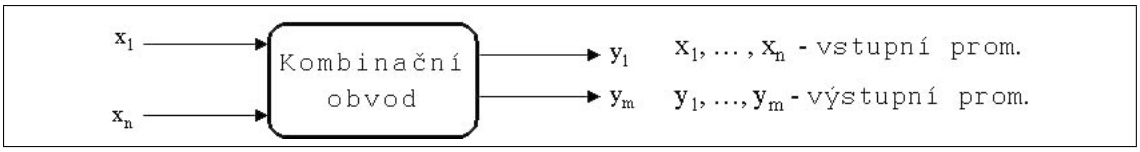
Kapitola 2

Logické systémy a jejich návrh

2.1 Logické funkce a jejich návrh

Vstupní/výstupní písmeno - každá kombinace hodnot všech vstupních/výstupních proměnných. Na vstup se tedy může přivést jedno z 2^n možných vstupních písmen.
Kombinační systém realizuje zobrazení $\Phi_k : X \to Y$, kde každému vstupnímu písmenu $X_i \in X$ přiřazuje výstupní písmeno $Y_i \in Y$. Zobrazení nemůže být libovolné, ale musí splňovat podmínku, že každému vstupnímu písmenu X_i odpovídá v zobrazení Φ_k pouze jedno výstupní písmenu $Y_i \Rightarrow$ *logická funkce*. Pro n proměnných můžeme nadefinovat celkem 2^{2^n} různých logických funkcí, např.



Obrázek 2.1: kombinační obvod

pro dvě vstupní proměnné je $2^{2^2} = 16$ různých logických funkcí.
Logické funkce můžeme popsat pomocí:

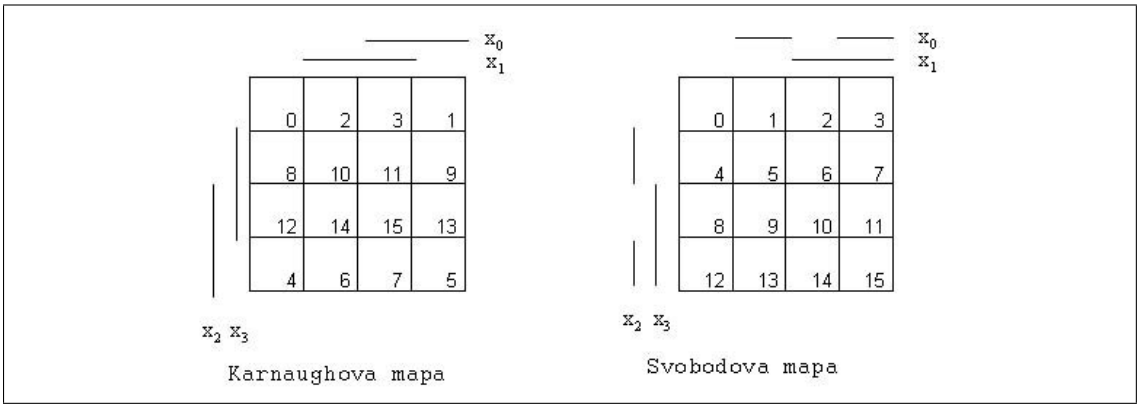
- pravdivostní tabulky
- logickým výrazem
- seznamem indexů vstupních písmen
- mapou

2.2 Úplné a minimální formy

Minterm - součinový term $(x_1.\overline{x}_2 \dots x_n)$, který obsahuje všechny vstupní proměnné.
Maxterm - součtový term $(x_1 + \overline{x}_2 + \dots + x_n)$, který obsahuje všechny vstupní proměnné.
Každý *minterm/maxterm* nabývá logické hodnoty 1/0 právě pro jediné vstupní písmeno $\Rightarrow$ konstrukce booleovského výrazu pro logickou funkci *f*.
Disjunktní forma (DF) - výraz ve tvaru součtu součinů termů $(abc + a\overline{b}c + ab\overline{c})$
Konjunktivní forma (KF) - výraz ve tvaru součinu součtových termů $((a+b+c)(a+\overline{b}+c)(a+b+\overline{c}))$
Úplná normální DF - výraz ve tvaru součtu *mintermů*.
Úplná normální KF - výraz ve tvaru součinu *maxtermů*.

Obě úplné normální formy reprezentují velice často neúsporný výraz, který jen v některých případech nejde zjednodušit. Pro zjednodušení můžeme použít de Morganovy zákony, mapy a metodu podle Quina.

de Morganovy zakony $\begin{cases} \overline{x_1.x_2} & = & \overline{x}_1 + \overline{x}_2 \\ \overline{x_1 + x_2} & = & \overline{x}_1.\overline{x}_2 \end{cases}$



Obrázek 2.2: Karnaughova a Svobodova mapa

2.3 Minimalizace podle map a Quina

Cílem minimalizace je nalézt co nejjednodušší výraz pro původní (zadanou) logickou funkci. Jde tedy o získání výrazu, který umožňuje realizovat obvod s nejmenším počtem základních členů vyjadřujících co nejmenší počet vstupů.

2.3.1 Minimalizace podle map

Implikant F logické funkce *f* je výraz ve tvaru součinového termu, pro který platí, že implikuje funkci *f*, tzn. jestliže nabývá hodnoty logické 1, funkce *f* nabývá také logické hodnoty 1.

Každá jednička v mapě reprezentuje určitý implikant, který je zároveň mintermem v *ÚDNF*. Vzájemná poloha jedniček v mapě pak nese informaci o případném výskytu sousedních implikantů. Tím se vyhledávají množiny přímých implikantů. Problém jejich vyhledávání děláme pomocí minimalizačních smyček.

Minimalizační smyčka - taková podoblast mapy, která obsahuje 2^k vnitřních políček ($k = 0, 1, 2, \ldots$), přičemž každému políčku přísluší v této smyčce právě k sousedních políček.

Při minimalizaci se snažíme získat co nejmenší počet co největších minimalizačních smyček, které zahrnují všechny jedničky v mapě. Tyto minimalizační smyčky popisujeme součinovými termy. Minimální *DF* pak získáme vytvořením součtu součinových termů, které odpovídají vybraným smyčkám.

Hledání minimální *KF* se děje stejným způsobem, ale pokrýváme pouze políčka s nulou. Výsledek popisujeme pomocí součtových termů.

K vytvoření minimální formy se použijí ty proměnné, které pro celou skupinu nemění svoji hodnotu.

Pro mapy z obrázku 2.3 postupně dostaneme:

a) $f = x_3 + x_1x_2 + x_0x_2$

b) $f = (x_2 + x_3)(x_0 + x_1 + x_3)$

c) $f = \overline{x_0}x_1 + \overline{x_1}x_2 + x_0\overline{x_1}x_2$

V případě, že se jedná o neúplně definovanou funkci:

1. vytvoříme smyčky, které zahrnují jedničky i neurčené hodnoty X (hodnoty X považujeme v tomto kroku za jedničky).
2. při výsledném výběru minimalizačních smyček se omezíme pouze na ty nejúspornější, které zahrnují všechny jedničky. Stavy X, které se v těchto smyčkách nevyskytují, považujeme nyní za nuly.

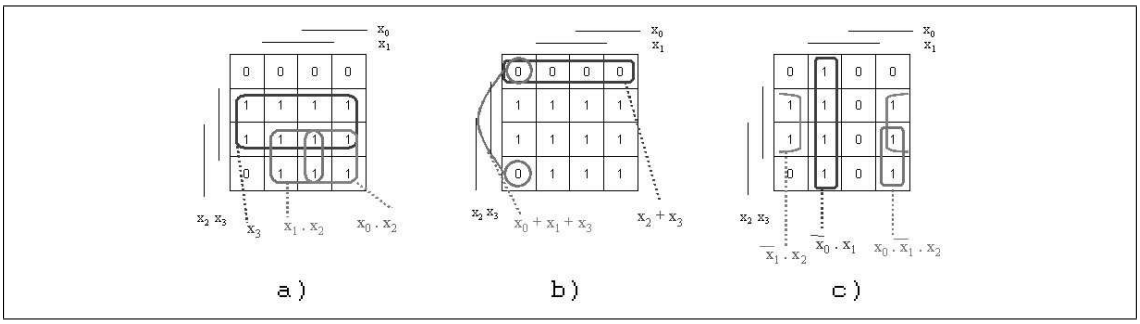
2.3.2 Minimalizace podle Quina-Mc-Cluskey

Používá se pro minimalizaci logických funkcí většího počtu proměnných.
Pro pochopení následujících kroků doporučuji pročíst kapitolu v Logických systémech (str. 47)
Základní kroky jsou:

1. nalezení množiny přímých implikantů
2. výběr minimálního počtu implikantů z množiny nalezené v kroku 1

ad 1.
- nalezneme všechny mintermy, které odpovídají zadaným indexům vstupních písmen
- nalezenou množinu rozdělíme na takové podmnožiny, ve kterých mají mintermy stejný počet přímých i negovaných proměnných
- hledáme sousední mintermy - porovnáváme pouze mintermy z těch podmonžin, ve kterých se počet negovaných proměnných liší o jedničku. V případě nalezení sousedních termů tyto termy nahradíme termem jednodušším. Tento postup opakujeme do té doby, dokud již neexistují žádné sousední termy, tj. není možné žádné další zjednodušení
- případné duplicitní termy přeškrtneme

ad 2.
- sestrojíme tabulku pokrytí, kde každý řádek odpovídá jednomu přímému implikantu. Sloupce reprezentují původní mintermy ÚDNF nebo také vstupní písmena, pro která zadaná funkce nabývá jedničky. Ve sloupcích, které odpovídají indexům původních mintermů, ze kterých následně vznikly přímé implikanty, si uděláme křížek. Cílem tohoto kroku je pokrytí všech sloupců co nejmenším počtem řádků. Začínáme u sloupců obsahující jediný křížek. Následně přeškrtneme všechny křížky tohoto řádku i v ostatních sloupcích a také křížky sloupců, které se nacházejí v jiných řádcích
- postupujeme do pokrytí všech sloupců



Obrázek 2.3: minimalizační smyčky

2.4 Návrh kombinačních logických obvodů

Kombinační systém - odezva v určitém časovém okamžiku je podmíněna pouze těmi hodnotami, které jsou v daném okamžiku na vstupech kombinačního obvodu.
Návrh kombinačního systému:

1. přesný popis chování kombinačního systému a případné sjednocení tohoto popisu
2. obvodová realizace kombinačního systému

Pokud máme k dispozici základní logické členy typu negátor, logický součet, logický součin, negovaný logický součet, negovaný logický součin, můžeme realizovat logický kombinační obvod dané funkce bez úprav na ÚDNF nebo ÚKNF. V podstatě nám stačí pouze logický součet/součin a negátor.

K realizaci součtové normální formy slouží tříúrovňová logická síť. Výstupy jednoho stupně jsou připojeny na vstupy dalšího stupně.

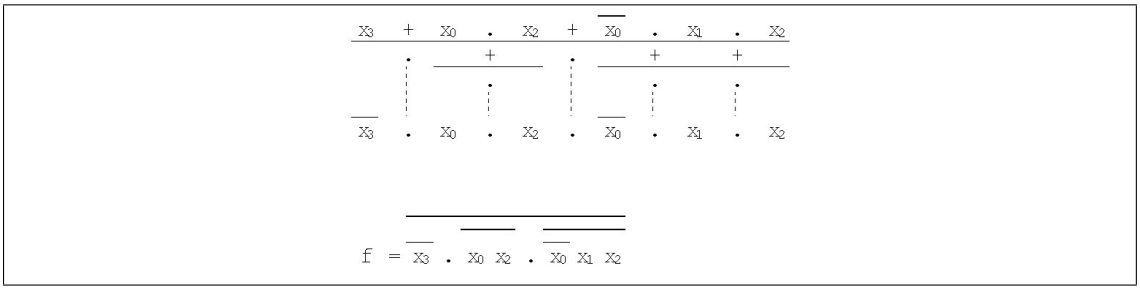
Rychlost kombinačního obvodu je dána zpožděním jednotlivých členů a počtem úrovní. Mnoho úrovní ⇒ větší zpoždění.

2.5 Realizace kombinačních logických obvodů z členů NAND a NOR

Převod booleovské funkce v minimálním tvaru je založen na využití De Morganových zákonů:
$\overline{a+b+c\ldots} = \overline{a}\cdot\overline{b}\cdot\overline{c}\ldots$
$\overline{a\cdot b\cdot c\ldots} = \overline{a}+\overline{b}+\overline{c}\ldots$

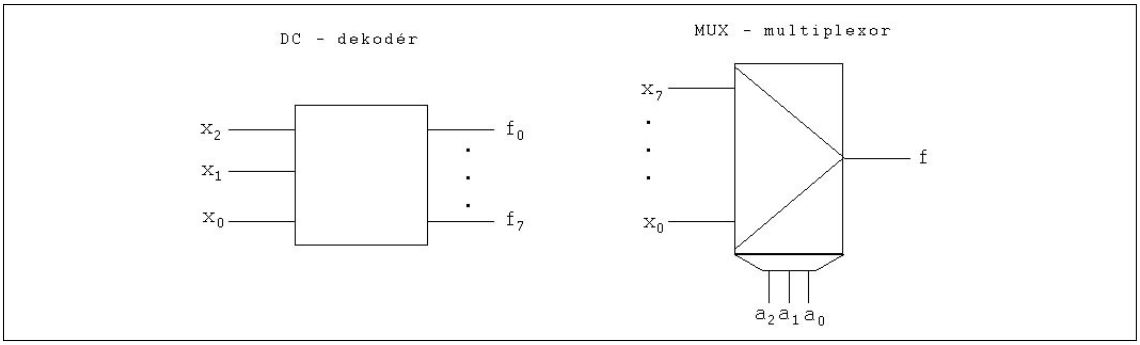
Chceme-li vytvořit síť se členy NAND, je výhodné vycházet z disjunktní normální formy a pro síť se členy NOR z formy konjunktivní. V případě, že máme omezený počet vstupů u použitých hradel, musíme implikanty, které překračují povolenou délku rozložit na kratší podle De Morganových pravidel.
Místo použití De Morganových pravidel můžeme použít Rottovy mřížky - obrázek 2.4:

- střídání symbolů operací log. součtu a součinu je vyjádřením De Morganových pravidel
- svislé čáry oddělují jednotlivé použité vstupy
- mřížka se vytváří odshora dolu a do posledního řádku opisujeme z výchozího výrazu jednotlivé vstupní proměnné v přímé nebo negované formě
- také v těchto mřížkách můžeme respektovat počet vstupů



Obrázek 2.4: Rottovy mřížky

2.6 Realizace s univerzálními moduly (DC, MUX)



Obrázek 2.5: DC & MUX

DC

Pokud se použije pozitivní logika, pak má DC následující vlastnosti:

- v každém okamžiku nabývá právě jedna výstupní proměnná 1, ostatní nabývají 0
- volba proměnné s hodnotou 1 je dána vstupním písmenem

MUX

- adresové vstupy $(a_0 \dots a_2)$ určují výběr jednoho ze vstupů $x_0 \dots x_7$, který má být přepnut na výstup f.

2.6.1 Použití multiplexorů

Vstupní písmeno X_i přivedeme na adresové vodiče. Každému vstupnímu písmenu je jednoznačně přiřazen určitý vstupní vodič, tj. na tento vstupní vodič se připojí přímo ta výstupní hodnota, která je dle realizované logické funkce přiřazena danému vstupnímu písmenu.

2.6.2 Použití dekodérů

Dekodér z binárního kódu do kódu *1 z N*. Aktivovaná výstupní proměnná dekodéru vždy jednoznačně určuje existující vstupní písmeno.

Kombinační obvod se realizuje tak, že se vytvoří logický součet hodnot výstupních proměnných odpovídajících těm vstupním písmenům, které produkují jedničkovou hodnotu logické funkce.

2.7 Hazardy v kombinačních systémech

V důsledku toho, že všechny logické členy vykazují jistá zpoždění a v důsledku existence nestejně dlouhých paralelních větví kombinačního systému, vznikají na jeho výstupech *přechodové děje*, které reprezentují *přechodné chyby*.

Všechny dynamické chyby se nedají bezezbytku eliminovat, např. se nemůže nastavit přesně stejná zpoždění ve všech větvích kombinačního obvodu, což se projevuje nesoučasnými změnami hodnot výstupních proměnných.

Funkční hazardy

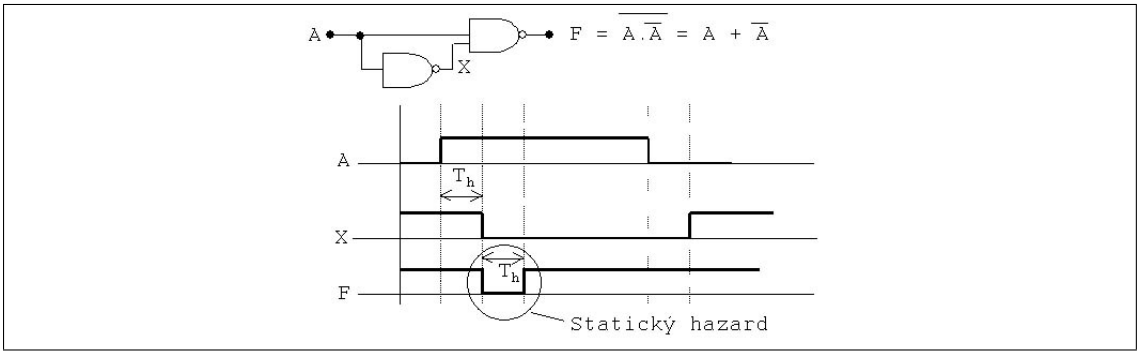
- správné odezvy na přechodně nesprávná vstupní písmena. Jejich odstranění by bylo možné pomocí filtru, který by „nepustil“ žádné krátkodobé změny hodnot výstupních proměnných.

Logické hazardy

- předpokladem je, že každá změna na vstupu se týká hodnoty jediné vstupní proměnné, tj. každá dvě po sobě následující vstupní písmena musí být sousední. Tím se eliminují funkční hazardy, ale neodstraňují se *přechodné chyby.*

A. Statické hazardy

Ke statickým hazardům dochází tehdy, jestliže při změně vstupní proměnné a stabilním stavu ostatních vstupních proměnných, dojde ke krátkodobé změně výstupní proměnné, i když vzhledem k platnosti základních zákonů Booleovy algebry k této změně dojít nemělo. Normálně by mělo

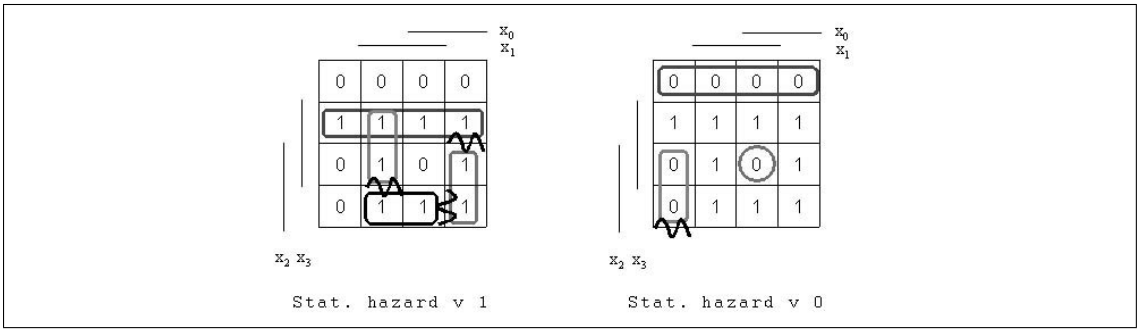


Obrázek 2.6: statický hazard

platit podle funkce z obrázku 2.6 $F = A + \overline{A} = 1$, tj. na výstupu vždy log. 1. Ale na výstupu se na dobu T_h objeví log. 0. Vlivem zpoždění invertorem X se zpozdila změna ze vstupu A.

Identifikaci statického hazardu v log. 1 můžeme provést pomocí časového diagramu(viz. obrázek 2.6) nebo mapy.

Nechť máme minimalizační smyčky v mapě. Zkoumáme pouze ty dvojice políček, které reprezentují sousední vstupní písmena a obě obsahují jedničky (pro statický hazard v log. 1) případně dvojice obsahující nuly (pro statický hazard v log. 0). Pokud tyto dvě jedničky/nuly nejsou zahrnuty v jediné smyčce, pak je příslušný hazard doprovázen hazardem v jedničce/nule - viz. obrázek 2.7.



Obrázek 2.7: statický hazard identifikovatelný mapou

Statický hazard v log. 1 je průvodním jevem pouze pro změnu hodnoty $x_i : 1 \rightarrow 0$.
Odstranění logického hazardu v 1:

- kompenzace delšího zpoždění vkládáním zpožďovacích členů do rychlejších větví obvodu

- vložení redundantních členů, tj. porušíme podmínku minimality

B. Dynamické hazardy

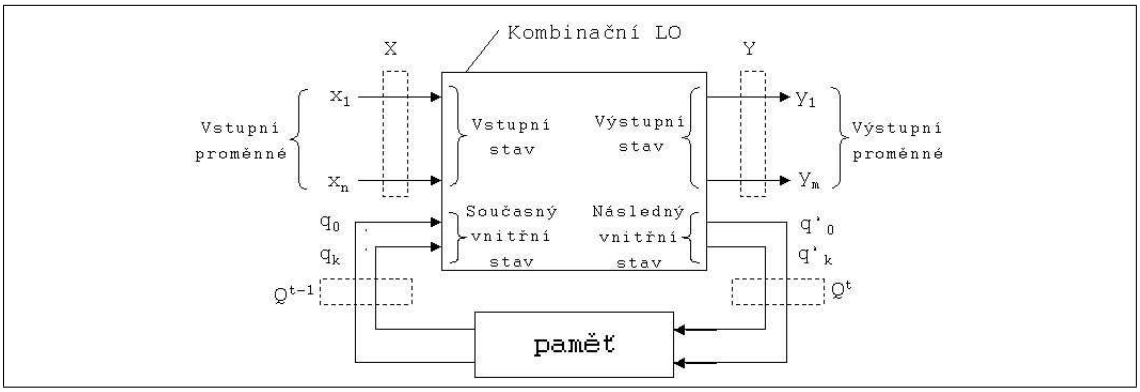
K dynamickým hazardům dochází tehdy, jestliže při změně vstupní proměnné a současně stabilním stavu ostatních vstupních proměnných dojde vlivem fyzikálních vlastností logických členů k několikanásobné změně výstupní proměnné ačkoliv podle zákonů Booleovy algebry mělo dojít ke změně pouze jednorázové.

Za dynamický hazard je vždy zodpovědný statický hazard, který se nachází v zapojení v některém z nižších bloků ⇒ pro vyvarování se dynamických hazardů, musíme v první řadě odstranit hazardy statické.

Dynamické hazardy se nedají zjistit z map, ale pouze z časového diagramu. Dynamický hazard nevzniká v obvodech sestavených dle disjunktivní či konjunktivní formy, ale v obvodech realizovaných dle smíšených forem výrazu.

2.8 Obecný model sekvenčního logického systému a formalizace popisu jako konečného automatu

Sekvenční obvod odvozuje výstupní písmena v závislosti na současných i předchozích vstupních písmenech. Obsahuje tedy paměťové členy. V paměťových členech je uložen předchozí stav. Výstupy paměťových členů se nazývají vnitřní proměnné. Kombinace vnitřních proměnných je vnitřním stavem. Vnitřní stav se vstupním písmenem určuje výstupní písmeno.



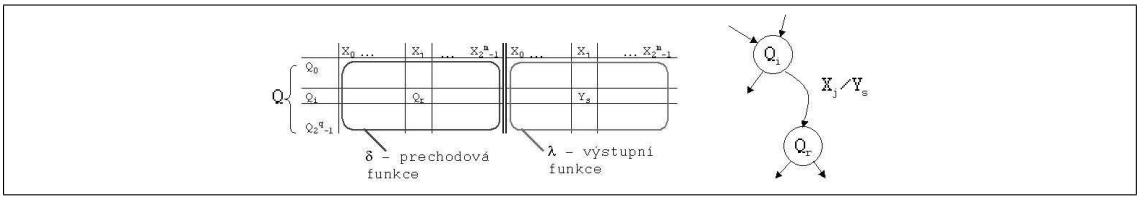
Obrázek 2.8: sekvenční systém

Na konečný automat můžeme pohlížet jako na model chování sekvenčních systémů. Jedná se o uspořádanou šestici $A = (X, Y, Q, \delta, \lambda, Q_0)$.
X - vstupní abeceda
Y - výstupní abeceda
Q - množina vnitřních stavů
δ - stavově přechodová funkce - ke každé dvojici $X_{i,t}, Q_{i,t-1}$ přiřazuje příští vnitřní stav $Q_{i,t}$
λ - výstupní funkce - definuje výstupní písmeno
Q_0 - počáteční stav

2.9 Automaty typu Moore a Mealy

2.9.1 Mealyho automat

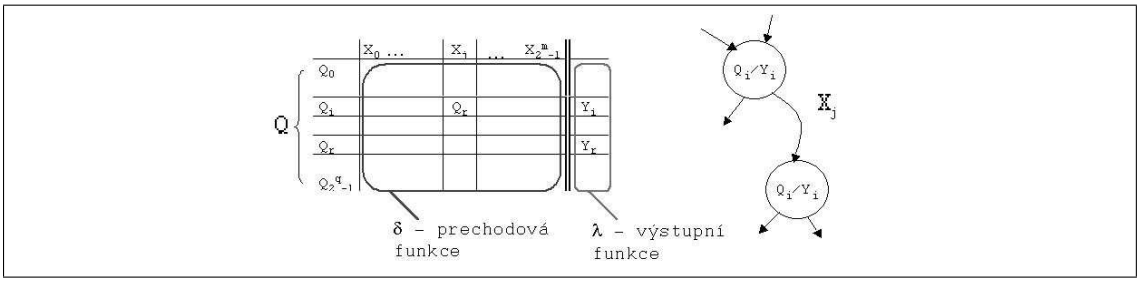
Pokud je automat ve stavu Q_i a na jeho vstup se přivede vstupní písmeno X_j, pak automat vysílá výstupní písmeno Y_s a přechází do stavu Q_r. Automat má m vstupních proměnných, na kterých může existovat 2^m různých vstupních písmen. Množina vnitřních stavů obsahuje 2^q vnitřních stavů Q_0 až Q_{2^q-1}.



Obrázek 2.9: Mealyho automat

2.9.2 Moorův automat

Výstupní písmeno u Moorova automatu je nezávislé na právě existujícím vstupním písmenu.



Obrázek 2.10: Mealyho automat

Počáteční stav u obou tabulek automatů musíme nějak explicitně vyjádřit. Mealyho a Moorův aoutomat se mohou vzájemně přeměňovat.

2.10 Ekvivalence vnitřních stavů, třídy ekvivalence

Ekvivalence stavů - pro daný vstupní symbol a několik současných vnitřních stavů, přechody v tabulce přechodů končí ve stejném stavu nebo ve stavech o nichž již víme, že jsou vzájemně ekvivalentní.

Aby dva stavy nebyly vzájemně ekvivalentní, stačí vzájemná rozdílnost výstupních symbolů.

Důsledkem ekvivalence je snížení počtu vnitřních stavů. Důsledkem této redukce může být i redukce počtu vnitřních proměnných, tedy i klopných obvodů.

Prostá ekvivalence - pro dva vnitřní stavy platí, že mají shodné všechny následné vnitřní stavy i výstupní symbol, tj. jejich řádky v přechodové tabulce jsou identické.

Pseudoekvivalence - týká se neúplně určených automatů

K-ekvivalence - značí stavy, u kterých zjistíme ekvivalenci tehdy, jestliže provedeme k přechodů v tabulce přechodů. Dva stavy jsou *k-ekvivalentní* tehdy, jestliže pro daný vstupní symbol se z obou

stavů po k přechodech dostaneme do stejného stavu nebo do dvou různých stavů, o kterých víme, že jsou ekvivalentní. Musí dál platit, že výstupní symboly jsou pro oba výchozí stavy stejné.

2.11 Kódování vnitřních stavů v synchronních i asynchronních systémech

2.11.1 Synchronní režim

V synchronním režimu je přechod mezi jednotlivými vnitřními stavy uskutečňován synchronizačními impulsy. Šířka impulsů musí být kratší než nejkratší přechodový děj v kombinační i paměťové části. Výhodou synchronizace je to, že není třeba dodržovat podmínku změny pouze jedné proměnné, neboť všechny paměťové členy jsou řízeny ve stejný časový okamžik krátkým synchronizačním signálem.

Všechny přechody mezi vnitřními stavy jsou vázány na podmínku. Podmínkou přechodu může být jedničková úroveň hodinového signálu nebo náběžná/závěrná hrana hodinového impulsu.

Kódování

Pod kódováním se rozumí přiřazení q-tic nul a jedniček identifikátorům označující jednotlivé vnitřní stavy $(Q_1, Q_2, \ldots)$. Vnitřní kód můžeme tedy chápat jako izomorfní zobrazení $\varphi : Q \to \{0,1\}^q$.

V synchronním provedení můžeme realizovat sekvenční obvod pro jakýkoliv vnitřní kód. Volba vnitřního kódu ovlivňuje složitost příslušné kombinační části sekvenčního obvodu. V praxi se v synchronním režimu při volbě kódu vychází z celkové koncepce sekvenčního obvodu (binární kód, kód 1 z N apod.). To má za následek zjednodušení kombinační části v rámci dané koncepce.

2.11.2 Asynchronní režim

V asynchronním režimu vystupuje diskrétní čas jako posloupnost bodů na časové ose, v nichž se mění kterákoliv vstupní, výstupní nebo vnitřní proměnná. Časové okamžiky se nazývají takty. Asynchronní režim pracuje v *základním režimu*, jestliže je zaručeno, že nová změna vstupních proměnných nastává až po ustálení stavů po předchozí změně vstupních proměnných.

Asynchronní = nesoučasný. Dva nezávislé děje, které probíhají bez vzájemné vazby (žádný z nich nečeká na druhý).

Kódování

Na výběru kódu je v asynchronním režimu silně závislá jeho realizovatelnost, tj. není možné volit libovolný kód jako je to u synchronního režimu.
Souběh - přechod mezi dvěma různými vnitřními stavy, ve kterém dochází ke změně hodnot u více než jedné vnitřní proměnné. Pokud pořadí těchto změn může ovlivnit provedení požadovaného přechodu, pak se jedná o *kritický souběh*, jinak je to *nekritický souběh*.
Požadavek pro správné kódování:

> nechť $Q_i, Q_j \in Q$ jsou dva vnitřní stavy automatu. Nechť $X_i \in X$ je nějaké vstupní písmeno, pro něž platí, že $\delta(Q_i, X_i) = Q_j$ nebo $\delta(Q_j, X_i) = Q_i$, tj. existují přechody ze stavu Q_i do Q_j nebo naopak.
>
> správné kódování je takové kódování, které stavům Q_i a Q_j přiřadí sousední k-tice nul a jedniček.

Pro volbu vnitřního kódu s touto vlastností se používá pomůcka ve formě neorientovaného grafu, ve kterém uzly reprezentují opět vnitřní stavy automatu a neorientované hrany reprezentují existenci přechodu mezi stavy, které jsou těmito hranami propojeny. Do grafu se nemusí znázorňovat ty hrany, které reprezentují *nekritické souběhy*.

2.12 Hazardy v asynchronních systémech

Pozn. Začal bych od hazardů v kombinačních systémech - viz. 2.7 strana 21.
Struktura asynchronního obvodu vykazuje jistou odolnost vůči statickým hazardům. Klopným obvodům nevadí takový přechodný vstup, který vyžaduje právě funkci *paměť*.

Podstatné hazardy jsou pouze v asynchronních obvodech a způsobují chyby trvalého charakteru. Důvodem jejich existence je shoda dvou okolností - časový posun reakcí jednotlivých větví kombinační části na nové vstupní písmenu X_j a jistá choulostivost stavově přechodové funkce δ na tento posun, tj. již se prosadil nový stav Q_j, ale ještě je někde v kombinačním obvodu díky velkému časovému zpoždění staré vstupní písmeno X_i. Automat tedy může přejít do dalšího stavu, do kterého vůbec neměl přejít, ale měl zůstat ve stavu Q_i.
Iniciátor - vstupní proměnná, jejíž hodnotou se lišila vstupní písmena X_i a X_j.
Rezultant - vnitřní proměnná, jejíž změna nastala v důsledku změny iniciátoru.
Odstranění podstatného hazardu:

- zařazení zpožďovacího členu do všech vnitřních proměnných, které při některém přechodu vystupují v úloze *rezultanta*.
- zařazení filtračního členu do všech budících funkcí, které při některém přechodu vystupují v úloze chybových proměnných.

2.13 Vlastnosti paměťových členů - klopných obvodů

Klopné obvody dělíme na:

- **Obvody synchronní**
  - obvody synchronizované úrovní logického
  - obvody synchronizované hranou signálu
- **Obvody asynchronní** - na vstupní písmeno reaguje klopný obvod ihned

Klopné obvody můžeme dále dělit podle konstrukce na:

- **Statické** - jsou tvořeny hradly a při zastavení synchronizačních pulsů uchovávají svůj stav
- **Dynamické** - paměťový prvek je tvořen kapacitami. Při zastavení synchronizačních pulsů po čase ztrácejí svůj vnitřní stav

KO dále dělíme podle funkce na:

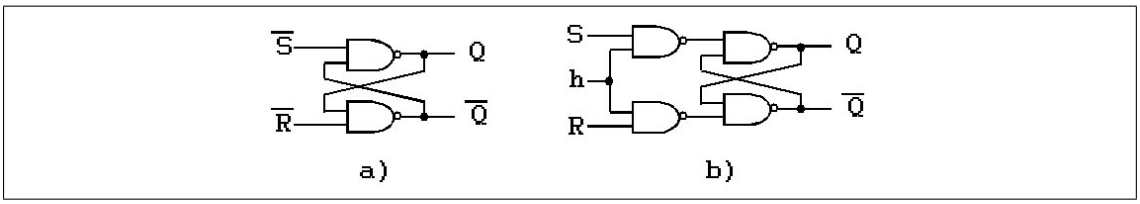
- **RS** - má dva vstupy SET a RESET a disponuje funkcemi *pamatuj*, *zapiš 1* a *zapiš 0*
- **JK** - má dva vstupy a disponuje funkcemi *pamatuj*, *zapiš 1*, *zapiš 0* a *překlop*
- **D** - má jeden vstup a disponuje funkcemi *zapiš 1* a *zapiš 0*
- **T** - má jeden vstup a disponuje funkcemi *pamatuj* a *překlop*

Podle počtu stabilních stavů se KO dělí na:

- **astabilní** - bez vnějšího buzení periodicky střídají na výstupu 0 nebo 1
- **monostabilní** - mají pouze jeden stabilní stav. Mohou být uvedeny do nestabilního stavu, ale po nějakém čase se vrací zpět do stabilního stavu
- **bistabilní** - mají dva stabilní stavy. Jsou schopny plnit funkci paměti

Tabulka 2.1: možné funkce RS klopného obvodu

S	R	Q_{n+1}	*funkce*
0	0	Q_n	„P“
0	1	0	„0“
1	0	1	„1“
1	1	–	„—“



Obrázek 2.11: RS klopný obvod

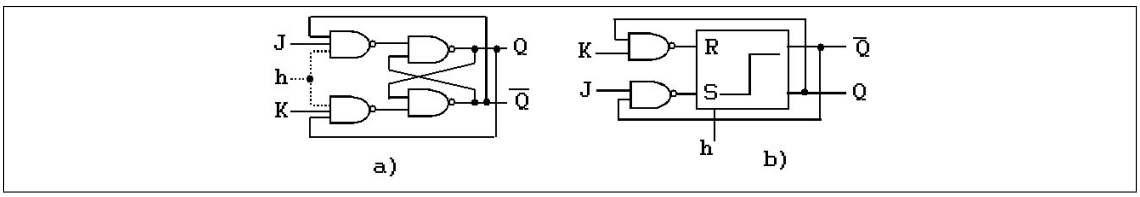
RS klopný obvod

RS klopný obvod může být asynchronní (obrázek 2.11 a) nebo synchronní (obrázek 2.11 b). V synchronním provedení plní obvod funkce uvedené v tabulce 2.1 při signálu $h = 1$.

Klopné obvody v provedení *master-slave* - jsou to dva klopné obvody zapojené v sérii a inverzně řízené. Druhý z obvodů otrocky sleduje výstup prvního. Smyslem celého zapojení je získat hranový klopný obvod, který na svém vstupu (tj. otroku) klopí se závěrnou hranou hodinového signálu.

JK klopný obvod

JK klopný obvod je nejobecnější - disponuje všemi funkcemi. Obrázek 2.12 a) znázorňuje asynchronní obvod. Pokud přidáme vstup pro hodinový signál (na obrázku 2.12 a) tečkovaně), dostaneme synchronní KO.



Obrázek 2.12: JK klopný obvod

Tabulka 2.2: možné funkce JK klopného obvodu

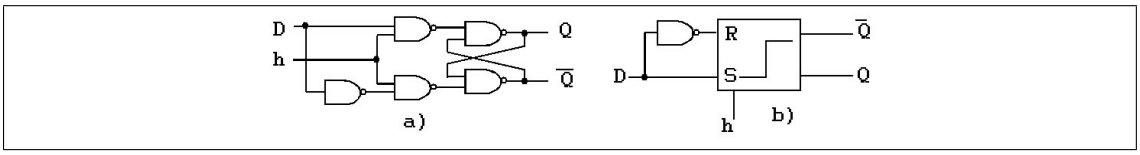
J	K	Q_{n+1}	*funkce*
0	0	Q_n	„P“
0	1	0	„0“
1	0	1	„1“
1	1	$\overline{Q_n}$	„K“

Tabulka 2.3: možné funkce D klopného obvodu

D	Q_{n+1}	*funkce*
0	0	„0“
1	1	„1“

D klopný obvod

Realizace tohoto obvodu má smysl pouze v synchronním provedení. Má pouze jeden vstup (kromě hodin). Výstup obvodu kopíruje vstup za přítomnosti hodinového signálu $h = 1$. Funkce tohoto obvodu jsou v tabulce 2.3



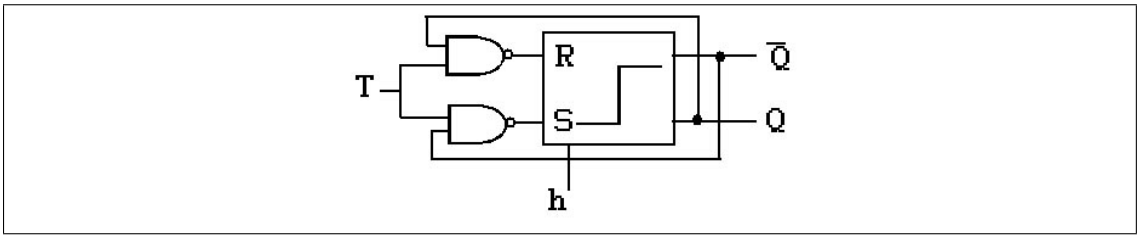
Obrázek 2.13: D klopný obvod

T klopný obvod

Tento KO má pouze jeden hlavní vstup. Synchronní hladinové provedení vznikne použitím JK obvodu s tím, že vstupy J a K sloučíme do vstupu T (obrázek 2.14), tj. J=K=T. Z původních čtyř funkcí JK KO se využijí pouze dvě.

Tabulka 2.4: možné funkce T klopného obvodu

T	Q_{n+1}	*funkce*
0	Q_n	„P“
1	$\overline{Q_n}$	„K“



Obrázek 2.14: T klopný obvod

2.14 Návrh synchronních sekvenčních systémů

1. sestavení minimální formy automatu

2. kódování vnitřních stavů automatu
3. volba typu klopných obvodů
4. sestrojení přechodové tabulky δ a výstupní tabulky λ pro zakódované vnitřní stavy, vstupní a výstupní písmena
5. odvození map pro jednotlivé budící proměnné a pro jednotlivé vstupní proměnné
6. sestrojení minimalizačních smyček v mapách budících a výstupních proměnných a analytické vyjádření budících a výstupních proměnných
7. konstrukce sekvenčního obvodu
8. návrh parametrů hodinových impulsů
9. ověření návrhu

2.15 Programovatelné logické struktury (PLA, PGA, ROM)

Při návrhu nějakého logického systému nemusíme vždy vytvářet speciální struktury realizující požadovanou funkci, ale můžeme využít relativně univerzální strukturu postavenou na bázi nějakého mikroprocesoru nebo mikroprogramovatelného obvodu. Požadované funkce dosáhneme vhodným obsahem paměti. Tímto máme zaručenu výhodu, co se týče rychlosti návrhu a také snadné pozdější modifikace požadovaného chování.
Návrhy integrovaných obvodů:

- **zákaznický návrh** - při výrobě se vytvářejí speciální masky
- **polozákaznický návrh** - návrhář používá předem připravené a na čipu již rozmístěné funkční celky. Návrhář ovlivňuje pouze některé konečné fáze výroby ⇒ personifikace obvodu
- **oddělení personifikace obvodů od jejich realizace** - čipy typu FPLA, PLD, PAL. U těchto obvodů může být jejich vnitřní konfigurace definována mimo výrobní závod → odpadá spolupráce návrhář-výrobce

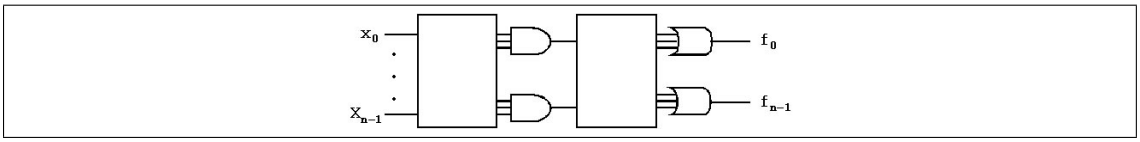
Spojovací body, které požadovaným způsobem konfigurují připravené dílčí buňky jsou realizovány jako:

- **tavné bipolární pojistky** - hlavní nevýhodou je pouze jednorázové programování, tj. není možná pozdější modifikace
- **tranzistory CMOS** - slouží jako řízení spojovacích spínačů. Po vymazání ultrafialovým světlem a novým naprogramováním lze měnit konfiguraci
- **bistabilní klopné obvody** - slouží jako řízení spojovacích bodů. Po vypnutí se však jejich obsah ztrácí ⇒ po zapnutí se musí program nahrát z trvale připojené paměti ROM. Případné změny lze provést výměnou nebo přeprogramováním připojené paměti

PLA - Programmable Logic Array

PLA patří do skupiny specificky orientovaná struktura, tj. možnosti v propojení vnitřní stuktury čipu jsou již při výrobě určeny do té míry, že uživatel pouze ruší nebo ponechává předem dané spoje.

Základem konfigurovatelné kombinační části jsou programovatelné matice spojů. Modelem obvodu musí být logický výraz v disjukntní formě. První matice odpovídá součinovým termům a druhá vytváří výsledné součty těchto termů - viz. obrázek 2.15.



Obrázek 2.15: PLA

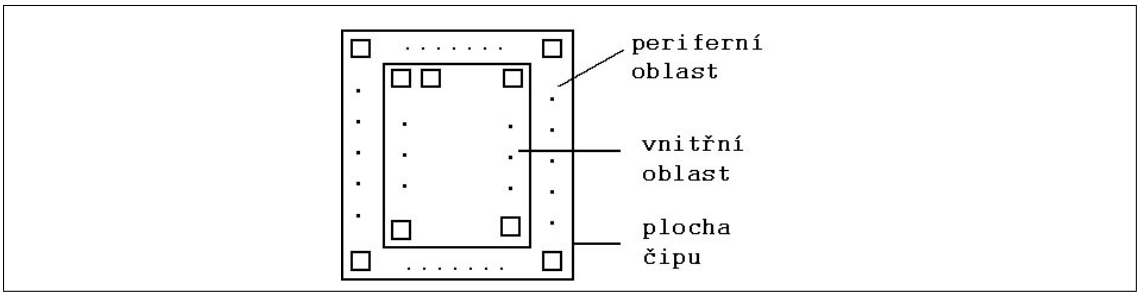
ROM - permanentní paměť

Pomocí dvou matic jako ve struktuře *PLA* můžeme vytvořit permanentní paměť *ROM* tím, že v první matici naprogramujeme dekodér z binárního kódu do kódu *1 z N*.

PGA - Programmable Gate Array

PGA patří do obvodů s obecně orientovanou strukturou, tj. čip se skládá z určitého počtu stejných vnitřních konfigurovatelných buněk. Pomocí vstupů a výstupů těchto buněk a pomocí předem připravených propojovacích kanálů můžeme vytvářet výslednou obecnou strukturu. Tyto mezibuňkové kanály nejsou předem připojeny k žádným buňkám, ale jsou volně dostupné - viz. obrázek 2.16.

Periferní oblast je obsazena vstupními a výstupními buňkami. Vnitřní oblast obsahuje matici shodných vnitřních konfigurovatelných buněk. Mezi konfigurovatelnými buňkami jsou propojovací kanály.



Obrázek 2.16: PGA