6.4 SYNTÉZA LOGICKÝCH OBVODOV Z MODULOV

Univerzálnou funkciou n premenných $x_1, x_2, ..., x_n$ sa nazýva Boolova funkcia $U(u_1, u_2, ..., u_v)$, pre ktorú platí

$$U(u_1, u_2, ..., u_v) = f_j(x_1, x_2, ..., x_n)\ \forall j \in <1, 2^{2^n}> \qquad (6.28)$$

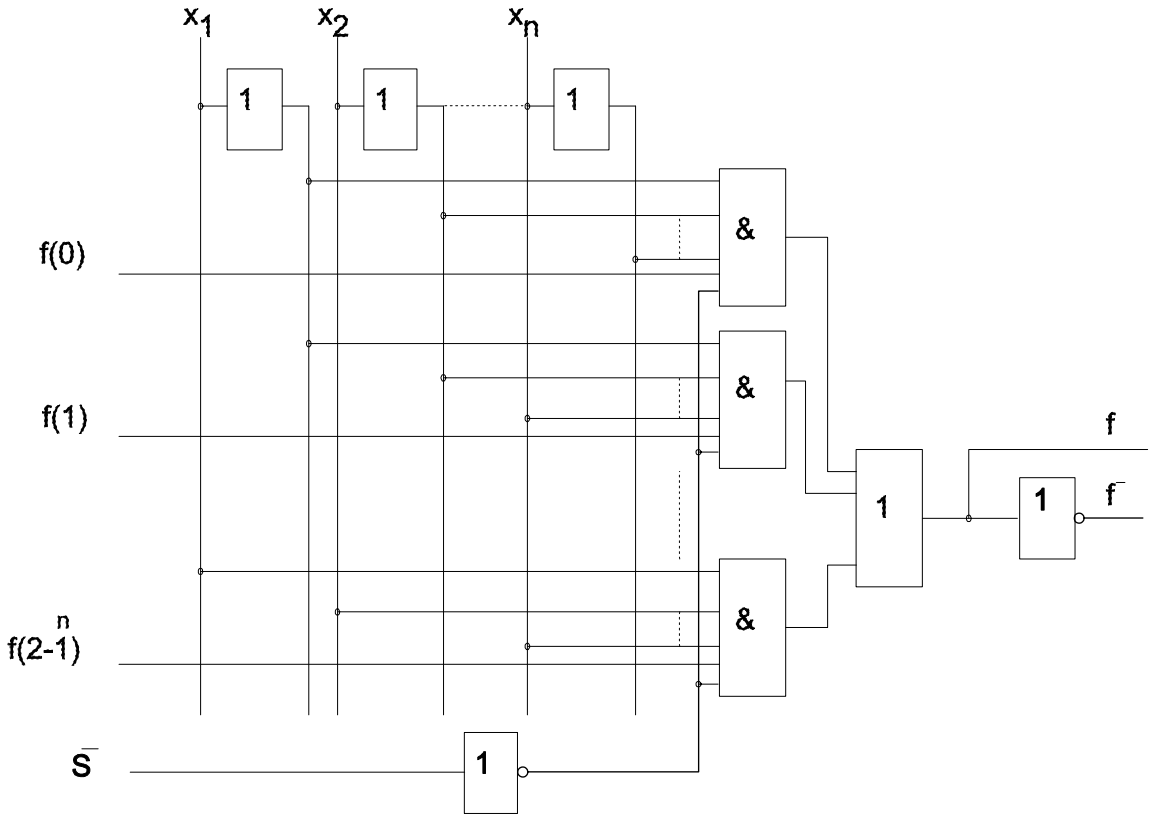
kde $u_i \in \{0, 1, x_1, x_2, ..., x_n, g_1, g_2, ..., g_s\}$, pričom
g_k je funkciou premenných $x_1, x_2, ..., x_n\ \forall\ k \in \{1,2,...,s\}$

Univerzálna funkcia v kanonickom tvare

$$f(x_1, x_2, \dots x_n) = \bigvee_{i=0}^{2^n-1} a_i\ x_1^{i_1}\ x_2^{i_2} \dots\ x_n^{i_n} \qquad (6.29)$$

kde $a_i \in \{0, 1\}$, $i_1\ i_2\ ...\ i_n$ tvoria binárne vyjadrenie čísla **i** a $x^0_j = \overline{x}_j$ **a** $x^1_j = x_j$ **pre j = 1, 2, ..., n.**

Multiplexorom sa nazýva kombinačný logický obvod s 2^n **údajovými vstupmi** $d_0, d_1, ..., d^{2n-1}$, **n riadiacimi vstupmi** $c_1, c_2, ..., c_n$ **a jedným výstupom f**, pre ktorý platí, že $f = d_i$, $0 \le i \le 2^{n-1}$, práve vtedy, ak číslo $c_1, c_2, ..., c_n$ je binárnym vyjadrením čísla **i** ($c_j \in \{0, 1\}$ pre $1 \le j \le n$).



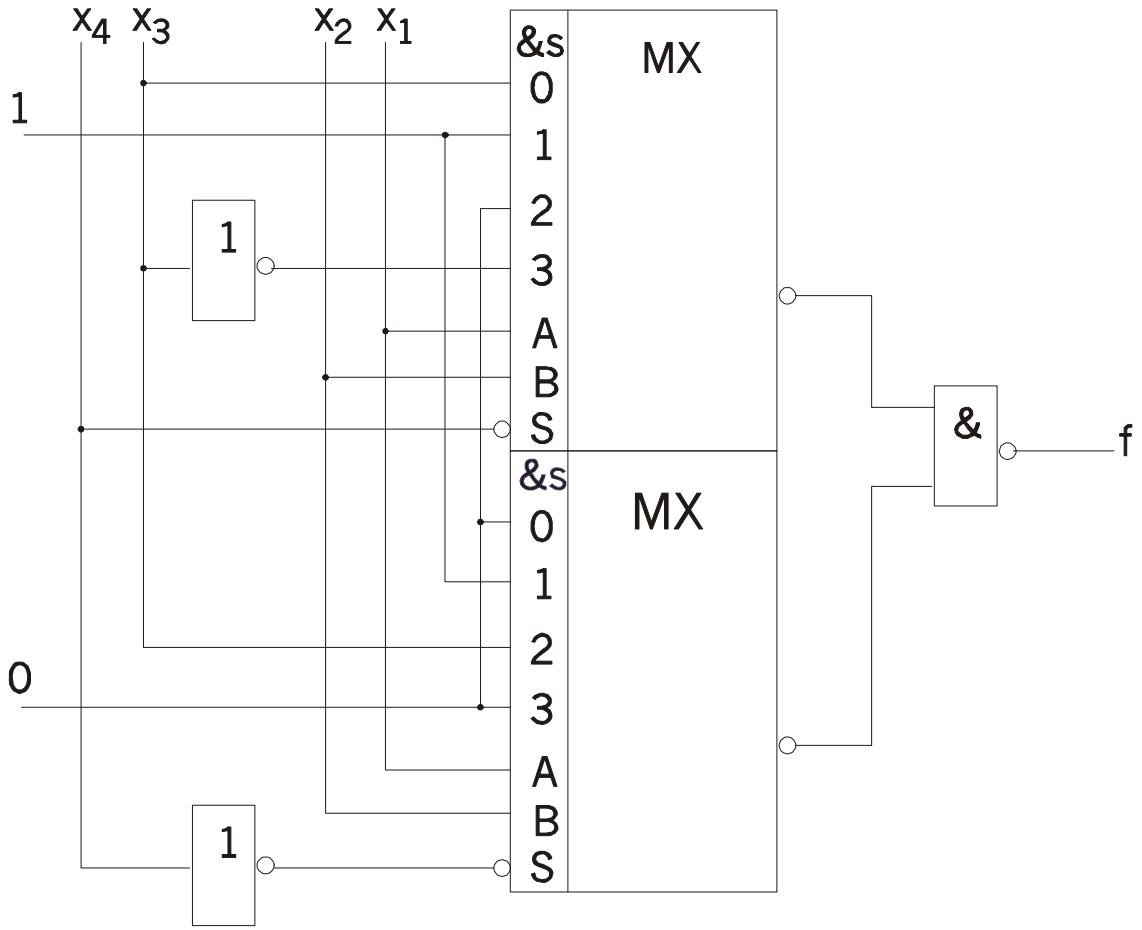
Obr. 6.24 Štruktúrna schéma multiplexora

Nevýhoda - veľký počet vstupov obvodu - **$(n + 2^n)$**

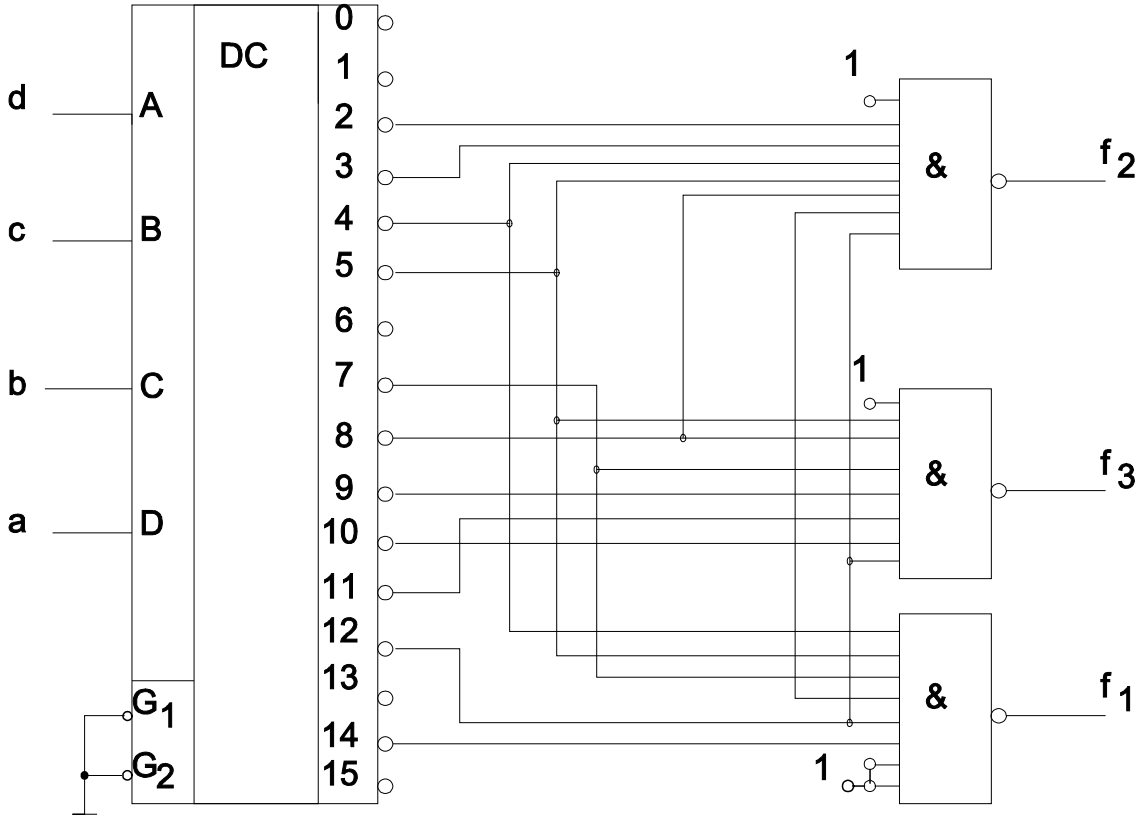
Celkový počet vstupov sa môže znížiť takmer o polovicu, ak sa realizovaná B-funkcia **$f(x_1, x_2, ..., x_n)$** na základe Shannonovho pravidla postupne rozloží podľa **n - 1** premenných, čím vznikne vyjadrenie v tvare

$$f(x_1, x_2, \dots, x_n) = \bigvee_{i=0}^{2^{n-1}-1} x_1^{i1} x_2^{i2} \dots x_{n-1}^{in} . f(i_1, i_2, \dots, i_{n-1}, x_n) \qquad (6.14)$$

kde $f(i_1, i_2, \dots, i_{n-1}, x_n)$ sú reziduálne funkcie, ktoré nadobúdajú hodnoty z množiny **{0, 1, $\overline{x_i}$, $\overline{x_i}$}.**

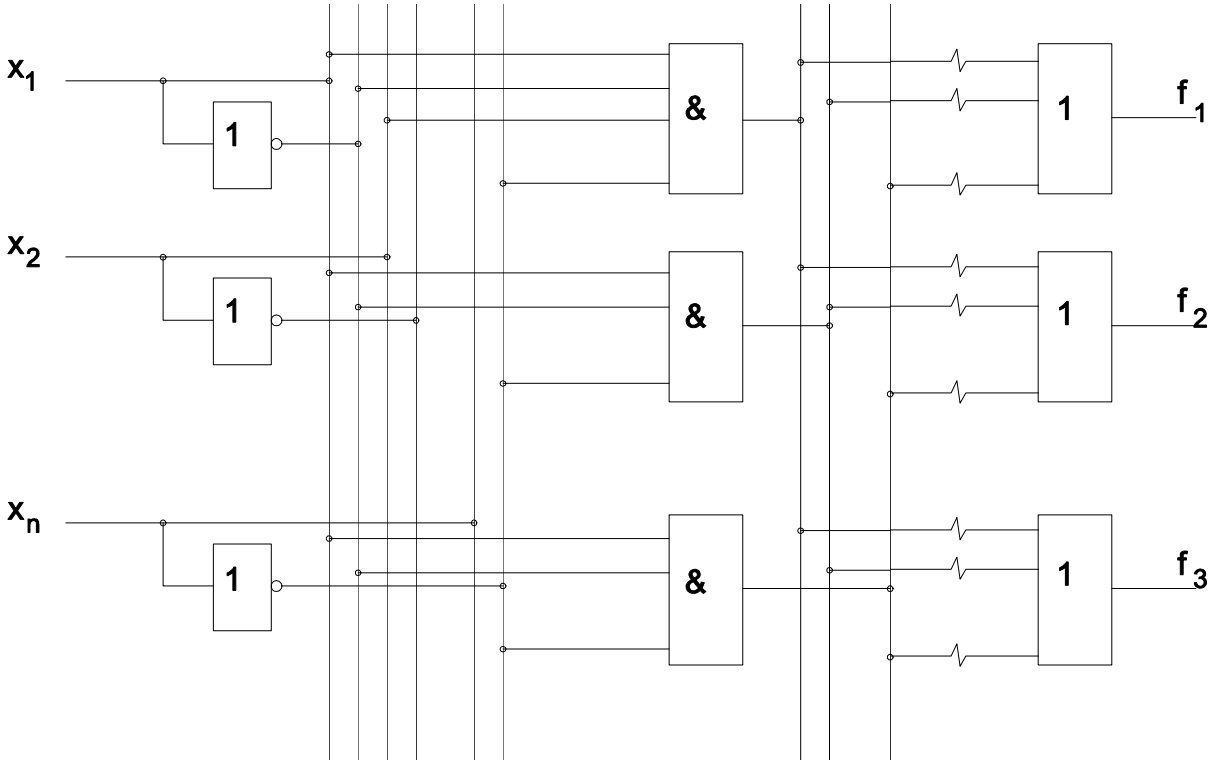


Obr.6.1 Realizácia B-funkcie štyroch premenných dvoma multiplexormi s dvoma riadiacimi vstupmi



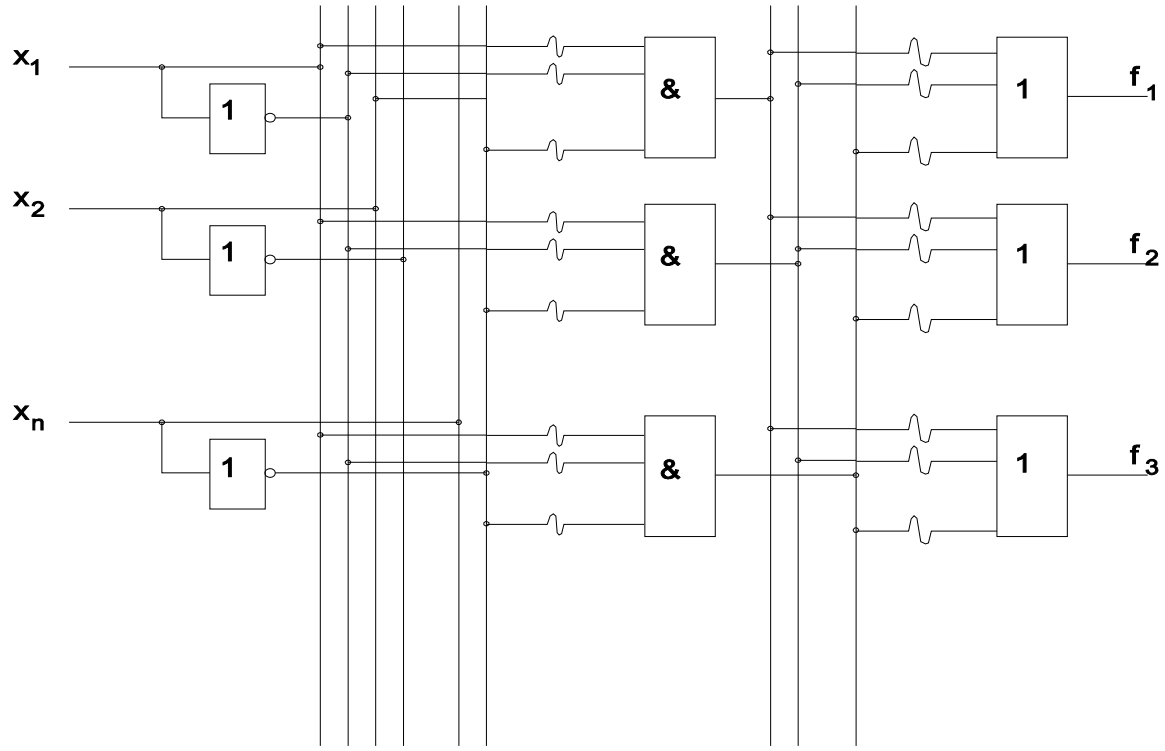
Obr. 6.26 Realizácia systému B-funkcií prostredníctvom demultiplexorov

Použitie pamäti typu ROM pre realizáciu ľubovoľnej B-funkcie

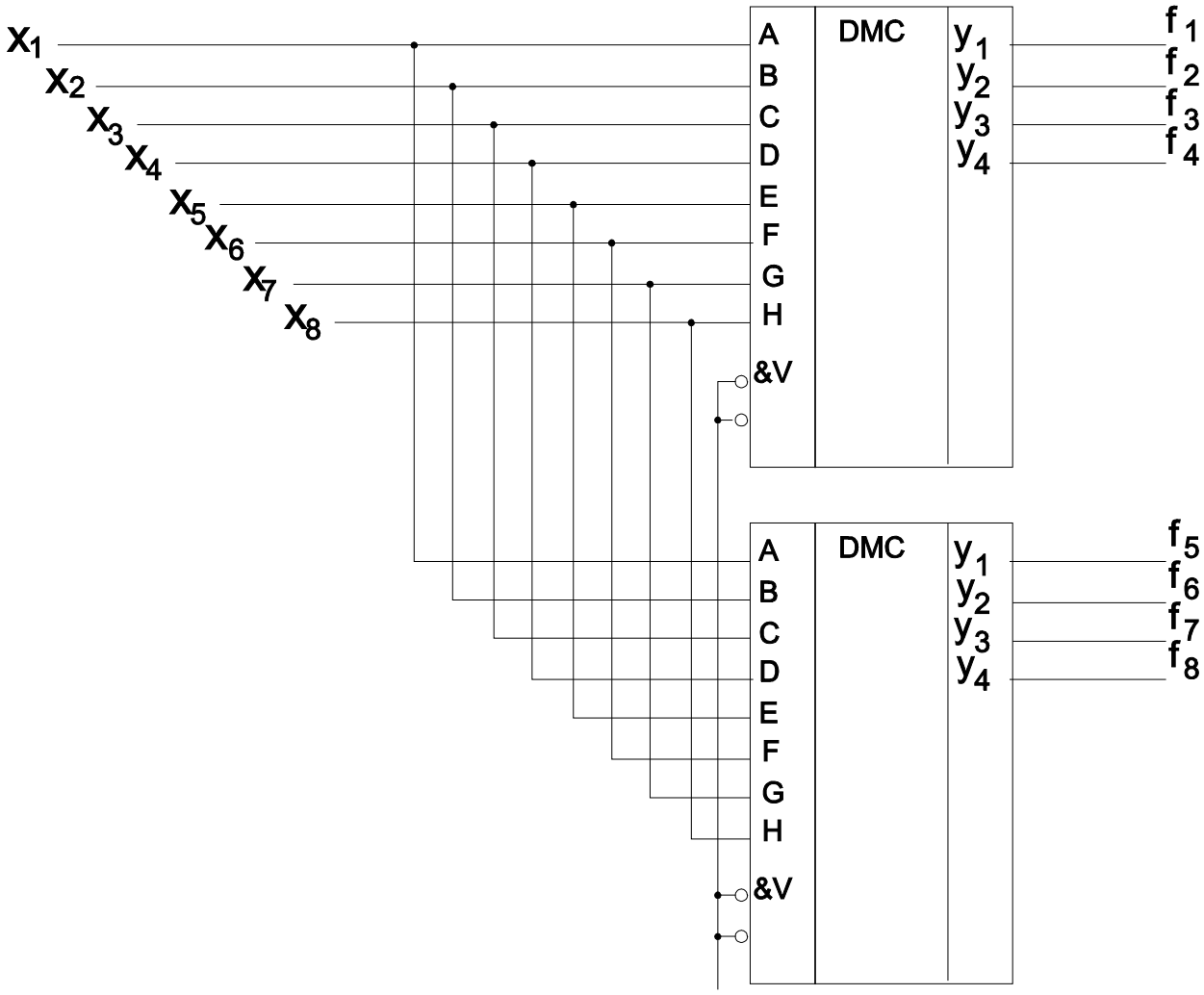


Obr. 6.27 Štruktúra pamäti PROM

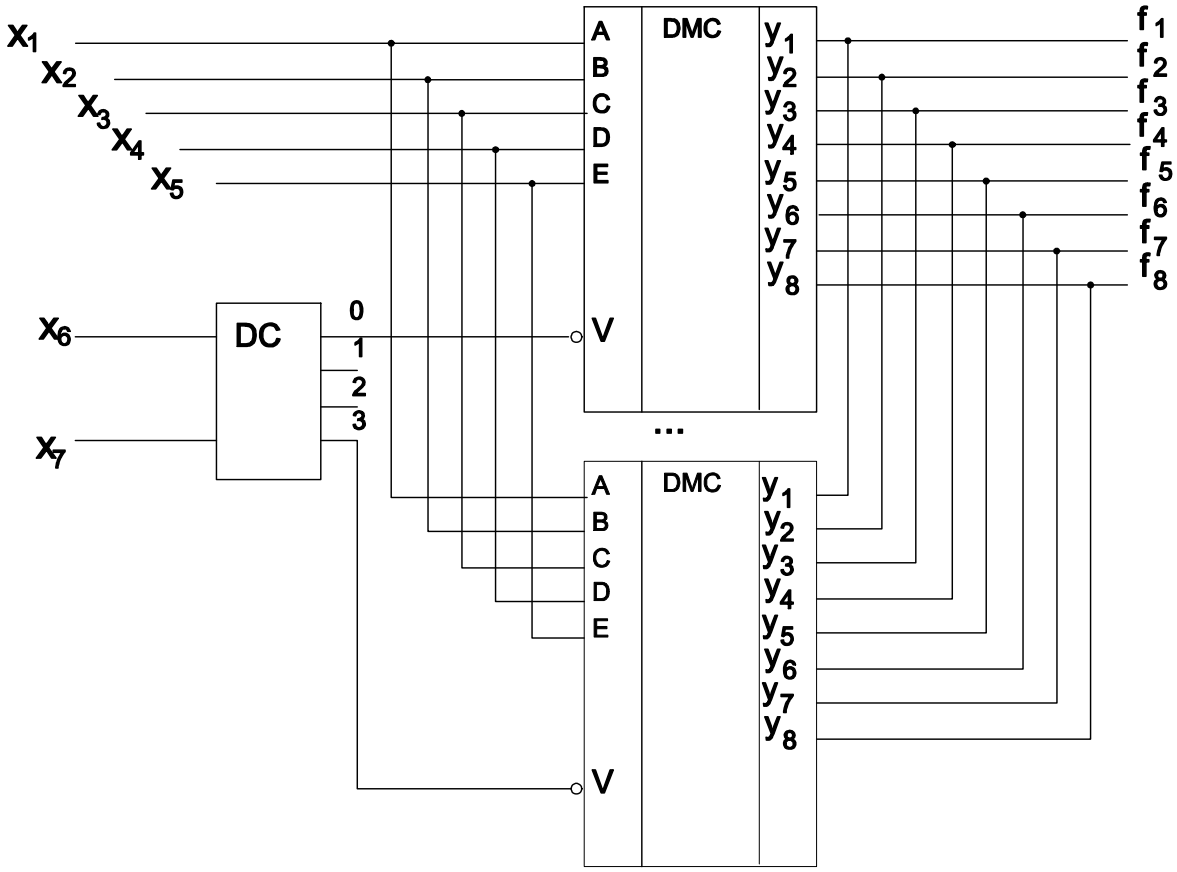
Realizácia B-funkcií prostredníctvom programovateľných logických polí (**PLA - Programmable Logic Array**),)



Obr. 6.28 Štruktúra programovateľného logického poľa

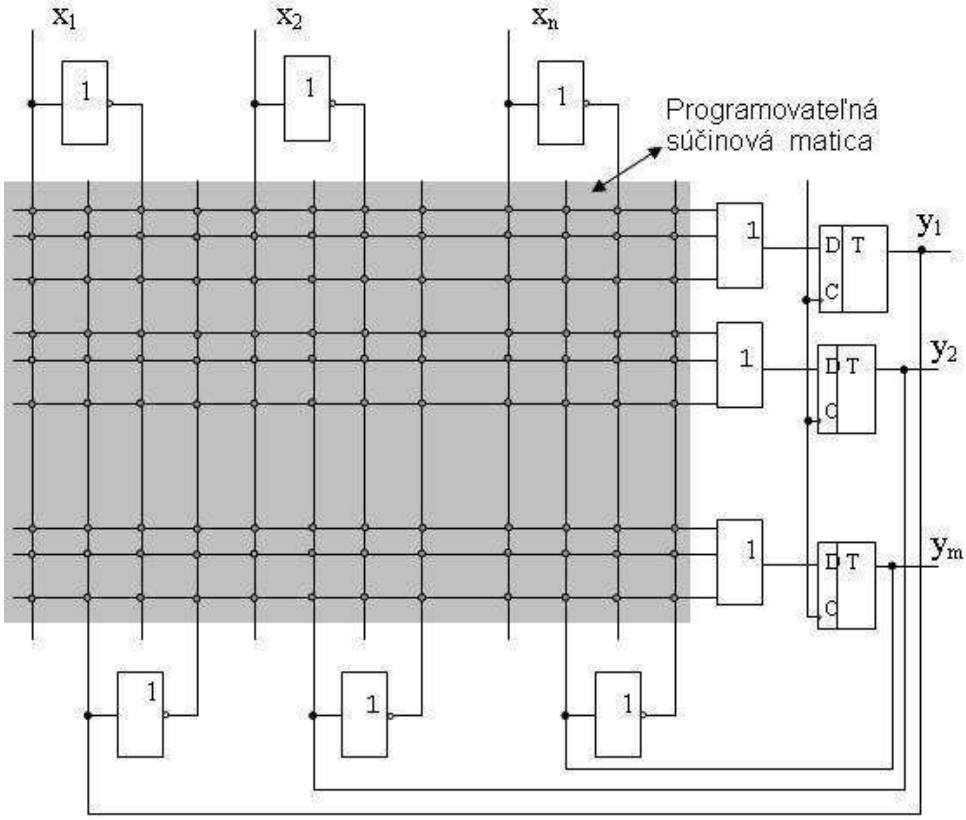


Obr. 6.29 Zapojenie PROM pre zväčšenie počtu výstupných funkcií

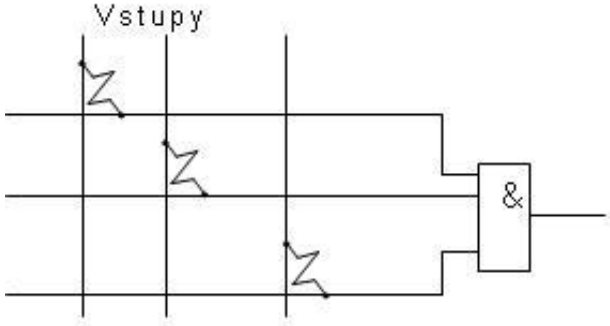


Obr. 6.30 Zapojenie PROM pre zväčšenie počtu adresných vstupov

Vnútorná štruktúra jednoduchých programovateľných logických obvodov - Simple Programmable Logic Device (SPLD).



Obr.6.2 Vnútorná štruktúra SPLD typu PAL



Obr.6.3 Súčinový LČ

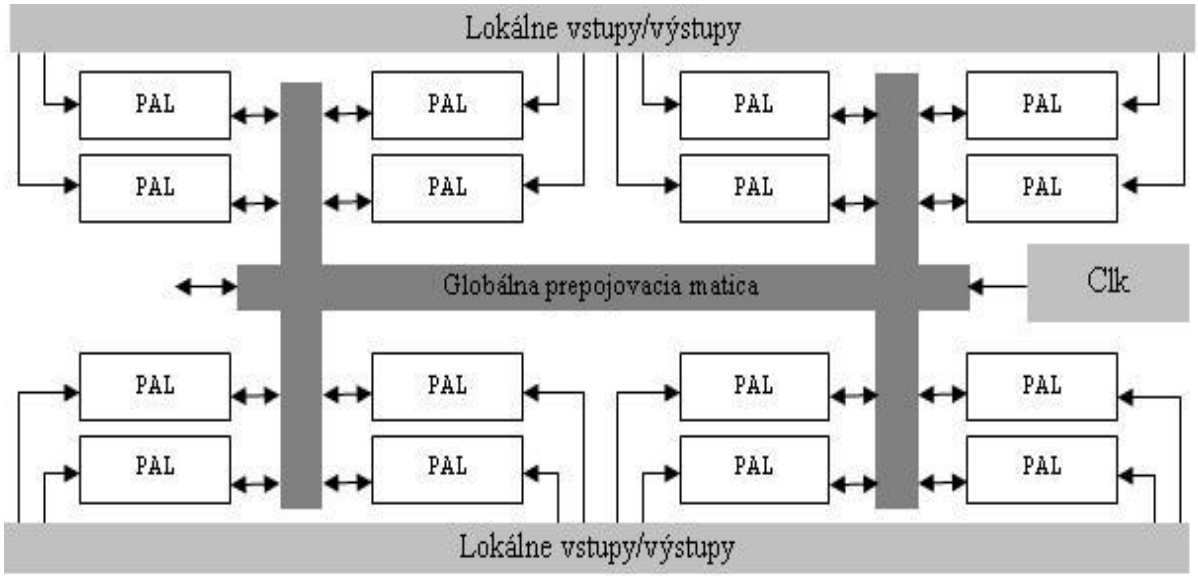
V technológii CMOS sú prepínače využívané podobne ako u pamätí PROM, EPROM alebo EEPROM. Zložitejšie obvody kategórie FPGA mávajú často spínače riadené statickou pamäťou RAM.

Do kategórie SPLD je možné zaradiť obvody nasledujúcich typov :

PAL - obvody typu PAL (Programmable Array Logic) majú štruktúru podľa Obr.6.2 a Obr.6.3. Niektoré staršie typy nemali napríklad výstupné registre, takže boli vhodné skôr pre kombinačnú logiku.

PLA - obvody typu PLA (Programmable Logic Array) majú všeobecnejšiu štruktúru než PAL. Majú totiž programovateľnú nielen maticu logických súčinov, ale i nasledujúcu maticu logických súčtov. Obvody SPLD majú veľmi obmedzené prostriedky, takže umožňujú realizovať len jednoduchšie funkcie.

Preto výrobcovia začali združovať viacero takýchto obvodov na jednom čipe spolu s nutnými prostriedkami pre prepojenie. Obvody tejto kategórie nazývame komplexnými programovateľnými logickými obvodmi - Complex Programmable Logic Device (CPLD). Typická štruktúra obvodu CPLD je znázornená na Obr.6.4. CPLD od rôznych výrobcov sa obvykle líši vo vyhotovení blokov vlastnej programovateľnej logiky i keď väčšinou vychádza z klasickej štruktúry PAL.



Obr.6.4 Štruktúra obvodu CPLD

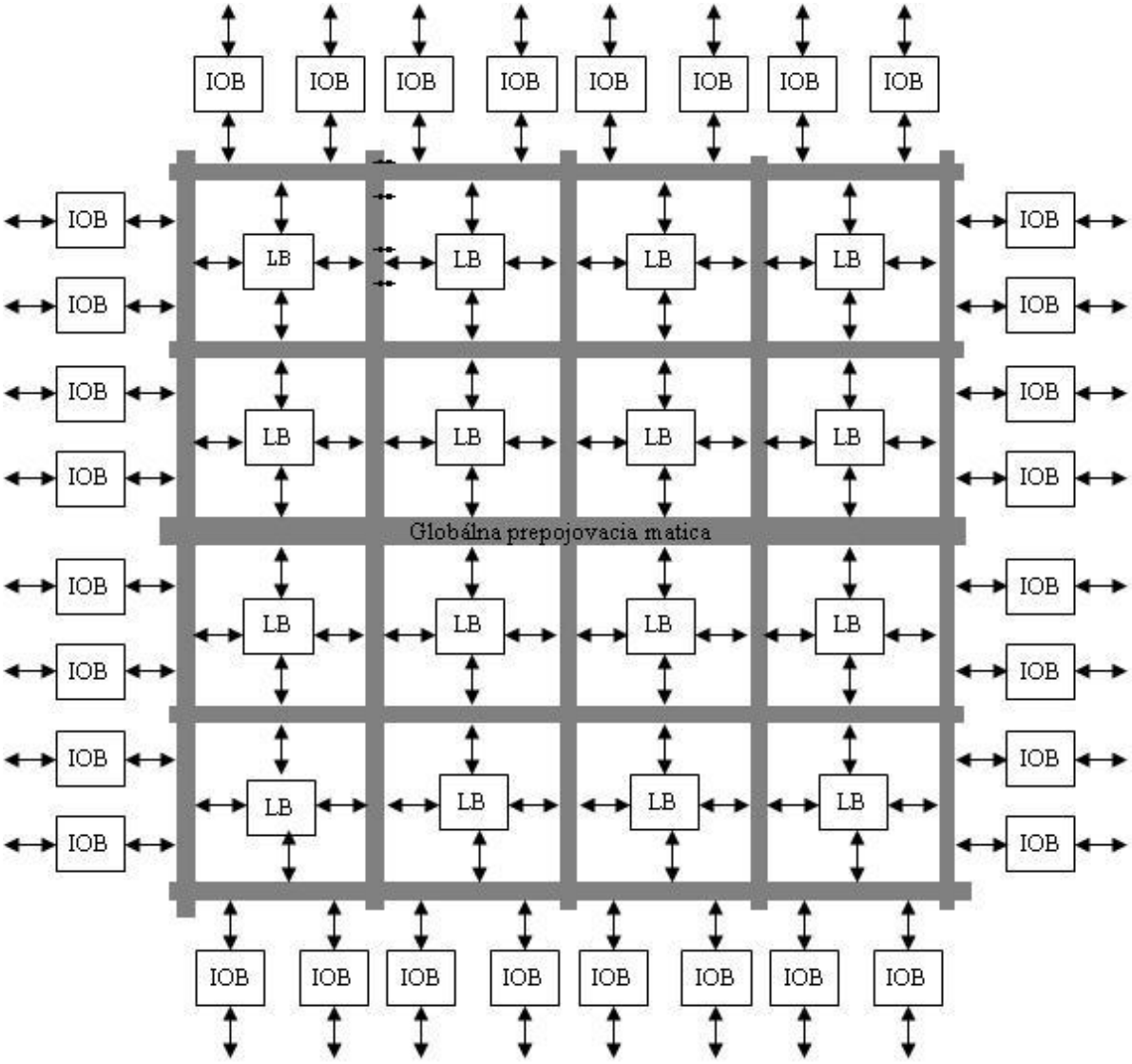
Podstatne vyššiu hustotu integrácie je možné dosiahnuť v **hradlových poliach**, obsahujúcich značný počet systematicky usporiadaných hradiel, prepojenie ktorých sa uskutočňuje na základe požiadaviek používateľa.

Základný kryštál hradlového poľa obsahujúci maticu hradiel, vstupno-výstupných prevodníkov, prípadne ďalšie prvky je vyrábaný až do určitej úrovne hromadne nezávisle od výslednej funkcie obvodu pomocou tzv. štandardných operácií. Pre tento základný kryštál navrhne používateľ prepojenie jednotlivých buniek tak, aby obvod realizoval ním požadované funkcie.

Pre výrobu základného kryštálu sa používajú rozličné technológie, ako napr. **ECL, TTL, STTL, LSTTL, I^2L, NMOS, CMOS**. Každá z použitých technológií je charakterizovaná svojim vlastným súborom predností a nedostatkov. Najpoužívanejšou technológiou sa stáva **CMOS**, ktorá je charakterizovaná malým príkonom, malým oneskorením a vysokým stupňom integrácie. Druhou najpoužívanejšou technológiou je technológia **STTL.**

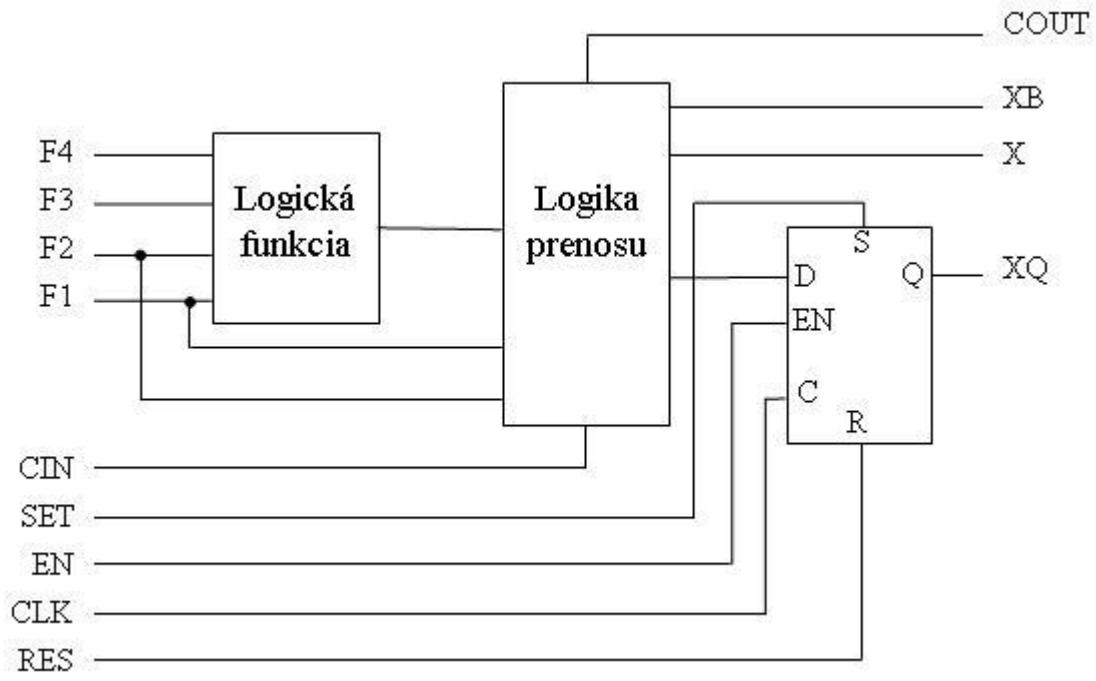
Fyzická realizácia logickej schémy navrhovaného obvodu sa môže uskutočňovať tak, že sa používajú len základné bunky obsahujúce prvky NAND alebo je možné používať štandardné zapojenia často používaných modulov, ako sú rozličné typy preklápacích obvodov a pod., ktoré sú v knižnici modulov.

Najvšeobecnejšiu štruktúru z programovateľných obvodov majú programovateľné hradlové polia (Field Programmable Gate Arrays – FPGA), ktoré obsahujú najviac logiky. Súčasné najväčšie obvody FPGA obsahujú milióny ekvivalentných hradiel (typické dvojvstupové hradlo NAND). Typická štruktúra obvodu FPGA je na Obr.6.5.



Obr.6.5 Štruktúra obvodu FPGA

Bloky označené IOB (Input/Output Block) predstavujú vstupno-výstupné obvody pre každý v-v pin FPGA. Tieto bloky obvykle obsahujú register, budič, multiplexor a ochranné obvody. Bloky LB (Logic Block) predstavujú samotné programovateľné logické bloky. Všetky bloky môžu byť rôzne prepojené globálnou prepojovacou maticou. Najpoužívanejšia štruktúra kofigurovateľného logického bloku je znázornená na Obr.6.6.



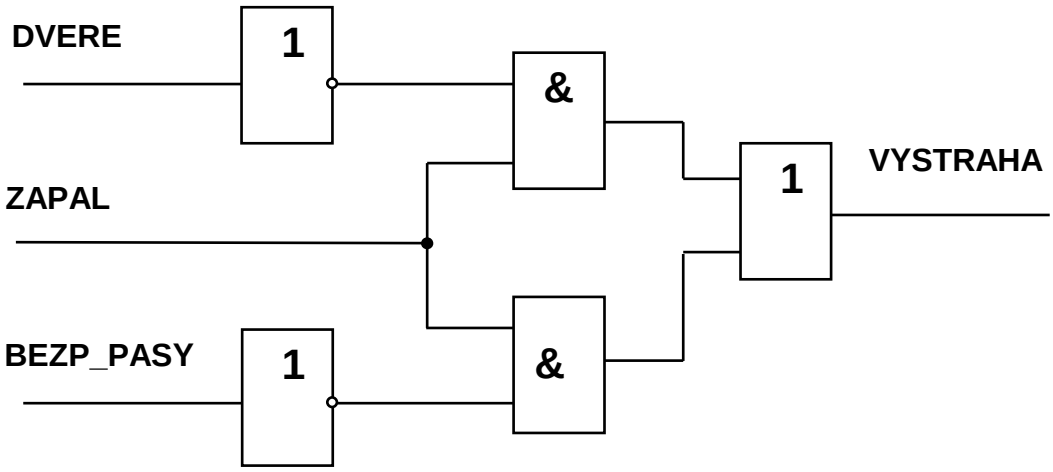
Obr.6.6 Štruktúra logického bloku

FPGA obvykle umožňujú prepojiť niektoré signály logických blokov priamo so susednými bez nutnosti využívať globálnu prepojovaciu maticu. Takéto spoje majú omnoho menšie oneskorenie a umožňujú tak realizovať napríklad rýchle obvody šírenia prenosu, čo je nevyhnutné pre sčítačky alebo násobičky. Okrem blokov znázornených na Obr.6.5 integrujú výrobcovia do FPGA ďalšie prvky. Väčšina moderných FPGA obsahuje obvody pre obnovenie charakteristík hodinového signálu, prípadne pre násobenie alebo delenie jeho frekvencie.

Použitie hradlových polí sa stáva efektívnym, len pri použití prostriedkov automatizovaného návrhu, ktorý značne urýchľuje a zefektívňuje implementáciu hradlového poľa pre realizáciu zadaných funkcií. Proces automatizovaného návrhu pozostáva z dvoch etáp, ktoré na seba tesne nadväzujú. Prvá etapa určuje rozmiestnenie jednotlivých častí logickej schémy na ploche cípu a druhá určuje vzájomné prepojenie jednotlivých prvkov hradlového poľa. Ak sa pre zvolené rozmiestnenie logickej schémy nepodarí nájsť vhodné prepojenie, proces rozmiestnenia sa opakuje, čím vzniká iteračný proces návrhu.

Hlavnou výhodou použitia hradlových polí je skrátenie doby návrhu a realizácie systému, zvýšenie jeho kompaktnosti a spoľahlivosti, zníženie príkonu a pri strednom počte (rádovo 1000 až 100000 kusov) aj zníženie nákladov na realizáciu.

Popis logických systémov prostredníctvom jazyka VHDL



Štruktúrna reprezentácia obvodu pre signalizáciu otvorených dverí a nezapnutých pásov

Chovanie:

VYSTRAHA = ZAPAL_ZAP **and** (DVERE_OTVOR **or** BEZP_PAS_NEZAP)

VHDL popis číslicových systémov na úrovni:

štruktúry: zadávajú sa prvký a vzájomné väzby medzi nimi

chovania: tok **dát – medziregistrové prenosy**

súbežné príkazy, ktoré sa vykonávajú paralelne akonáhle prídu vstupné údaje.

Algoritmus - VHDL dovoľuje súbežné aj sekvenčné príkazy.

Základné komponenty VHDL jazyka

deklarácia entity

telo architektúry

Príklad:

```
    entity  VYSTRAHA is
        port (DVERE, ZAPAL, BEZP_PASY: in
    std_logic;
             VYSTRAHA: out std_logic);
    end VYSTRAHA;
architecture CHOVANIE of VYSTRAHA is
    begin
        VYSTRAHA <= (not DVERE and ZAPAL) or
                    (not BEZP_PASY and ZAPAL);
    end chovanie;
```

```
architecture STRUCTURA of VYSTRAHA is
        -- Declarations
        component AND2
            port (in1, in2: in std_logic;
                  out1: out std_logic);
        end component;
        component OR2
            port (in1, in2: in std_logic;
                  out1: out std_logic);
        end component;
        component NOT1
            port (in1: in std_logic;
                  out1: out std_logic);
        end component;
        -- declaracia signalov použitých na
prepojenie LČ
        signal DVERE_NIE, BEZP_PASY_NIE, B1, B2:
    std_logic;
    begin
        -- Component instantiations statements
        U0: NOT1 port map (DVERE, DVERE_NIE);
        U1: NOT1 port map (BEZP_PASY, BEZP_PASY_NIE);
        U2: AND2 port map (ZAPAL, DVERE_NIE, B1);
        U3: AND2 port map (ZAPAL, BEZP_PASY_NIE, B2);
        U4: OR2  port map (B1, B2, VYSTRAHA);
    end STRUCTURA;
```