitelný vstupním signálem *x:*
pro x = log I je nastaven směr posuvu vpravo,
pro x = log 0 je nastaven směr posunu vlevo.

Zapojení obou těchto registrů je naznačeno na obr. 4-52 a obr. 4-53.



Obr. 4-52: Zapojení posuvného registru s posunem vlevo

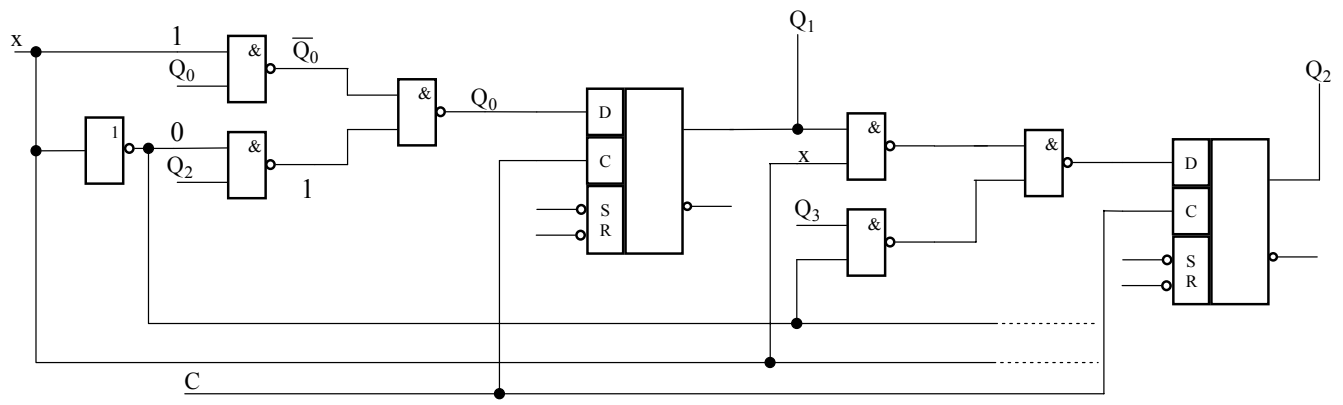


Obr. 4-53: Zapojení části posuvného registru s oboustranným posunem

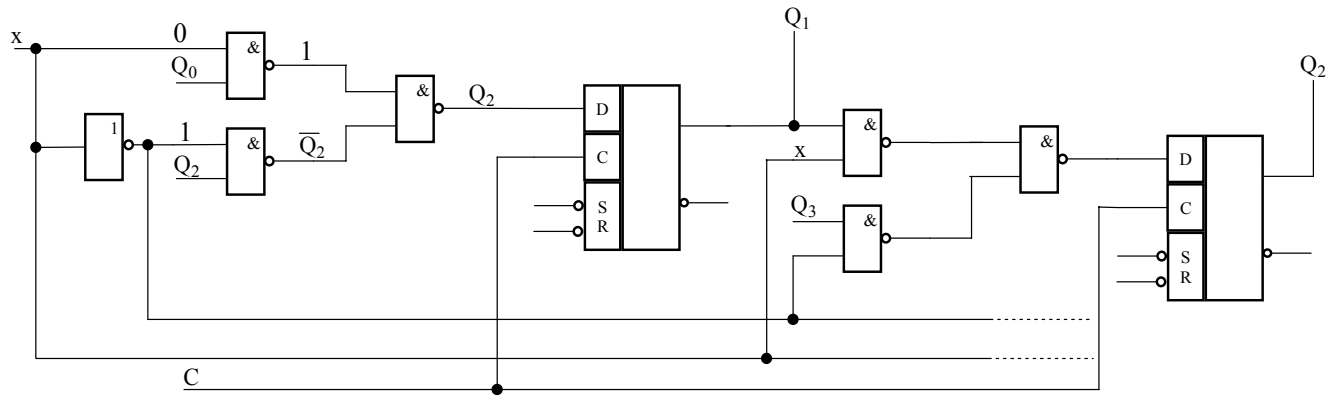
Hodnota log I na vstupu x obousměrného posuvného registru způsobí, že následující hradlo NAND přenese změny ze vstupu Q_0 na svůj výstup, a tím pádem i na vstup následujícího paměťového členu. Dochází k posunu informace vpravo.

Pokud je na vstupu x logická hodnota 0, přenesení změny ze vstupu Q_0 je blokováno, dochází však k přenesení změny ze vstupu Q_2 logického členu NAND na jeho výstup, a tím pádem i na vstup předcházejícího paměťového členu. Dochází k posunu informace vlevo.

Tato skutečnost je naznačena na následujících obrázcích 4-54 a 4-55.



Obr. 4-54: Zapojení části posuvného registru s oboustranným posunem při x = log I (posun vpravo)



Obr. 4-55: Zapojení části posuvného registru s oboustranným posunem při x = log 0 (posun vlevo)

K ZAPAMATOVÁNÍ 10

Společným pojmem **čítače** se označují zapojení paměťových členů, které umožňují registrovat počty impulsů.

Používají se dva způsoby registrace. V prvním případě se jedná o prosté počítání impulsů s výstupem v binárním tvaru a ve druhém případě o odečítání od přednastavené hodnoty.

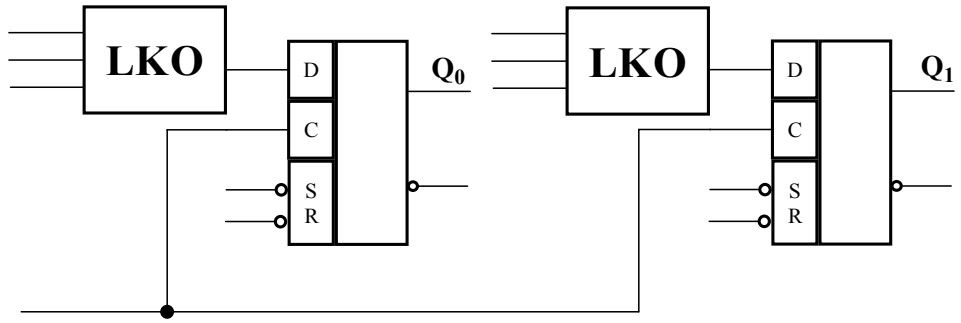
Čítače se realizují jako **synchronní** nebo **asynchronní** obvody.

Synchronní čítače

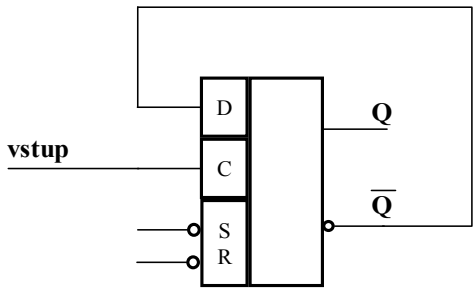
Synchronní čítače jsou navrhovány způsobem shodným s návrhem synchronních sekvenčních obvodů. Blokové schéma synchronního čítače je naznačeno na obr. 4-56 a je charakteristické tím, že každý paměťový člen má logickou hodnotu vstupní proměnné generovanou funkcí realizovanou kombinačním obvodem.

Asynchronní čítače

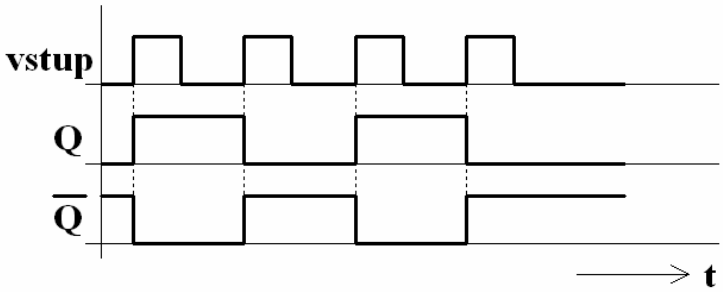
U asynchronních čítačů jsou počty impulsů registrovány rovněž v binárním tvaru, a protože váha vyššího řádu je vždy dvojnásobná, používá se k jejich pojmenování někdy názvu **dělička**. Základní zapojení čítače (resp. děličky 2) je uvedeno na obr. 4-57 a odpovídající časové průběhy na obr. 4-58. Vstupní signály *R* a *S* slouží k nastavení počátečního stavu.



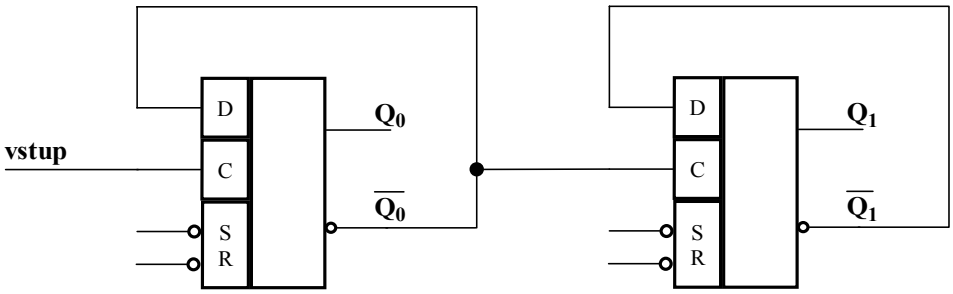
Obr. 4-56: Blokové schéma synchronního čítače



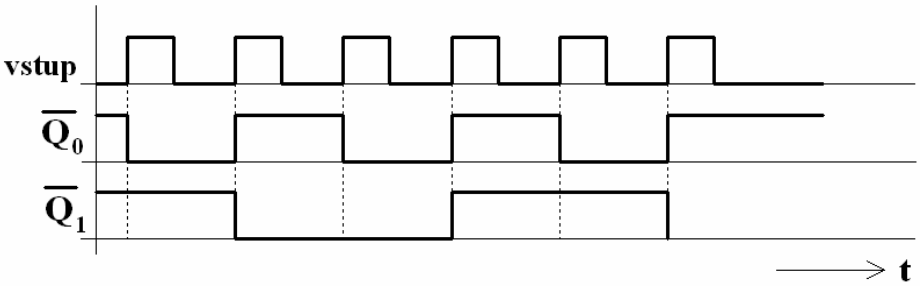
Obr. 4-57: Schéma asynchronního čítače s modulem 2 (děličky 2)



Obr. 4-58: Časové průběhy asynchronního čítače s modulem 2



Obr. 4-59: Schéma asynchronního čítače s modulem 4 (děličky 4)

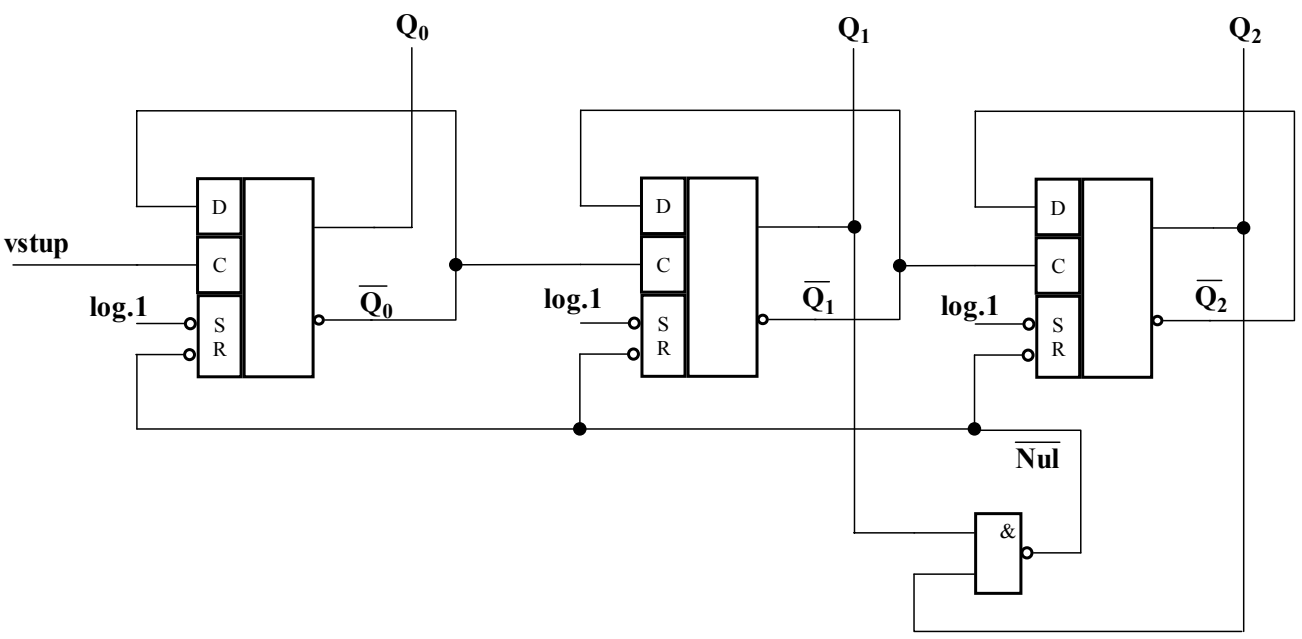


Obr. 4-60: Časové průběhy asynchronního čítače s modulem 4

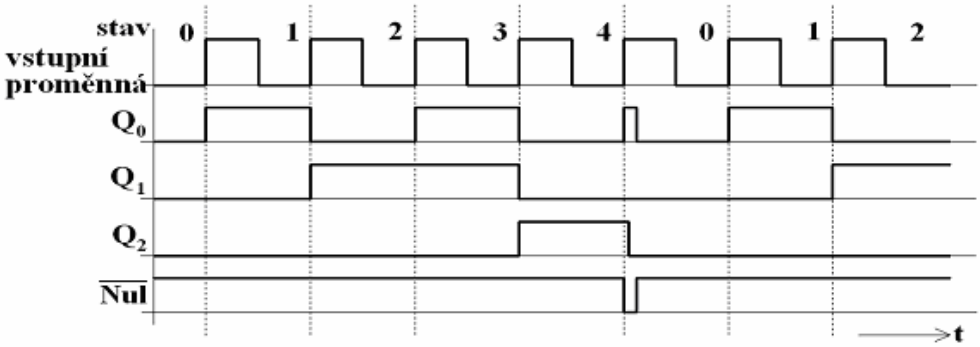
Kaskádní řazení čítačů (děliček).

Je zřejmé, že **děličky s modulem 2^n** lze vytvořit kaskádním zapojením děliček 2. Na obr. 4-59 je uvedeno blokové schéma děličky 4. Odpovídající časové průběhy jsou uvedeny na obr. 4-60.

V případě, že je zapotřebí realizovat **děličku, která není v modulu 2**, doplňuje se standardní zapojení asynchronní děličky kombinačním odvodem, který zajišťuje nulování čítače. Nulování čítače se provádí vstupním signálem *R* a kombinační obvod představuje v podstatě dekodér modulu čítače. Jako příklad je uvedeno zapojení čítače s modulem 5, obr. 4-61.



Obr. 4-61: Schéma asynchronního čítače s modulem 5 (děličky 5)



Obr. 4-62: Časové průběhy asynchronního čítače s modulem 5

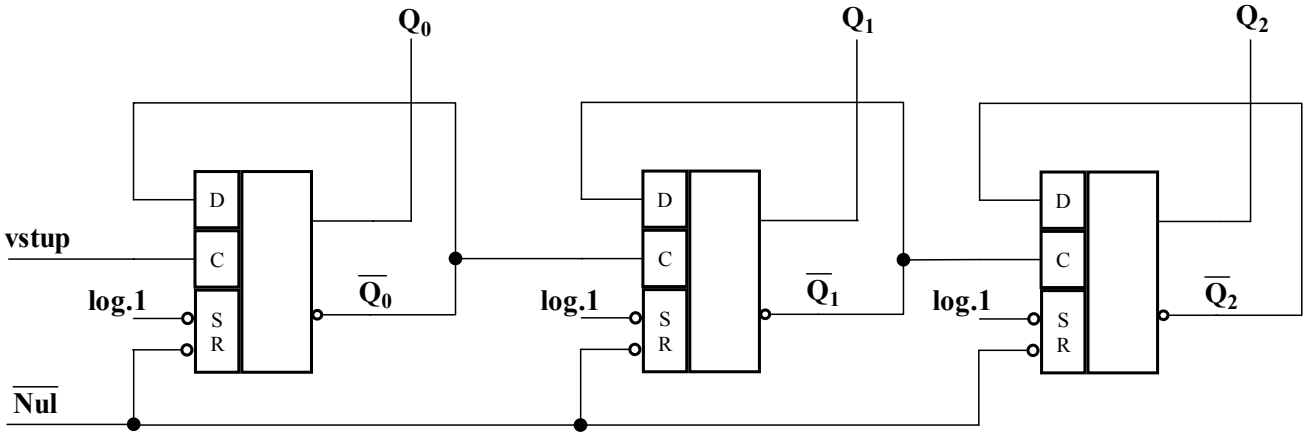
ŘEŠENÝ PŘÍKLAD

Nakreslete schéma zapojení a časové průběhy asynchronního čítače s modulem 8 (dělička 8).

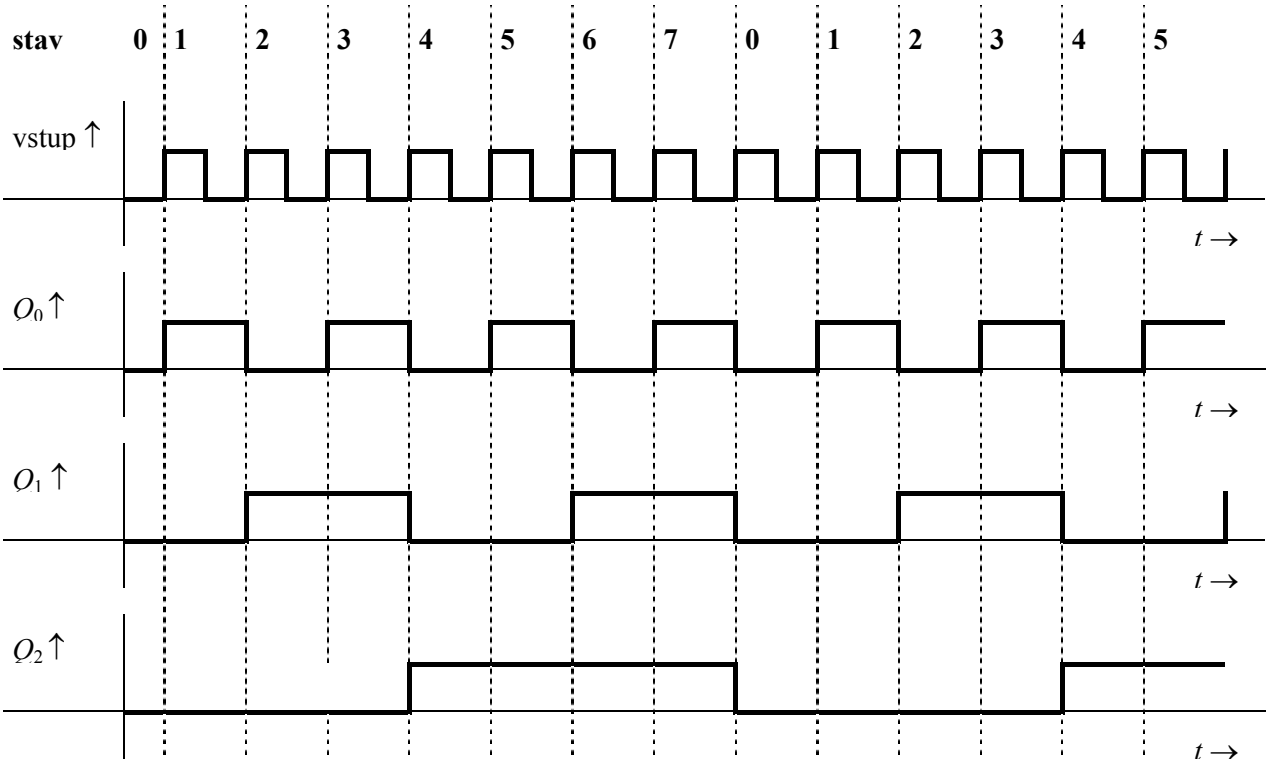
Řešení příkladu

Pro realizaci čítačů s modulem od 5 do 8 sestavíme děličku ze tří paměťových členů ($2^3 = 8$).

Schéma zapojení asynchronního čítače modulo 8 s možností nastavení počátečního stavu:



Odpovídající časové průběhy:



*

ŘEŠENÝ PŘÍKLAD

Nakreslete časové průběhy a schéma zapojení asynchronního čítače s modulem 10.

Řešení příkladu

Pro realizaci čítačů s modulem od 9 do 16 je zapotřebí sestavit děličku ze čtyř paměťových členů ($2^4 = 16$). Protože realizujeme děličku 10, která není v modulu 2, musíme doplnit standardní zapojení kombinačním obvodem (funkce *f*), který zajišťuje nulování čítače.

Pravdivostní tabulka funkce *f*:

stav	Q_3	Q_2	Q_1	Q_0	f
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
	1	0	1	0	0
	1	0	1	1	X
	1	1	0	0	X
	1	1	0	1	X
	1	1	1	0	X
	1	1	1	1	X

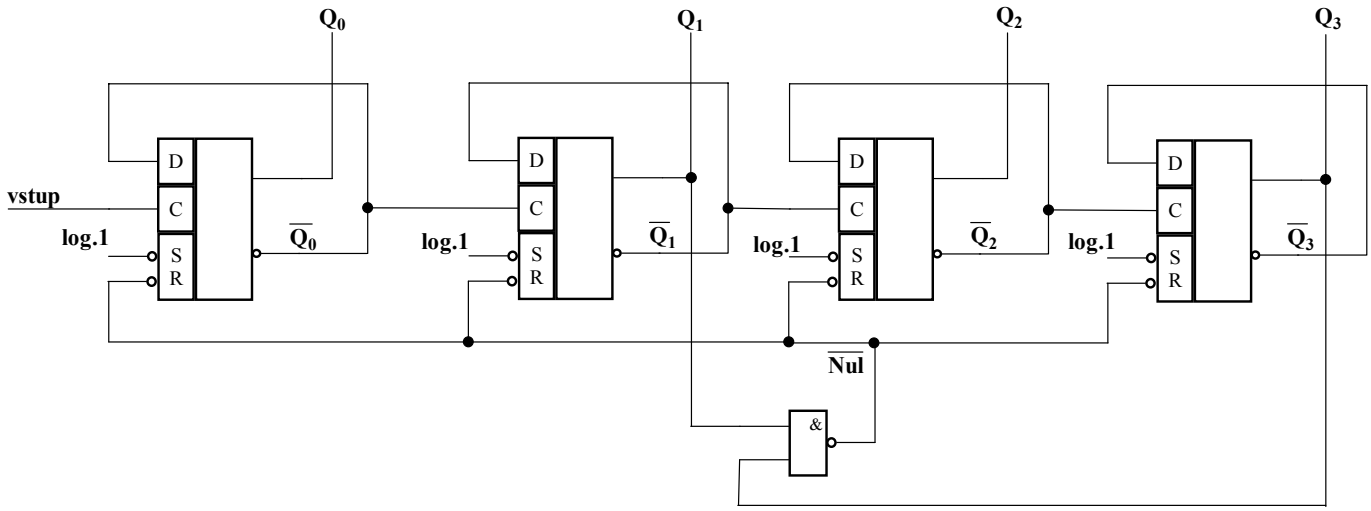
Karnaughova mapa funkce *f:*

$Q_3 \backslash Q_0 \backslash Q_1 \backslash Q_2$				
	1	1	1	1
	1	1	1	1
	1	X	X	X
	1	0	X	X

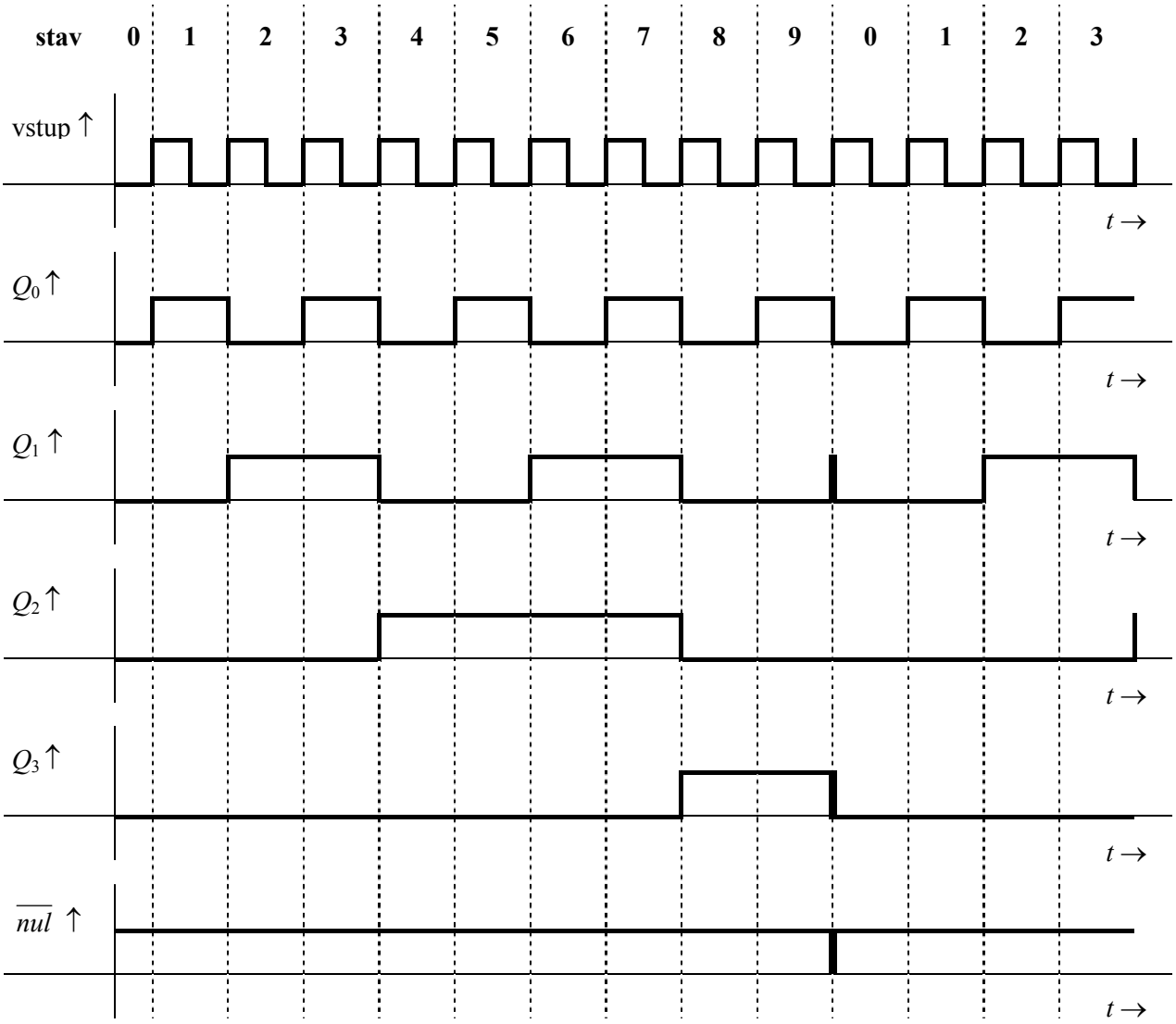
Funkce *f* je dána tímto booleovským výrazem:

$$f = \overline{Q_3} + \overline{Q_1} = \overline{\overline{\overline{Q_3} + \overline{Q_1}}} = \overline{Q_3 \cdot Q_1}$$

Schéma zapojení asynchronního čítače modulo 10:



Časové průběhy asynchronního čítače modulo 10:



*

4.4 Návrh generátoru binárních posloupností

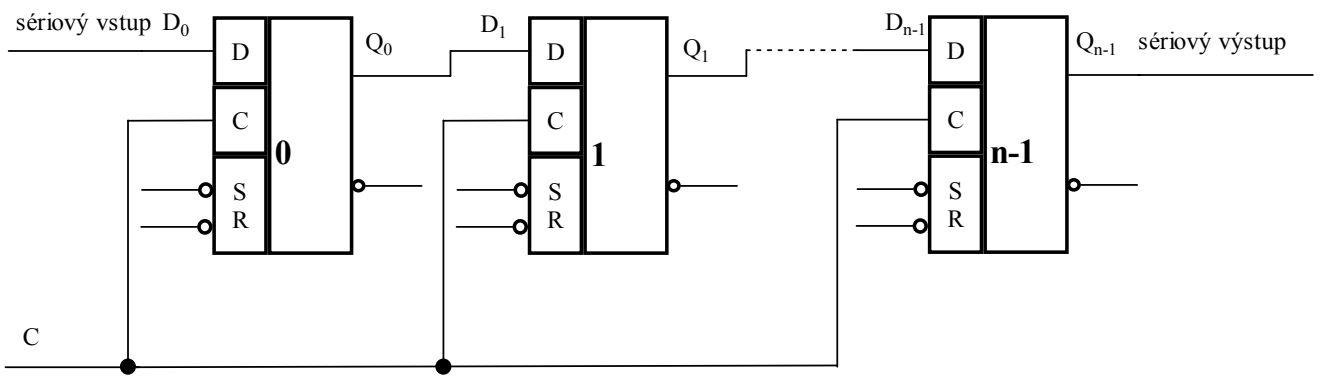
Samostatnou skupinu mezi synchronními sekvenčními obvody tvoří posuvné registry. Pro tyto obvody je charakteristické sériové zapojení paměťových členů, které na synchronizační impuls přenesou svoji informaci. Vstupní proměnná je tedy jedna, nazývaná rovněž jako sériový vstup stejně jako výstupní proměnná, zvaná sériový výstup. Blokové schéma paměťového registru je na obr. 4-63.

Matematicky lze sériové zapojení vyjádřit pro paměťové členy typu D podmínkou $D_{i+1} = Q_i$ a pro paměťové členy typu JK podmínkou $J_{i+1} = Q_i$ a $K_{i+1} = \overline{Q_i}$, kde i - představuje pořadové číslo paměťového členu. Nejrozšířenější je jejich aplikace při generování binární posloupnosti využívané v aritmetických jednotkách počítačů, resp. převodníků sériově - paralelních.

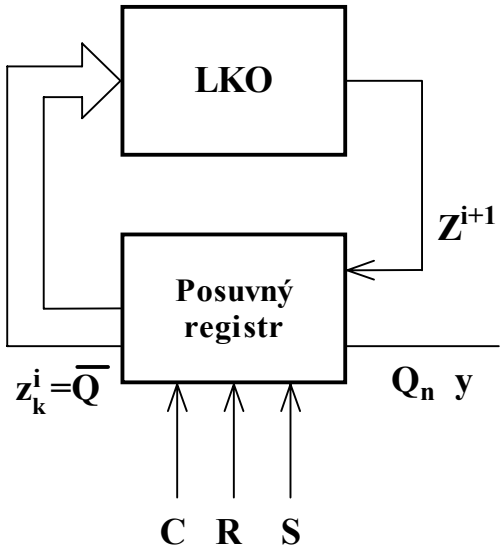
Jestliže máme generovat zadanou binární posloupnost a zpožďovací člen sekvenčního obvodu má být posuvný registr, redukuje se problém na stanovení jedné budící funkce, protože posuvný registr má pouze jednu vstupní proměnnou. Obecné schéma synchronního sekvenčního obvodu přechází do tvaru uvedeného na obr. 4-64.

Vektor proměnných $\overline{Q}$ představuje vnitřní sběrnici a je tvořen výstupy jednotlivých paměťových členů posuvného registru. Pro výstupní proměnnou y často platí, že $y = Q_n$. Blok LKO reprezentuje logický kombinační obvod, který dekóduje vnitřní stav posuvného registru a podle tohoto stavu vytváří funkci pro sériový vstup. Hledaná funkční závislost proto je

$$Z^{i+1} = f\left(\overline{Q}\right) = f\left(Q_0, Q_1, \ldots, Q_{n-1}\right) \qquad (4\text{-}40)$$



Obr. 4-63: Blokové schéma synchronního posuvného registru.



Obr. 4-64: Blokové schéma generátoru binární posloupnosti

Postup návrhu generátoru binární posloupnosti generované pomocí posuvného registru budeme demonstrovat na řešeném příkladu. Při návrhu musíme:

- stanovit délku posuvného registru,
- generovanou posloupnost transformovat do orientovaného grafu synchronního sekvenčního obvodu,
- stanovit budící funkci,
- realizovat generátor posloupnosti.

ŘEŠENÝ PŘÍKLAD

Generujte cyklickou binární posloupnost 1000101100110.

Řešení příkladu

Stanovení délky posuvného registru

Stanovení délky posuvného registru vychází z délky generované posloupnosti. V principu jsou možná dvě řešení:

- délka posuvného registru se rovná délce generované posloupnosti (u této varianty není nutné navrhovat kombinační obvod, protože pro budící funkci platí $Z^{i+1} = Q_n$; nevýhodou je značná délka posuvného registru),
- délka posuvného registru je menší než délka generované posloupnosti (vzhledem k podstatné redukci paměťových členů potřebných k realizaci se tato varianta běžně využívá a v dalším uvažujeme pouze s touto variantou).

K tomu, abychom jednoznačně transformovali binární posloupnost do orientovaného grafu, musí platit:

$$d \leq 2^n, \qquad (4\text{-}41)$$

kde d je délka binární posloupnosti a n je počet paměťových členů posuvného registru. V našem případě je $d = 13$ a podle rovnice *(4-40)* je $n = 4$.

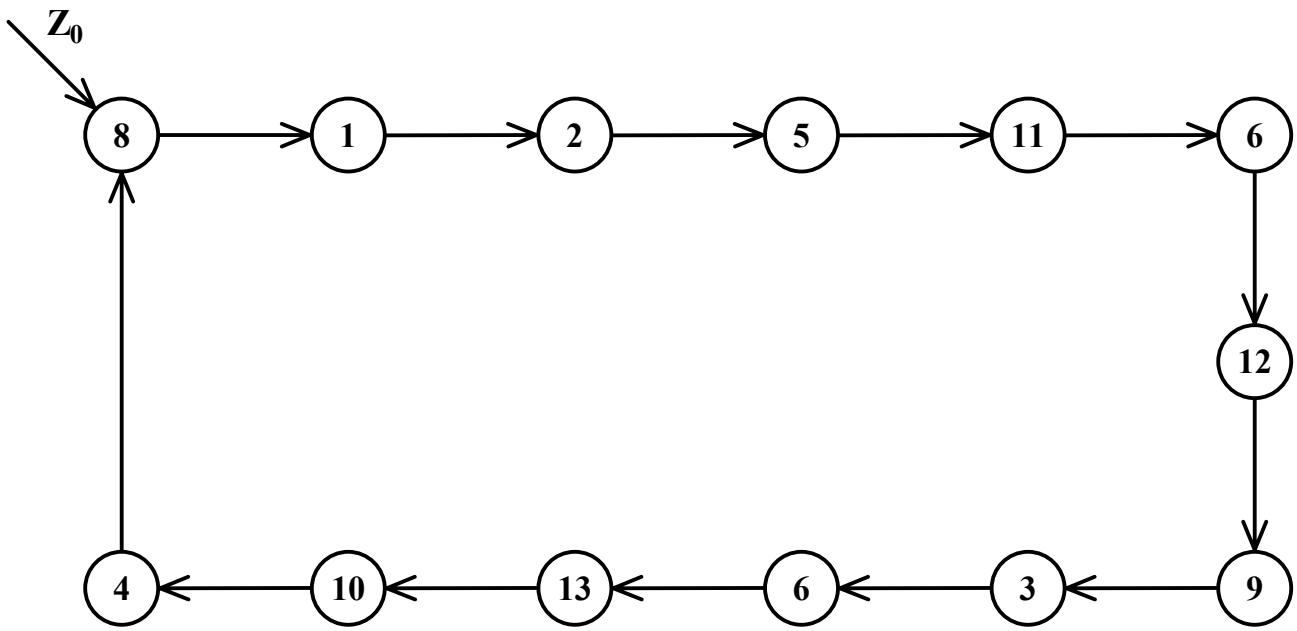
Transformace binární posloupnosti do orientovaného grafu

Transformace binární posloupnosti do orientovaného grafu se provede rozdělením binární posloupnosti na posloupnost *n*-bitových čísel (stavů posuvného registru). Ke kódování stavů se standardně používá binární kódování.

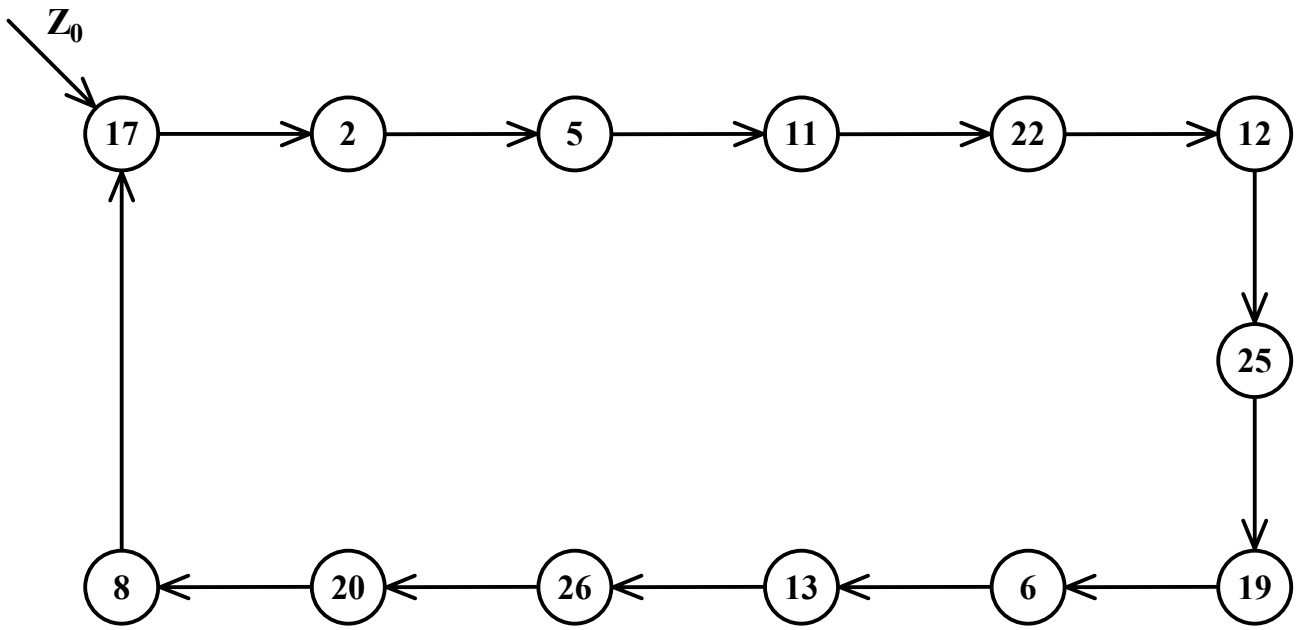
Z orientovaného grafu pro n = 4 vyplývá, že zvolený způsob transformace není jednoznačný, protože stav č.6 se v průběhu generované posloupnosti vyskytuje dvakrát, přičemž v prvním případě po stavu 6 následuje stav 12 a ve druhém případě stav 13.

Je proto nutné zvýšit redundantnost transformace, což znamená zvýšení délky posuvného registru na n = 5 a celou transformaci opakovat.

Orientovaný graf pro binární posloupnost $n = 4$:



Orientovaný graf pro binární posloupnost $n = 5$:



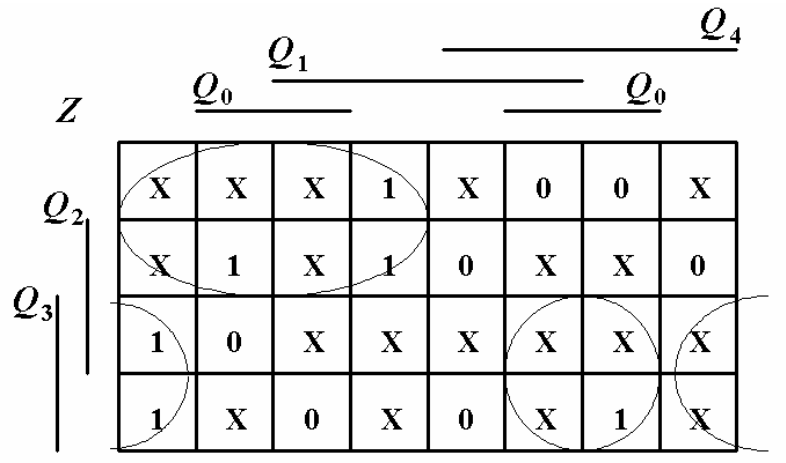
Stanovení budící funkce pro sériový vstup se provádí běžným způsobem, např. pomocí Karnaughových map. Obsah mapy potom tvoří logické hodnoty budící funkce následujícího stavu. Budící funkce má podle vztahu *(4-41)* v tomto případě 5 proměnných.

Stanovení budící funkce ***Z***

Pravdivostní tabulka budící funkce sériového vstupu ***Z***:

stav	Q_4	Q_3	Q_2	Q_1	Q_0	Z
17	1	0	0	0	1	0
2	0	0	0	1	0	1
5	0	0	1	0	1	1
11	0	1	0	1	1	0
22	1	0	1	1	0	0
12	0	1	1	0	0	1
25	1	1	0	0	1	1
19	1	0	0	1	1	0
6	0	0	1	1	0	1
13	0	1	1	0	1	0
26	1	1	0	1	0	0
20	1	0	1	0	0	0
8	0	1	0	0	0	1

Pravdivostní tabulka budící funkce ***Z***



Karnaughova mapa budící funkce ***Z***

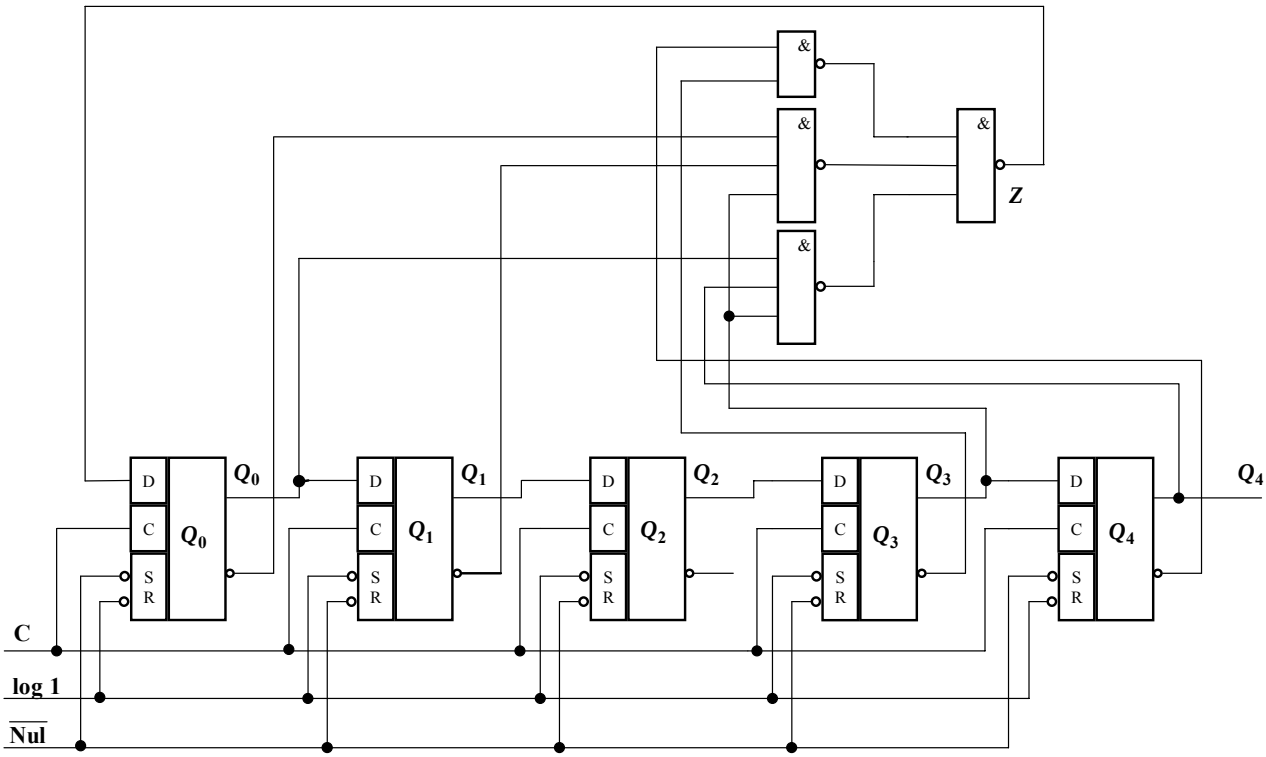
Protože se jedná o funkci rozšířenou, je při realizaci funkce převedena na funkci úplnou a místo neurčitých hodnot je přiřazena logická hodnota log. 0, resp. log. I

Pokud pro budící funkci zvolíme např. výpis v součtovém tvaru, dostaneme:

$$Z = \overline{Q_4} \cdot \overline{Q_3} + Q_3 \cdot \overline{Q_1} \cdot \overline{Q_0} + Q_4 \cdot Q_3 \cdot Q_0 \qquad (4\text{-}42)$$

Realizace generátoru posloupnosti je uvedena na následujícím schématu, včetně obvodů pro nastavení počátečního stavu. Posuvný registr je zapojen z paměťových členů typu D.

Realizace generátoru posloupnosti



*

ŘEŠENÝ PŘÍKLAD

Generujte binární posloupnost 1000110010110 pomocí posuvného registru. Nakreslete schéma zapojení s paměťovými členy MH 7474.

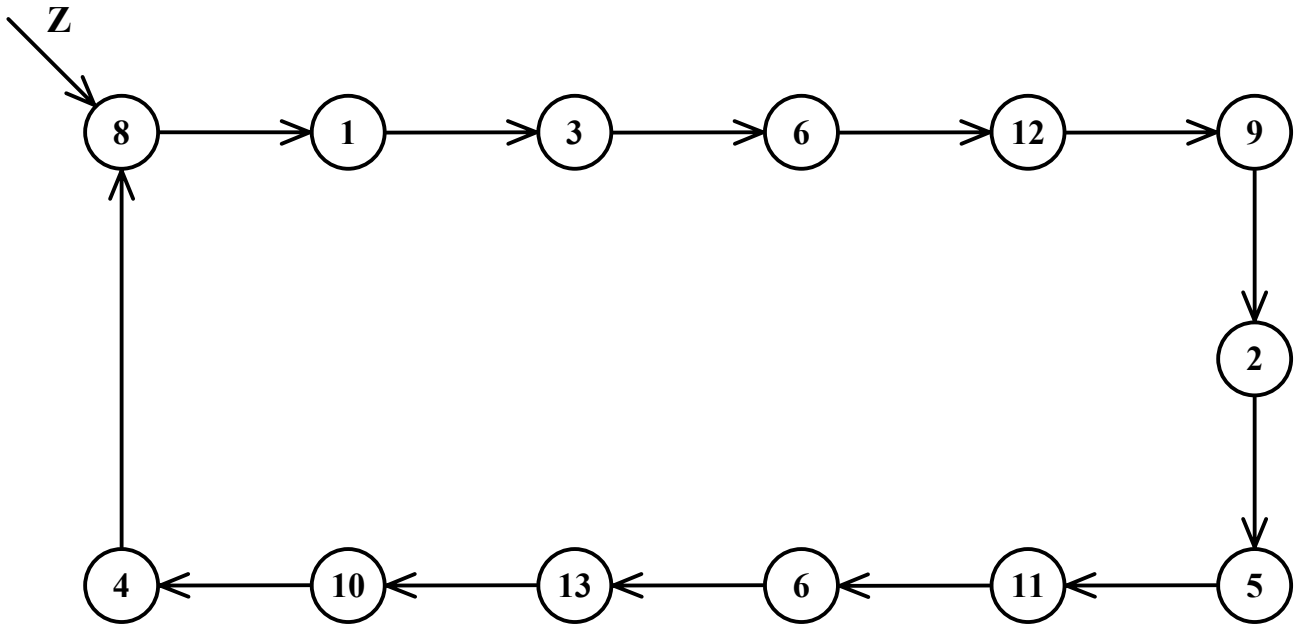
Řešení příkladu

Délka posuvného registru je $d = 13$ a podle vztahu *(4-41)* jsou pro realizaci generátoru binární posloupnosti nutné alespoň 4 paměťové členy ($n = 4$).

Stanovení délky posuvného registru

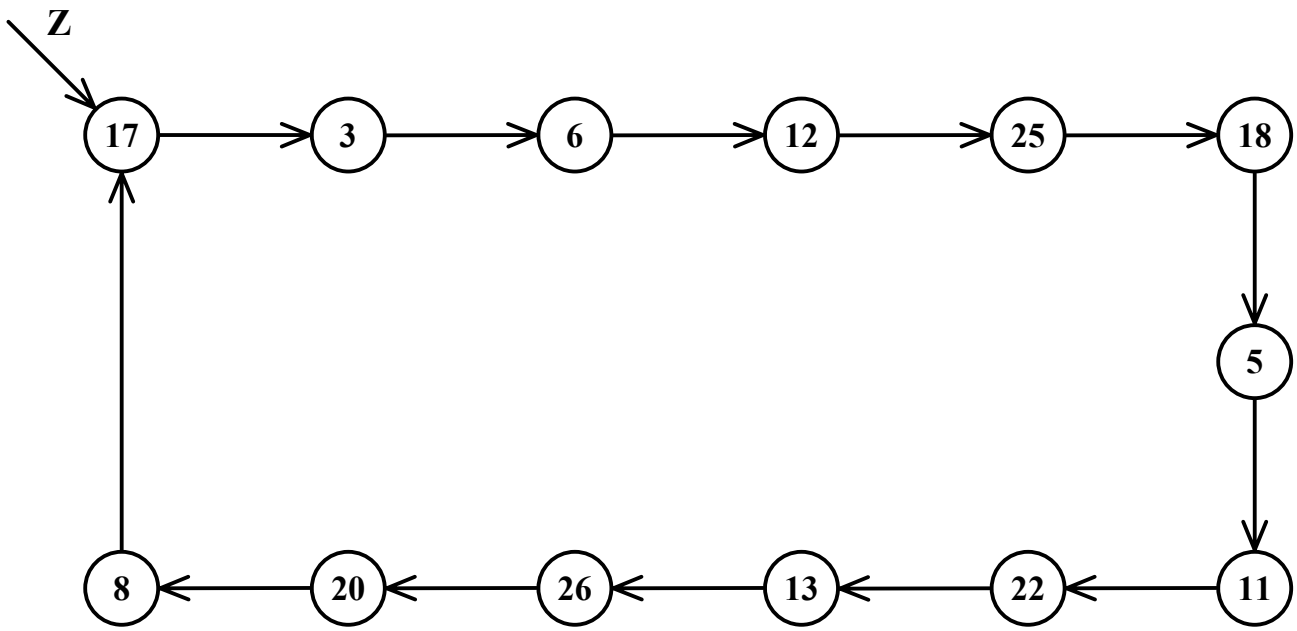
Orientovaný graf pro binární posloupnost $n = 4$:

Transformace binární posloupnosti do orientovaného grafu



V diagramu stavů se vyskytuje dvakrát stav 6, a proto musíme zvýšit počet paměťových členů a sestavit nový orientovaný graf.

Orientovaný graf pro binární posloupnost $n = 5$:



Sestavení budící funkce sériového vstupu **Z**:

Pravdivostní tabulka budící funkce **Z**

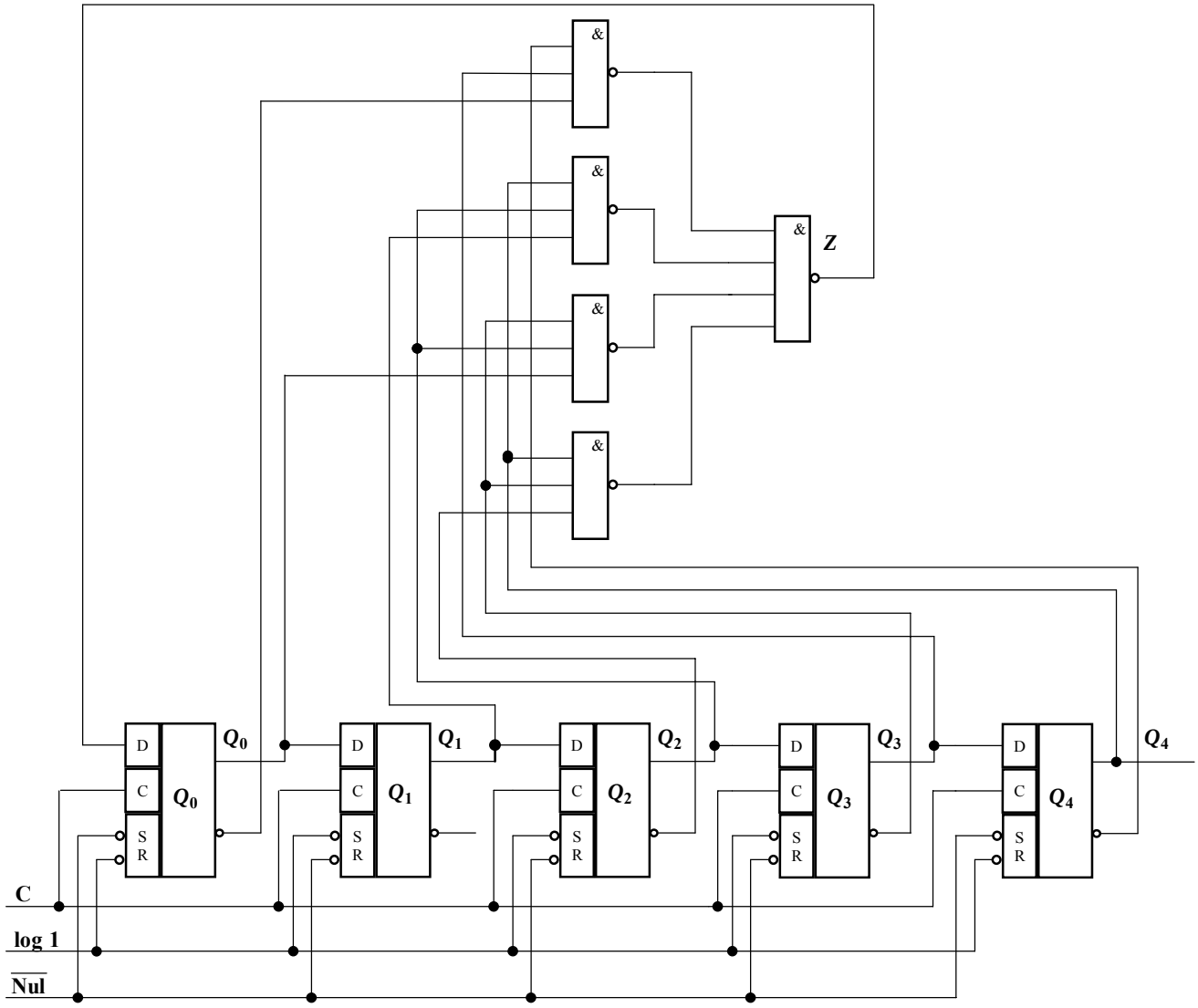
stav	Q_4	Q_3	Q_2	Q_1	Q_0	Z
17	1	0	0	0	1	1
3	0	0	0	1	1	0
6	0	0	1	1	0	0
12	0	1	1	0	0	1
25	1	1	0	0	1	0
18	1	0	0	1	0	1
5	0	0	1	0	1	1
11	0	1	0	1	1	0
22	1	0	1	1	0	0
13	0	1	1	0	1	0
26	1	1	0	1	0	0
20	1	0	1	0	0	0
8	0	1	0	0	0	1

Karnaughova mapa budící funkce ***Z***

Z	Q_4	Q_3	Q_2				Q_4	
	X	X	X	1	1	X	0	X
Q_1	X	1	0	X	X	X	1	0
Q_0	0	X	X	0	X	X	X	X
	X	1	0	X	0	X	X	1

$$Z = \overline{Q}_4 Q_3 \overline{Q}_0 + Q_4 Q_2 Q_1 + \overline{Q}_3 Q_2 Q_0 + Q_4 \overline{Q}_3 \overline{Q}_2$$

Schéma zapojení posuvného registru:



SHRNUTÍ KAPITOLY LOGICKÉ SEKVENČNÍ OBVODY

Σ

Shrnutí

Po prostudování této kapitoly jsme schopni navrhnout i realizovat logický sekvenční obvod pomocí různých paměťových členů. Seznámili jsme se s funkcí, strukturou, návrhem a s realizací posuvných registrů, synchronních a asynchronních čítačů a s generátory binárních posloupností.

5 ARITMETICKO-LOGICKÁ JEDNOTKA

ČAS POTŘEBNÝ KE STUDIU

Celkový doporučený čas k prostudování KAPITOLY je **10** hodin.

RYCHLÝ NÁHLED DO PROBLEMATIKY KAPITOLY ARITMETICKO-LOGICKÁ JEDNOTKA

Základem každého číslicového počítače je centrální jednotka, jejíž součástí je i aritmeticko logická jednotka. Tato jednotka zabezpečuje matematické operace s čísly, a to sčítání, odčítání, násobení, dělení, porovnávání. Hardwarová realizace všech těchto operací však souvisí s typem procesoru a především s jeho instrukčním souborem. Dalším důležitým hlediskem je způsob zobrazování čísel, kdy se standardem stalo zobrazování v pohyblivé řádové čárce.

CÍLE KAPITOLY ARITMETICKO-LOGICKÁ JEDNOTKA

- Budete umět sestavit logické funkce pro realizaci aritmetických operací,
- získáte ucelený přehled o problematice sčítání, odčítání, dělení, násobení, porovnávání a jejich následné aplikaci pomocí logických funkcí a hradel,
- po úspěšném zvládnutí této problematiky budete schopni navrhnout aritmeticko logickou jednotku.
-

KLÍČOVÁ SLOVA KAPITOLY ARITMETICKO-LOGICKÁ JEDNOTKA

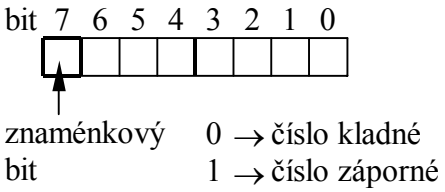
Klíčová slova

Aritmeticko - logická jednotka, sčítání, odčítání, násobení, dělení, porovnávání, jednotkový doplněk, dvojkový doplněk, sčítačka, násobička, mantisa, exponent

5.1 Způsob zobrazování celých čísel

V předchozích kapitolách jsme při zobrazování čísel uvažovali pouze čísla kladná, a to buď celá nebo desetinná. Pokud potřebujeme zobrazovat zároveň čísla záporná, je zřejmé, že musíme počet bitů čísla zvětšit o 1, o tak zvaný znaménkový bit. Standardně je pro znaménko využíván bit s nejvyšší váhou. Pokud budeme uvažovat pro zobrazování dat 8 bitů, můžeme zobrazovat celá kladná čísla v rozsahu $0 \div 127$, viz. obr. 5-1.

Je potřebné mít na paměti, že tomuto způsobu zobrazování čísel by odpovídaly dvě nuly, nula kladná 00_H a nula záporná 80_H. Z tohoto důvodu se záporná čísla vyjadřují doplňkem.



Obr. 5-1: Struktura zobrazení celých čísel.

5.1.1 Vyjádření záporných čísel jednotkovým doplňkem

Máme-li číslo A = -41 vyjádřit v jednotkovém doplňku, potom platí

$$^{1}A = 2^{n} - 1 - A, \qquad (5\text{-}1)$$

kde ^{1}A je jednotkový doplněk a člen $2^{n} - 1$ představuje číslo, které má ve všech bitech jedničky. Jestliže máme pro vyjádření čísla k dispozici 8 bitů, n = 8, potom člen $2^{8} - 1 = 225$ představuje číslo, které má na všech bitech jedničky. Když od tohoto čísla odečteme číslo A, např. 41, dostaneme $^{1}A = 214$, což představuje číslo záporné. Zobrazení těchto čísel je na obr. 5-2. Porovnáme-li tato dvě zobrazení zjistíme, že logické hodnoty v odpovídajících si bitech jsou vzájemně negovány. Je zřejmé, že při zobrazování celého záporného čísla v jednotkovém doplňku můžeme postupovat tak, že binárně vyjádříme číslo kladné a potom všechny bity negujeme. Je však nutné upozornit, že k číslu $+0$ je jednotkovým doplňkem záporná nula FF_H a že problém dvojí nuly tento způsob zobrazení neřeší.

7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	0

$^{1}\boldsymbol{A} = 214$

0	0	1	0	1	0	0	1

$\boldsymbol{A} = 41$

Obr. 5-2: Zobrazení čísla v jednotkovém doplňku.

5.1.2 Vyjádření záporných čísel ve dvojkovém doplňku

Zobrazení záporných čísel ve dvojkovém doplňku je dáno vztahem

$$^{2}A = 2^{n} - A \qquad (5\text{-}2)$$

a po dosazení do vztahu *(5-1)* dostaneme

$$^{2}A = {}^{1}A + 1 \qquad (5\text{-}3)$$

Ze vztahu *(5-3)* vyplývá, že vyjádření čísla ve dvojkovém doplňku docílíme tak, že číslo nejprve vyjádříme v jednotkovém doplňku a potom přičteme 1. Zobrazení čísla $A = -41$ ve dvojkovém doplňku je uvedeno na obr. 5-3.

7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	1

Obr. 5-3: Zobrazení ve dvojkovém doplňku, A = -41.

Rozsah zobrazovaných čísel v 8-bitovém slově tedy je pro celá kladná čísla ⟨0,127⟩ a pro celá záporná čísla ⟨-1,-128⟩. V tomto vyjádření již existuje pro nulu pouze jedno zobrazení, a to 0.

5.2 Sčítání

Sčítání je nejdůležitější operací, neboť tvoří základ pro všechny další aritmetické operace. Sčítání se v binární soustavě provádí podle stejného algoritmu jako v desítkové soustavě. Máme-li čísla *A*, *B* a chceme-li je sečíst, můžeme psát

$$S = A + B \qquad (5\text{-}4)$$

nebo ve tvaru polynomu

$$\sum_{i=0}^{n} S_i 2^i = \sum_{i=0}^{n-1} a_i \cdot 2^i + \sum_{i=0}^{n-1} b_i \cdot 2^i \qquad (5\text{-}5)$$

Ve vztahu *(5-5)* je nutné si uvědomit, že v závislosti na koeficientech a_i, b_i může docházet k tzv. přenosu do vyššího řádu (při sčítání čísel v desítkové soustavě rovněž platí, že $5+5=10$). Z tohoto důvodu musí mít polynom součtu větší rozsah o 1 bit.

Pro součet dvou odpovídajících si členů polynomů, resp. pro jeden řád proto můžeme psát

$$c_{i+1} \cdot 2^{i+1} + S \cdot 2^i = a_i \cdot 2^i + b_i \cdot 2^i + c_i \cdot 2^i, \qquad (5\text{-}6)$$

kde koeficient c_{i+1} představuje přenos do vyššího řádu a koeficient c_i přenos z nižšího řádu. Rovnice *(5-6)* je algoritmem pro sčítání dvou celých kladných čísel.

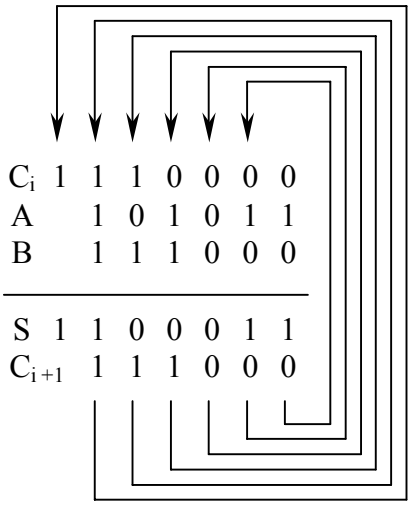
ŘEŠENÝ PŘÍKLAD

Sečtěte čísla *A* = 43 a *B* = 56ve dvojkové soustavě.

Řešení příkladu

Pro číslo *A* = 43 vyjádřené v binární soustavě platí $A = 101011_2$ a pro číslo *B* = 56, $B = 111000_2$.

Algoritmus sčítání



Součet zadaných čísel $S = 11000011_2 = 99_{10}$. Když provedeme součet v desítkové soustavě, dostaneme $S = 43 + 56 = 99$. Popsaný algoritmus je nazýván úplnou binární sčítačkou.

*

Úplná binární sčítačka

Logický obvod realizující úplnou binární sčítačku je obvod kombinační a lze ho pro jeden bit blokově znázornit podle obr. 5-4. Chování obvodu je potom dáno tabulkou na obr. 5-5.

Obrázek 5-4: Blokové schéma binární sčítačky.

a_i	b_i	c_i	S_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Obr. 5-5: Pravdivostní tabulka binární sčítačky.

S_i

c_i \ a_i, b_i	0	1	0	1
	1	0	1	0

c_{i+}

c_i \ a_i, b_i	0	0	1	0
	0	1	1	1

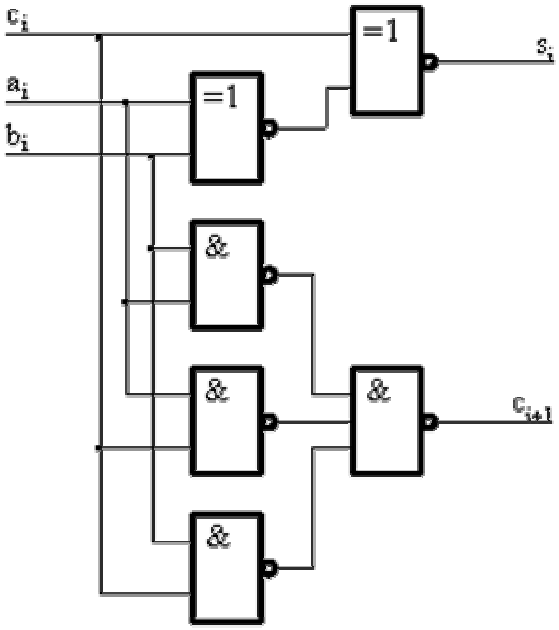
Obr. 5-6: Mapy pro proměnné S_i a c_{i+1}

K sestavení výrazů výstupních proměnných zvolíme metodu Karnaughových map, viz. obr. 5-6. Pro výpis funkcí v součtovém tvaru dostaneme vztahy:

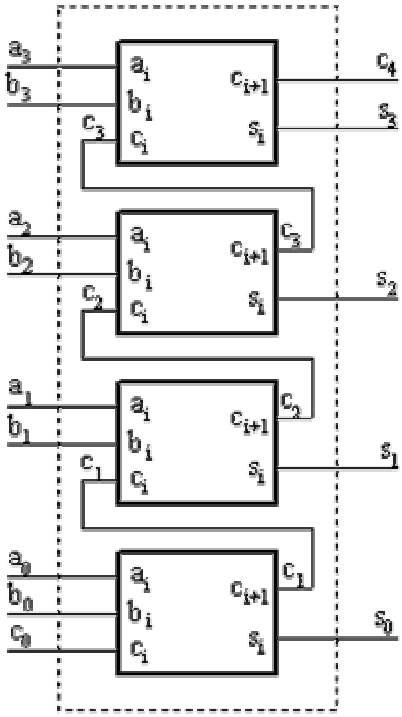
$$S_i = \overline{c}_i\left(a_i\overline{b}_i + \overline{a}_i b_i\right) + c_i\left(\overline{a}_i\overline{b}_i + a_i b_i\right) \qquad (5\text{-}7)$$

$$c_{i+1} = a_i b_i + \left(a_i + b_i\right) \qquad (5\text{-}8)$$

K realizaci výrazu *(5-7)* je vhodné zvolit hradlo s funkcí EXCLUSIVE-OR (MH 7486) a k výrazu *(5-8)* hradel NAND. Logická síť pro jeden bit úplné binární sčítačky je uvedena na obr. 5-7.
Logická síť *n*-bitové sčítačky potom představuje zapojení *n*-stupňů. Schéma 4-bitové sčítačky, hradlo MH 7483, je uvedeno na obr. 5-.8. V souladu s algoritmem sčítání je u zapojení obvodu jako sčítačka na vstupu c_0 úroveň log 0.



Obr. 5-7: Logická síť úplné jednobitové sčítačky.



Obr. 5-8: Blokové schéma 4-bitové sčítačky.

Kanál zrychleného přenosu

Když uvážíme využitelnost uvedeného algoritmu pro sčítačku s větší délkou datového slova, např. 32 nebo 64 bitů, musíme algoritmus charakterizovat jako zdlouhavý. Je to tím, že nejdříve se musí vyhodnotit přenos z nižšího stupně a až potom je možné realizovat sčítání ve vyšším řádu. K potřebnému času na součet je nutno přičíst i zpoždění jednotlivých hradel. Z těchto důvodů se u vícebitových sčítaček realizuje tzv. **kanál zrychleného přenosu**, jehož princip spočívá v možnosti bezprostředního výpočtu všech přenosů do vyšších řádů. Samozřejmě tato cesta vede ke složitějším funkcím popisujících generování přenosu, a tím i k složitějšímu obvodovému řešení.

Když aplikujeme vztah *(5-8)* na výpočet přenosu c_1, dostaneme

$$c_1 = a_0 b_0 + c_0 (a_0 + b_0) \qquad (5\text{-}9)$$

a analogicky pro přenos c_2

$$c_2 = a_1 b_1 + c_1 (a_1 + b_1), \qquad (5\text{-}10)$$

přenos c_3

$$c_3 = a_2 b_2 + c_2 (a_2 + b_2) \qquad (5\text{-}11)$$

a přenos c_4

$$c_4 = a_3 b_3 + c_3 (a_3 + b_3) \qquad (5\text{-}12)$$

Dosazením rovnice *(5-9)* do rovnice *(5-10)* dostaneme funkční závislost přenosu pouze na vstupních proměnných

$$\begin{aligned} c_2 &= a_1 b_1 + [a_0 b_0 + c_0 (a_0 + b_0)](a_1 + b_1) = \\ &= a_1 b_1 + a_0 b_0 (a_1 + b_1) + c_0 (a_0 + b_0)(a_1 + b_1) \end{aligned} \qquad (5\text{-}13)$$

Dosazením výrazu *(5-13)* do výrazu *(5-11)* potom dostaneme

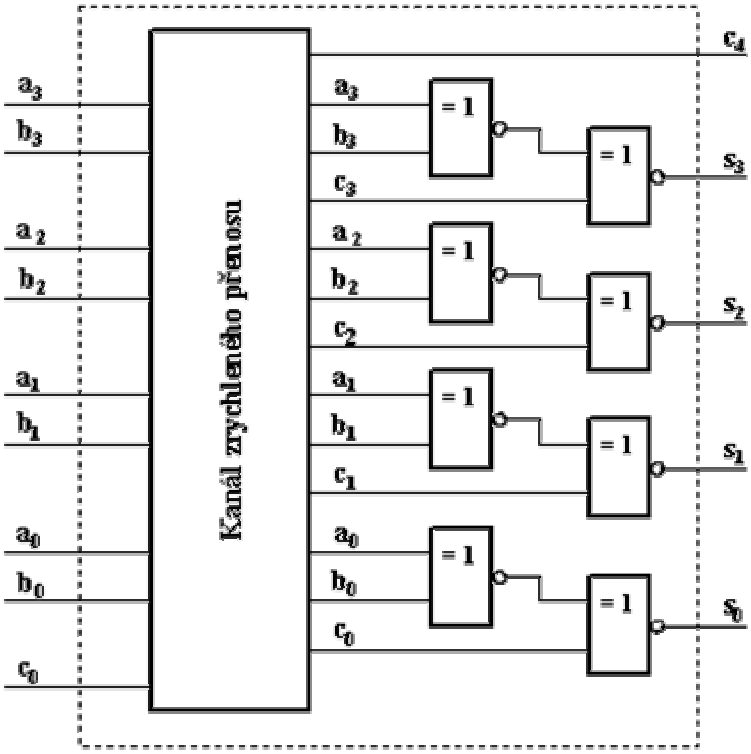
$$\begin{aligned} c_3 &= a_2 b_2 + a_1 b_1 (a_2 + b_2) + a_0 b_0 (a_1 + b_1)(a_2 + b_2) + \\ &+ c_0 (a_0 + b_0)(a_1 + b_1)(a_2 + b_2) \end{aligned} \qquad (5\text{-}14)$$

a podobně

$$c_4 = a_3b_3 + a_2b_2(a_3+b_3) + a_1b_1(a_2+b_2)(a_3+b_3) + \\ + a_0b_0(a_1+b_1)(a_2+b_2)(a_3+b_3) + c_0(a_0+b_0)(a_1+b_1)(a_2+b_2)(a_3+b_3) \qquad (5\text{-}15)$$

Funkční vztahy *(5-9), (5-13), (5-14), (5-15)* jsou potom vztahy realizované v kanálu zrychleného přenosu. Zapojení 4-bitové sčítačky takto dostane tvar uvedený na obr. 5-9.

Obvody, které mají realizováno sčítání tímto způsobem jsou např. MH 74LS83A, 74LS283A nebo ALU 74181.



Obr. 5-9: Blokové schéma sčítačky s kanálem zrychleného přenosu.

5.3 Odčítání

Stejným způsobem jako je uvedeno v kapitole 5.2, by bylo možné odvodit základní vztah pro odčítání

$$a_i \cdot 2^i - b_i \cdot 2^i - c_i \cdot 2^i = r_i - c_{i+1} \cdot 2^i \qquad (5\text{-}16)$$

Pokud ale uvážíme, že odčítání je vlastně přičítání čísla opačného, můžeme k odčítání dvou čísel využít obvodů sčítačky. Je však zapotřebí definovat číslo opačné, tedy číslo záporné, což bylo provedeno v kapitole 5.1. Rozlišujeme tedy odčítání s jednotkovým doplňkem a dvojkovým doplňkem.

5.3.1 Odčítání s jednotkovým doplňkem

Máme-li realizovat rozdíl dvou kladných čísel definovaný jako $R = A - B$ musíme rozlišovat dva případy.

- Když $A > B$ můžeme psát $R = A - B = A +^1 B - 2^n + 1$, odkud vyplývá, že

$$A - B = A +^1 B + 1 - 2^n \qquad (5\text{-}17)$$

K dosažení správného rozdílu čísel $A - B$ potřebujeme přičíst chybějící jedničku a odečíst hodnotu 2^n, čehož se docílí tzv. kruhovým přenosem.

ŘEŠENÝ PŘÍKLAD

Vypočítejte rozdíl čísel $R = A - B$ kdy $A = 54$ a $B = 13$

Řešení příkladu

$$A_{10} = 54 \Rightarrow A_2 = 110110_2$$

$$B_{10} = 13 \Rightarrow B_2 = 1101_2 \Rightarrow {}^1B = 11110010_2$$

Algoritmus odečítání s jednotkovým doplňkem $R = A - B$, $A > B$

A		0	0	1	1	0	1	1	0
${}^1B$		1	1	1	1	0	0	1	0
kruhový přenos	1	0	0	1	0	1	0	0	0
								+	1
R	1	0	0	1	0	1	0	0	1

kladný výsledek $\rightarrow A - B = 41$

Pokud výsledek zkontrolujeme s výpočtem v desítkové soustavě dospějeme ke shodě.

*

- Když $A \le B$, potom výsledný rozdíl ${}^1R = A - B = 2^n - 1 - R \Rightarrow$ bude záporný a tedy $R = 2^n - 1 - A + B = {}^1A + B$. Odtud vyplývá, že pokud je rozdíl záporný, je výsledek přímo v jednotkovém doplňku.

ŘEŠENÝ PŘÍKLAD

Vypočítejte rozdíl čísel $R = B - A$ kdy $A = 54$ a $B = 13$.

Řešení příkladu

Algoritmus odečítání s jednotkovým doplňkem $R = B - A$, $B \le A$

A_2		0	0	1	1	0	1	1	$0_2 \rightarrow {}^1A = 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1$
B_2					1	1	0	1_2	
B		0	0	0	0	1	1	0	1
${}^1A$		1	1	0	0	1	0	0	1
${}^1R$	0	1	1	0	1	0	1	1	0

záporný výsledek $\rightarrow B - A = -41_{10}$

*

5.3.2 Odčítání s dvojkovým doplňkem

Odčítání dvou čísel pomocí dvojkového doplňku se provádí stejným způsobem jako při jednotkovém doplňku. Je-li rozdíl čísel definován jako $R = A - B$ opět musíme rozlišovat:

- Když $A > B$, potom $R = A - B = A + 2^n - B$ odtud platí, že

$$A - B = A + 2^n - B \qquad (5\text{-}18)$$

K dosažení správného výsledku rozdílu čísel $A - B$ musíme ještě přičíst hodnotu 2^n. Protože člen 2^n představuje bit s váhou větší než lze v datovém slově zobrazit, je tento bit umístěn vlevo od znaménkového bitu, nemusíme ho v dalším zpracování akceptovat.

ŘEŠENÝ PŘÍKLAD

Vypočítejte rozdíl čísel $R = A - B$, kdy $A = 42$ a $B = 14$.

Řešení příkladu

$A_{10} = 42 \Rightarrow A_2 = 00101010$

$B_{10} = 14 \Rightarrow B_2 = 00001110$

${}^1B = 11110001$

${}^2B = 11110010$

Algoritmus odečítání s dvojkovým doplňkem $R = A - B$, $A > B$

$$\begin{array}{lcccccccc} A & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ {}^2B & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ \hline R & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{array}$$

$\downarrow$

kladný výsledek $\rightarrow A - B = 28_{10}$

*

- Když $A \leq B$, potom rozdíl bude záporný a bude ve dvojkovém doplňku. Můžeme proto psát, že
  ${}^2R = A - B = 2^n - R = A - B \Rightarrow R = 2^n - A + B$ a tedy platí

$$R = B + {}^2A \qquad (5\text{-}19)$$

Na základě vztahu *(5-19)* platí, že pokud je rozdíl čísel záporný je výsledek přímo jeho dvojkovým doplňkem.

ŘEŠENÝ PŘÍKLAD

Vypočítejte rozdíl čísel $R = B - A$, kdy $A = 42$ a $B = 14$.

Řešení příkladu

Algoritmus odečítání s dvojkovým doplňkem
$R = B - A,\ B \leq A$

A_2	0	0	1	0	1	0	1	0
${}^1A$	1	1	0	1	0	1	0	1
${}^2A$	1	1	0	1	0	1	1	0
B_2	0	0	0	0	1	1	1	0
${}^2R$	1	1	1	0	0	1	0	0

↓

záporný výsledek → $B - A = -28_{10}$

*

Realizace odčítání potom vychází ze sčítačky, obr. 5-9, před kterou je třeba předřadit kombinační obvod realizující jednotkový doplněk. V případě provádění výpočtu ve dvojkovém doplňku, je možné přičíst +1 prostřednictvím přenosu c_0.

Vzhledem k jednoduchosti řešení se standardně využívá zobrazování čísel ve dvojkovém doplňku, čímž se ušetří čas nutný na přičtení kruhového přenosu.

5.4 Násobení

Máme-li dvě celá čísla *A, B* mezi sebou vynásobit v binární soustavě, postupuje se podle stejného algoritmu jako v soustavě desítkové. Nejdříve se každým bitem čísla *B* vynásobí celé číslo *A* a jednotlivé mezivýsledky postupně sečteme. Vzhledem ke shodnosti výsledků aritmetického součinu a logického součinu je pro vynásobení dvou proměnných volen operátor logického součinu.

ŘEŠENÝ PŘÍKLAD

Vynásobte čísla *A, B* kdy $A = 13$ a $B = 9$.

Řešení příkladu

$A_{10} = 13 \Rightarrow A_2 = 00001101$

$B_{10} = 9 \Rightarrow B_2 = 00001001$

Algoritmus násobení

$$
\begin{array}{rcccccccccl}
A & & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \\
\times B & & & & & & 1 & 0 & 0 & 1 & \\
\hline
 & & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & & & \\
\hline
A \times B = 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \rightarrow 117_{10}
\end{array}
$$

*

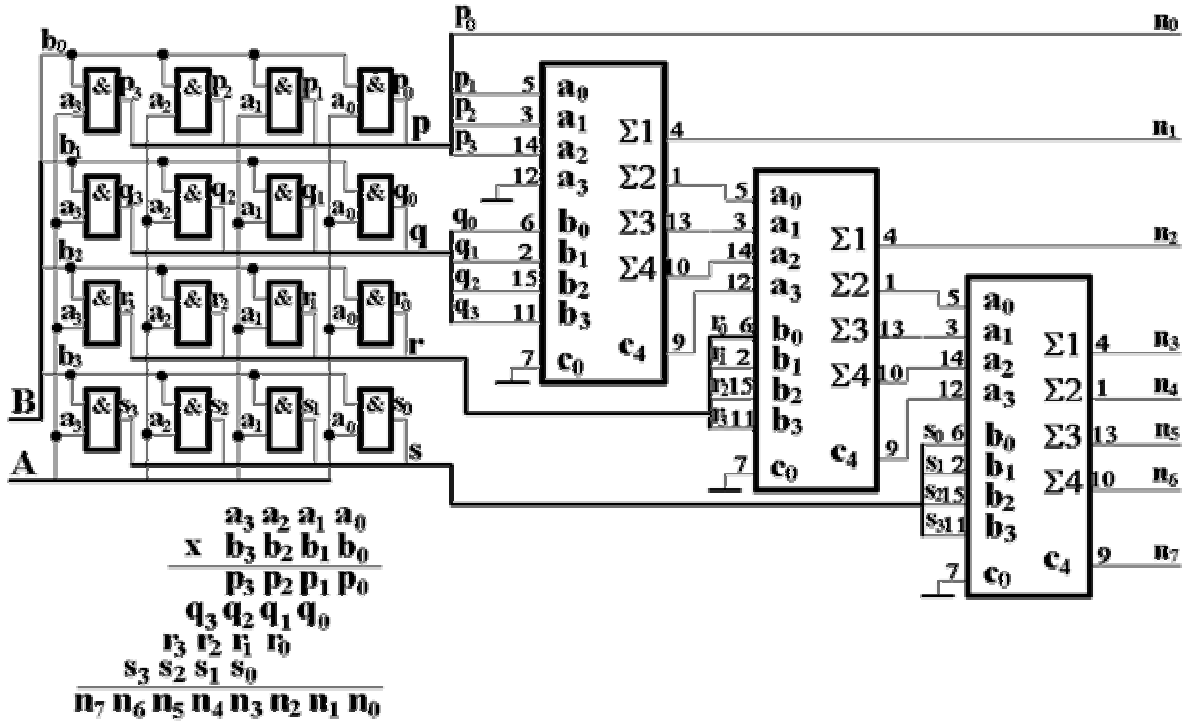
Z příkladu vyplývá, že popsaný algoritmus má obecnou platnost, přestože má dva základní nedostatky.

- Délka slova součinu musí být větší než délka slova činitelů.. Tato skutečnost je řešena jiným způsobem zobrazování čísel než je uvedeno, a to zobrazováním v pohyblivé řádové čárce nebo v dvojnásobné přesnosti.

- Nelze násobit čísla se znaménkem. Apriorně proto třeba stanovit znaménko výsledku a potom čísla násobit jako čísla kladná.

Objasnění těchto principů je však mimo rámec těchto textů.

Realizace 4-bitové násobičky pro celá kladná čísla je uvedena na obr. 5-10.



Obr. 5-10: Logická síť 4-bitové násobičky.

5.5 Dělení

Dělení binárních čísel se provádí podle stejného algoritmu jako dělení v soustavě desítkové. V desítkové soustavě může podíl části dělence a dělitele dosáhnout hodnot 0 až 9, v soustavě binární pouze hodnoty 0 nebo 1. Tím se problém dělení redukuje na zjištění, zda dělitel je větší (výsledek 0) nebo menší (výsledek 1) než část dělence. Algoritmus se nejsnáze realizuje pomocí odčítaček, sčítačky s použitím dvojkového doplňku dělitele, a dále pomocí přepínače, který zajišťuje do dalšího výpočtu přenos rozdílu části dělence a dělitele (rozdíl je kladný) nebo původní části dělence s přidaným bitem z dalšího řádu (rozdíl je záporný).

ŘEŠENÝ PŘÍKLAD

Sestavte algoritmus pro dělení dvou kladných celých čísel *A, B* kdy $A = 54$ a $B = 6$.

Řešení příkladu

$A_2 = 0110110$ $\qquad B_2 = 0110 \Rightarrow {}^2B = 1010$

```
                znaménko        Výsledek
A          0 1 1 0 1 1 0
:²B        1 0 1 0
           ---------------------------
       1 0 0 0 0 1 1 0          kladný → 1
:²B          1 0 1 0
           ---------------------------
           1 0 1 1 1 0          záporný → 0
znaménko
           0 0 0 1 1 0
:²B          1 0 1 0
           ---------------------------
             1 1 0 1 0          záporný → 0
znaménko
           0 0 0 1 1 0
:²B          1 0 1 0
           ---------------------------
             1 0 0 0 0          kladný → 1
                 zbytek
```

algoritmus dělení

$D_2 = 1001 \Rightarrow 9_{10}$

5.6 Porovnávání

Jestliže máme dvě n-bitová čísla A a B můžeme pro porovnávání jejich logických hodnot definovat minimálně šest logických funkcí, a to:

$$f_1 : A = B, \quad f_2 : A \neq B, \quad f_3 : A > B, \qquad (5\text{-}20)$$
$$f_4 : A \geq B, \quad f_5 : A < B, \quad f_6 : A \leq B.$$

Porovnáváním těchto funkcí zjistíme, že platí

$$f_1 = \overline{f_2}, \quad f_3 = \overline{f_6}, \quad f_4 = \overline{f_5} \qquad (5\text{-}21)$$

a pro logický součin funkcí $\overline{f_4}$ *a* $\overline{f_6}$ můžeme psát

$$f_4 \cdot f_6 = f_1 = \overline{f_5} \cdot \overline{f_3} \qquad (5\text{-}22)$$

Z vztahu *(5-22)* vyplývá, že k realizaci všech funkcí podle *(5-20)* nám postačuje realizovat pouze dvě funkce, a to buď f_4 a f_6 nebo f_3 a f_5. Z těchto funkcí lze potom generovat zbývající.

Jako příklad je uveden návrh obvodu pro porovnávání 2-bitových proměnných $A(a_1, a_0)$ a $B(b_1, b_0)$. Blokové schéma navrhovaného obvodu je na obrázku 5-11.

Jestliže jako základní funkce zvolíme funkce f_3 a f_5, můžeme sestavit pro řešený kombinační obvod pravdivostní tabulku viz. obr. 5-12.

Obr. 5-11: Blokové schéma obvodů porovnávání.

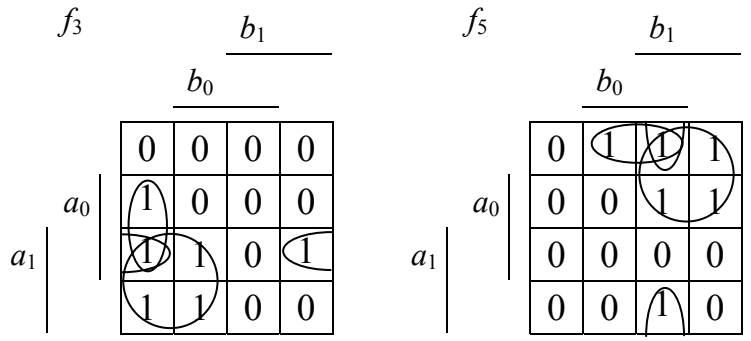
b_1	b_1	a_1	a_0	f_3	f_5
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	0

Obr. 5-12: Pravdivostní tabulka obvodu porovnávání

Z pravdivostní tabulky sestavíme pro proměnné funkcí f_3 a f_5 Karnaughovy mapy, viz. obr. 5-13, a funkce vyjádříme v součtovém tvaru:

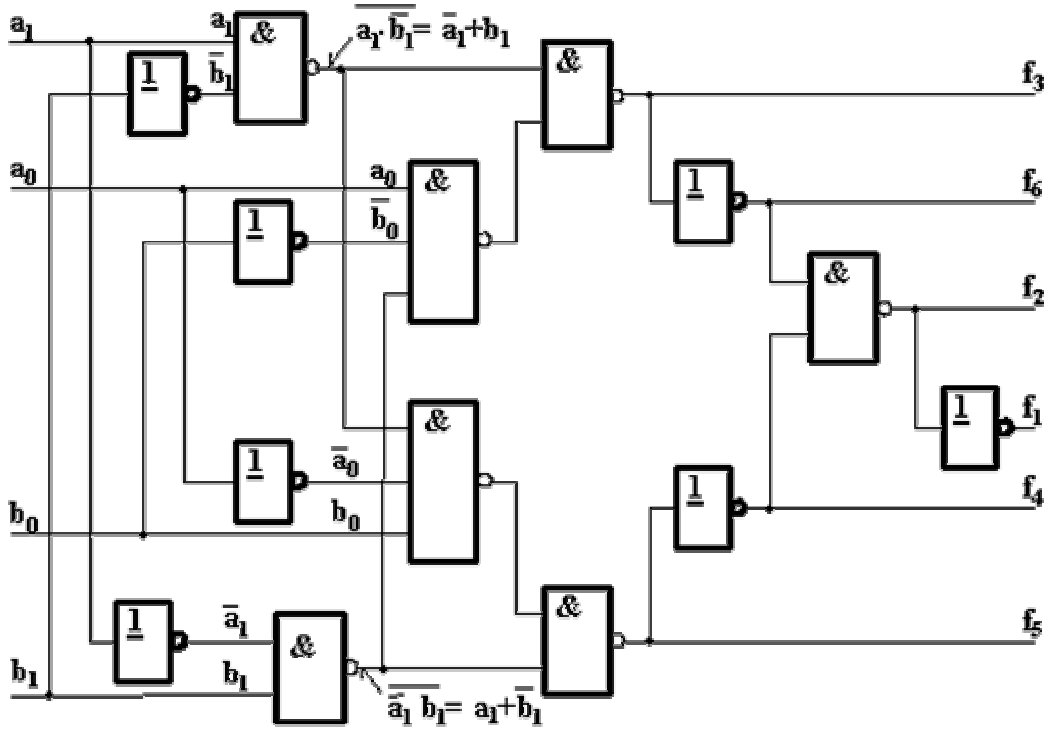
$$f_3 = a_1\overline{b_1} + a_0\overline{b_0}\,\overline{b_1} + a_0 a_1\overline{b_0} = a_1\overline{b_1} + a_0\overline{b_0}\left(a_1 + \overline{b_1}\right) \qquad (5\text{-}23)$$

$$f_5 = \overline{a_1}b_1 + \overline{a_0}\,\overline{a_1}b_0 + \overline{a_0}b_0 b_1 = \overline{a_1}b_1 + \overline{a_0}b_0\left(\overline{a_1} + b_1\right) \qquad (5\text{-}24)$$



Obr. 5-13: Karnaughova mapa pro obvod porovnávání

Komplexní realizace obvodu porovnávání dvou 2-bitových čísel pomocí hradel NAND je na obr. 5-14



Obr. 5-14: Blokové schéma obvodů porovnávání.

5.7 Zobrazování čísel v pohyblivé řádové čárce

Chceme-li pracovat s čísly v pohyblivé desetinné čárce, musíme je rozšířit o číslo vyjadřující hodnotu exponentu včetně jeho znaménka. Číslo s pohyblivou desetinnou čárkou pak bývá vyjádřeno ve tvaru

$$A = 0 \quad A = m \cdot B^{e} \text{ nebo } A = (m \cdot B) \cdot B^{e-1} \qquad (5\text{-}25)$$

Kde m je mantisa (část za desetinnou čárkou), B je základ číselné soustavy a e je hodnota exponentu. Pro ilustraci číslo 6725430, pak bude vyjádřeno buď jako $0{,}672543.10^{7}$ nebo $6{,}725430.10^{6}$. Pro dvojkovou soustavu, kterou se budeme zabývat, můžeme ve shodě s vyjádřením *(5-25)* rovnice psát

$$A = 0 \qquad A = \text{znaménko}\left(\tfrac{1}{0} + \text{mantisa}\right) \cdot 2^{\text{zanménko exponentu}} \qquad (5\text{-}26)$$

Mantisa představuje hodnotu na desetinných místech a můžeme ji v souladu s předcházejícím textem vyjádřit výrazem

$$\text{mantisa} = a_{-1}B^{-1} + a_{-2}B^{-2} + \ldots + a_{-n+1}B^{-n+1} + a_{-n}B^{-n} \qquad (5\text{-}27)$$

Závorka v rovnici *(5-26)* může nabývat hodnoty v intervalu $\langle 1.0;2.0)$ nebo $\langle 0{,}5;1{,}0)$ podle toho, je-li k mantise připočítávána hodnota 1,0 či nikoliv. Obě vyjádření se od sebe liší o jedničku v hodnotě exponentu. Přesnost čísla na desetinných místech je dána hodnotou 2^{-n}.

Hodnotu exponentu vyjádříme celým číslem se znaménkem

$$\text{znaménko exponentu} = z_e a_{m-2} a_{m-3} \ldots a_0 \qquad (5\text{-}28)$$

ŘEŠENÝ PŘÍKLAD

Určete počet bitů nezbytný k vyjádření čísel v pohyblivé desetinné čárce s přesností tří desetinných míst a exponentem v rozsahu $10^{\pm 19}$.

Řešení příkladu

Aby číslo v pohyblivé čárce mělo přesnost tří desetinných míst, musí hodnota n splňovat vztah

$$2^{-n} \le 10^{-3} \qquad n \ge \frac{3}{\log 2} \ge 9{,}92 \tag{5-29}$$

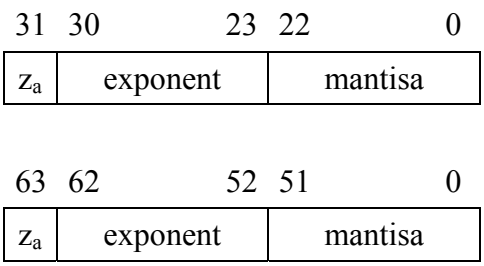
Odtud n=10. Nezbytný počet bitů m k vyjádření exponentu určíme ze vztahu

$$2^{2^{m-1}} \ge 10^{19} \qquad m \ge 1 + \frac{\log\left(\frac{19}{\log 2}\right)}{\log 2} \ge 6{,}97 \tag{5-30}$$

Odtud m = 7. Celé číslo A bude vyjádřeno včetně znaménka 18 bity – symboly, které jsou zobrazeny následujícím vztahem *(5-31)*.

*

$$\underset{\text{znaménko}}{z_a} \quad \underset{\text{mantisa}}{a_{-1}a_{-2}a_{-3}a_{-4}a_{-5}a_{-6}a_{-7}a_{-8}a_{-9}a_{-10}} \quad \underset{\text{exponent}}{z_e a_5 a_4 a_3 a_2 a_1 a_0} \tag{5-31}$$



Obr. 5-15:

Standard pro vyjádření čísel v pohyblivé desetinné čárce v jednoduché i dvojnásobné přesnosti je dána předpisem IEEE 754-1985. Čísla s jednoduchou přesností jsou vyjádřena 32 bity ve tvaru $(-1)^z \cdot (1+m) \cdot 2^{e+127}$, kde bity z a m představují číslo ve vyjádření $\pm$ absolutní hodnota a exponent $e \in <-126;127>$ je ve vyjádření $+1/2$ intervalu zmenšený o hodnotu jedna. Krajní hodnoty exponentu e+127=0 a e+127=255 se využívají k vyjádření čísel, která nelze ve formátu daném rovnicí *(5-26)* zobrazit. Při dvojnásobné přesnosti jsou čísla vyjádřena stejně jako při přesnosti jednoduché a tím, že k exponentu je přičítána hodnota 1023 $e \in <-1022;1023>$, jak vyplývá z obr. 5-15.

Oba formáty ve svém instrukčním souboru podporuje například signálový procesor s pohyblivou čárkou MOTOROLA DSP 96002 s tím, že je využito takzvaného skrytého bitu. To znamená, že v bitovém vyjádření výrazu (1+*m*) nenalezneme právě tu jedničku. V aritmetických knihovnách, které využíváme někdy v programech psaných v jazyce symbolických adres, se setkáváme s vyjádřením čísel, které se více či méně blíží tomuto standardu.

Například v aritmetikách INTEL pro μP 8080 nebo FL48 a FL51 – TESLA IMA pro μP 8048 a 8051 je využíváno vyjádření (1+*m*) ve tvaru $\pm$ absolutní hodnota se skrytým bitem (skrytou jedničkou) a exponentem je vyjádření +1/2 intervalu viz tabulka na obr. 5-16. Dojde-li při aritmetické operaci k přetečení nebo podtečení potom je situace indikována příznakem CY (8080) nebo F0 (8048,8051) a nikoliv maximální nebo minimální hodnotou exponentu. Nula je charakterizována nulovým exponentem a záporná nula je zakázána. Naproti tomu u signálových procesorů s pohyblivou desetinnou čárkou Texas instruments TMS320C3x a TMS320C4x, jejichž aritmeticko-logické jednotky podporují operace se zápornými čísly vyjádřené dvojkovým doplňkem, mají vyjádřenou hodnotu (1+*m*) i exponent ve vyjádření dvojkovým doplňkem.

Číslo		Vyjádření		
		z	mantisa	exponent
$1{,}000.10_0$	$1{,}000000.2_0$	0	0000000000	1000000
$8{,}000.10_0$	$1{,}000000.2_3$	0	0000000000	1000011
$0{,}000.10_0$	$0{,}000000.2_0$	0	0000000000	0000000
$-1{,}750.10_0$	$1{,}750000.2_0$	1	1100000000	1000000
$1{,}250.10_2$	$1{,}953125.2_0$	0	1111010000	1000110
$-1{,}250.10_2$	$-1{,}953125.2_6$	1	1111010000	1000110
$-7{,}812.10_{-2}$	$-1{,}249920.2_{-4}$	1	0011111111	011110

Obr. 5-16:

SHRNUTÍ LOGICKÝCH OBVODŮ

Shrnutí modulu

Po prostudování těchto textů, úspěšném vypracování a zapojení samostatných prací jste schopni navrhnout a realizovat zapojení kombinačních a sekvenčních logických obvodů. Získali jste základní znalosti o nejběžnějších integrovaných obvodech a paměťových prvcích, umíte analyzovat zapojení s logickými obvody a víte, jak pracuje aritmeticko-logická jednotka.

KLÍČOVÁ SLOVA LOGICKÝCH OBVODŮ

Klíčová slova

Binární soustava, oktální soustava, hexadecimální soustava, dekadická soustava, číselná soustava, koeficient, váha, základ, bit, polynom, logická proměnná, funkce rovnosti, logické operátory, logický součin, logický součet, negace (komplement), asociativní zákon, komutativní zákon, distributivní zákon, zákon absorpce, De Morganovy zákony, zákon absorpce konsensu, booleovská funkce, minimalizace, algebraická minimalizace, Mc-Cluskey, Karnaughova mapa, logický kombinační obvod, integrovaný obvod, obvody TTL, AND, NOR, AND-OR-INVERT, paměť ROM, statický hazard, dynamický hazard, souběhový hazard, analýza obvodu, asynchronní sekvenční obvod, synchronní sekvenční obvod, synchronizační signál, Mealyho typ, Moorův typ, paměťový člen RS, paměťový člen RST, paměťový člen D, paměťový člen JK, vývojový diagram, zpožďovací člen, paměťový registr, posuvný registr, synchronní čítač, asynchronní čítač, dělička číslicového signálu, generátor binární posloupnosti, aritmeticko - logická jednotka, sčítání, odčítání, násobení, dělení, porovnávání, jednotkový doplněk, dvojkový doplněk, sčítačka, násobička, mantisa, exponent.

DOPLŇUJÍCÍ ZDROJE

DIVIŠ, Zdeněk, CHMELÍKOVÁ, Zdeňka, ZDRÁLEK, Jaroslav: Logické obvody, Ostrava, 2000.

PETŘÍKOVÁ, Iva: Logické obvody – příklady, Ostrava 2001

KLÍČ K ŘEŠENÍ

Ú 1-1

a) $N_{10} = 444$

b) $N_{10} = 912$

c) $N_{10} = 999$

Ú 1-2

a) ${}^1N_2 = 111001010$, ${}^2N_2 = 10000101$, ${}^3N_2 = 1011111$

b) ${}^1N_8 = 1620$, ${}^2N_8 = 400$, ${}^3N_8 = 1144$

c) ${}^1N_{16} = 458$, ${}^2N_{16} = 34F$, ${}^3N_{16} = 19C$

Ú 1-3

a) ${}^1N_2 = 1001111$, ${}^2N_2 = 1101111$, ${}^3N_2 = 11111001$

b) ${}^1N_8 = 1602$, ${}^2N_8 = 251$ a ${}^3N_8 = 2350$

c) ${}^1N_{16} = 4DE$, ${}^2N_{16} = 7C6$ a ${}^3N_8 = 7AB$

Ú 1-4

a) ${}^1N_2 = 0{,}011000$, ${}^2N_2 = 0{,}000001$, ${}^3N_2 = 0{,}011010$

b) ${}^1N_8 = 0{,}076$, ${}^2N_8 = 0{,}223$ a ${}^3N_8 = 0{,}521$

c) ${}^1N_{16} = 0{,}49B$, ${}^2N_{16} = 0{,}456$ a ${}^3N_{16} = 0{,}65E$

Ú 1-5

a) $N_8 = 674$

b) $N_2 = 111101100$

c) $N_{16} = 2F9$

d) $N_2 = 11000101$

Ú 2-1

a) $\overline{x_2} \cdot x_1 + \overline{x_0}$

b) $x_2 \cdot (x_1 + x_0)$

c) $x_2 \cdot \overline{x_1} + x_1 \cdot x_0 + \overline{x_2} \cdot \overline{x_0}$

d) x_3

e) $\overline{x_2} \cdot \overline{x_0} + x_2 \cdot x_0$

f) $x_1 \cdot x_0 + x_2 \cdot x_1 + x_2 \cdot x_0$

g) x_2

h) $x_3 \cdot x_1$

Ú 2-2 $\overline{\left(\overline{\overline{x_2} + x_0}\right) + \left(\overline{x_1 + x_0}\right) + \left(\overline{\overline{x_1} + \overline{x_0}}\right)}$

Ú 2-4

a) $F_1^1 = \left(\overline{x}_2 \cdot x_0 + x_1 \cdot \overline{x}_0 + x_2 \cdot \overline{x}_1\right)$, $F_1^2 = \left(\overline{x}_1 \cdot x_0 + \overline{x}_2 \cdot x_1 + x_2 \cdot \overline{x}_0\right)$,

$F_1^3 = \left(x_2 + x_1 + x_0\right) \cdot \left(\overline{x}_2 + \overline{x}_1 + \overline{x}_0\right)$

b) $F_2^1 = \left(x_3 \cdot x_1 + \overline{x}_3 \cdot \overline{x}_1 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_0\right)$,

$F_2^2 = \left(x_3 + \overline{x}_1\right) \cdot \left(\overline{x}_3 + \overline{x}_2 + x_1\right) \cdot \left(\overline{x}_3 + x_1 + \overline{x}_0\right)$

Ú 2-5

$x_0 \backslash x_1 x_2$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$f = x_0x_1 + x_0x_2 + x_1x_2$

Ú 2-6

$x_3 x_0 \backslash x_1 x_2$	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	0	0	1	1
10	1	0	1	1

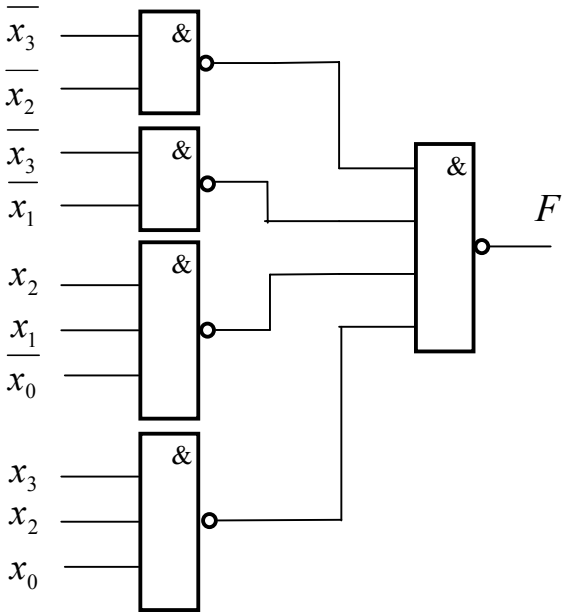
$f = \left(x_3 + \overline{x_2} + \overline{x_1}\right) \cdot \left(x_3 + \overline{x_2} + \overline{x_0}\right) \cdot \left(\overline{x_3} + x_2 + \overline{x_1}\right) \cdot \left(\overline{x_3} + x_2 + \overline{x_0}\right)$

Ú 2-7 $F = (x_3 + x_1) \cdot (x_3 + x_2) \cdot (\overline{x_3} + x_0)$

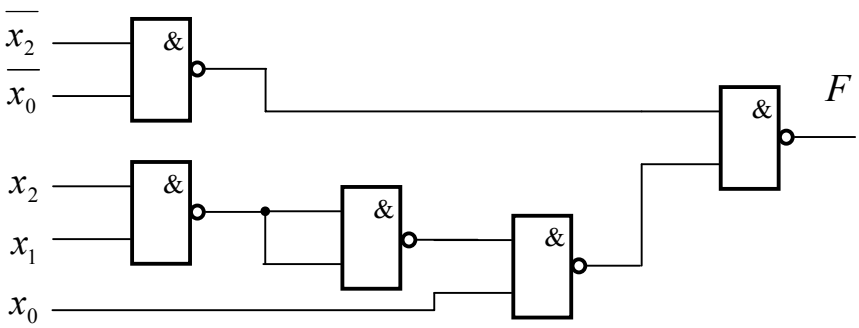
Ú 2-8 ${}^1F = (x_1 \cdot x_0) + (\overline{x_3} \cdot x_1) + (x_3 \cdot \overline{x_1} \cdot \overline{x_0}) + (x_2 \cdot \overline{x_1} \cdot \overline{x_0})$

${}^2F = (x_1 \cdot x_0) + (\overline{x_3} \cdot x_1) + (x_3 \cdot \overline{x_1} \cdot \overline{x_0}) + (\overline{x_3} \cdot x_2 \cdot \overline{x_0})$

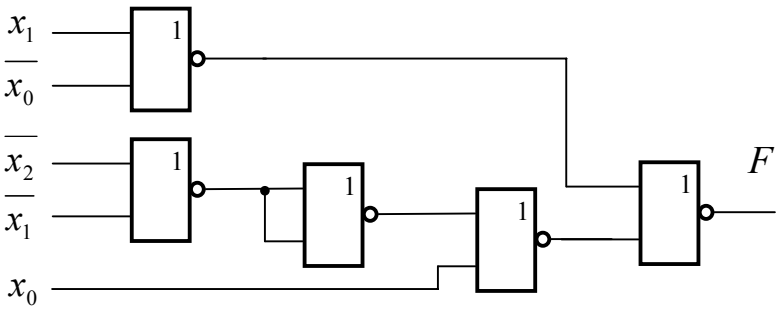
Ú 3-1


$\overline{x_3}$
$\overline{x_2}$
$\overline{x_3}$
$\overline{x_1}$
x_2
x_1
$\overline{x_0}$
x_3
x_2
x_0
&
&
&
&
&
F

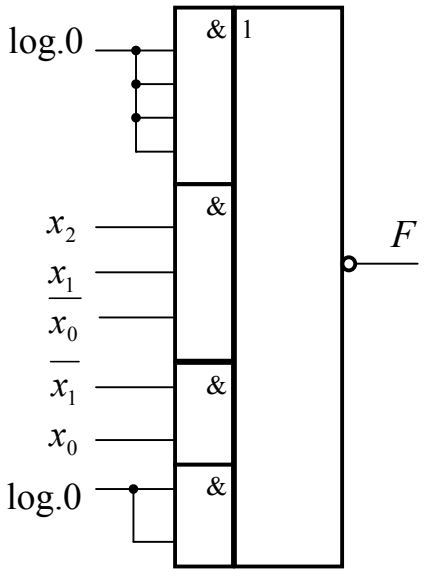
Ú 3-2


$\overline{x_2}$
$\overline{x_0}$
x_2
x_1
x_0
&
&
&
&
&
F

Ú 3-3


x_1
$\overline{x_0}$
$\overline{x_2}$
$\overline{x_1}$
x_0
1
1
1
1
1
F

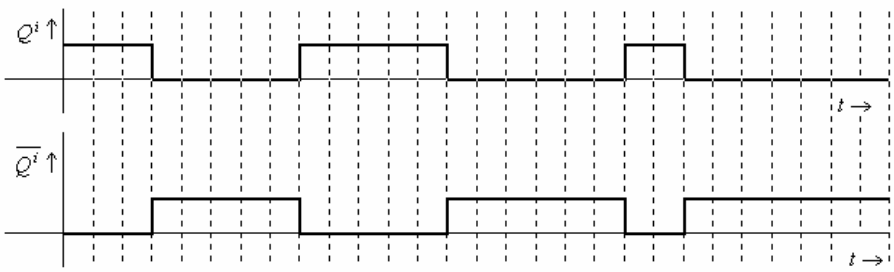
Ú 3-4



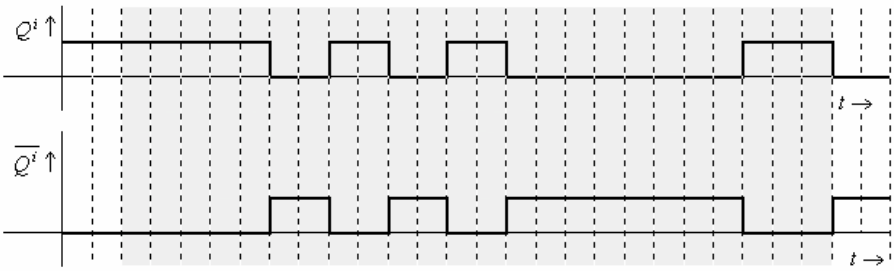
Ú 3-5 $F_{11} = \overline{x_1} \cdot \overline{x_0} + x_1 \cdot x_0 + \overline{x_2} \cdot \overline{x_0} + \overline{x_2} \cdot x_1$

$F_{11} = (x_1 + \overline{x_0}) \cdot (\overline{x_2} + \overline{x_1} + x_0)$ Funkce je bez hazardu.

Ú 4-1



Ú 4-2



O 1

a) Pro realizaci funkce pomocí hradel NAND se používá minimální součtový tvar, který použitím De Morganových zákonů upravíme na tvar obsahující pouze operaci logický součin.

b) Pro realizaci funkce pomocí hradel NOR se používá minimální součinový tvar, který použitím De Morganových zákonů upravíme na tvar obsahující pouze operaci logický součet.

O 2

a) Zpožďovací člen lze realizovat pomocí zpožďovacích linek, paměťovými členy nebo zpožděním v logických členech.

b) Sekvenční obvody na základě použitého typu zpožďovacího členu dělíme na synchronní a asynchronní.

c) Rozeznáváme základní způsob a impulsní způsob řízení asynchronních obvodů.

d) Podmínkou je konstantní vstupní signál, případně signály.

e) Funkce přechodů mohou být reprezentovány algebraickým výrazem, tabelárním vyjádřením, grafickým vyjádřením, časovým diagramem, vývojovým diagramem a programem činnosti.