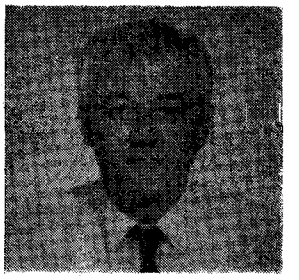
Mikroszámítógépes program logikai függvények minimalizálására

KÁLDI TIBOR
Országos Mérésügyi Hivatal
SZENTIDAY KLÁRA
Kandó Kálmán Villamosipari Műszaki Főiskola



ÖSSZEFOGLALÁS

A kombinációs hálózatok tervezésének egyik alapvető szempontja, hogy az előírt specifikációt a lehető leggazdaságosabban valósítsuk meg. A logikai szintézisnek a mikroelektronikai tervezésben is jelentős szerepe van. Egy- és többkimenetű (max. 10) logikai függvények minimalizálására alkalmas, ZX Speetrum személyi számítógépen futtatható, ASSEMBLY nyelven írt programot készítettünk, ami az M. A. Breuer által bevezetett, irredundáns lefedési algoritmusra épül. Eredményeinket dekóder áramkör tervezési példájával illusztráljuk.

Bevezetés

A mikroelektronikai tervezés fontos elemét képezi a logikai szintézis. A hazánkban is üzembe állított, elsősorban a berendezés-orientált áramkörök fejlesztésére alkalmas Hierarchikus Tervező Rendszer (HTR) a szintézis eljárásokkal megfelelően kiegészítve gyorsabb és hibamentesebb tervezést tenne lehetővé [1], [2]. Munkánk a logikai (Boole-függvények minimalizálásával foglalkozik, és az elkészített program kezdeti lépését jelentheti egy önálló, de a HTR-hez kapcsolódó Logikai Szintézis Alrendszer szoftvernek.

A logikai függvények — mint ismeretes — algebrai úton, grafikus eljárással vagy számítógépes módszerekkel minimalizálhatók. Nagyobb változószám esetén csakis ezen utóbbi megoldás jöhet szóba, mivel az elterjedten alkalmazott Karnaugh-tábla 8—10 változó felett már nehezen áttekinthető és lassan kezelhető. A számítógépes módszerek között figyelemre méltó a Melvin A. Breuer [3] által kidolgozott, ún. *irredundáns lefedési algoritmus-csoport*, ami szemléletessége ós kis tárkapacitás-igénye folytán az egyszerűbb kategóriájú személyi számítógépekre is könnyen adaptálható. Az algoritmusok által definiált műveletek kizárólag karakter füzéreken (string változókon) végrehajtott logikai műveletekre korlátozhatók, amelyek jól illeszkednek az ASSEMBLY nyelvhez.

Az irredundáns lefedési eljárás egyik sajátossága, hogy az áramkörtervező által előírt logikai függvény, az ún. *kezdeti lefedés*, tetszőleges lehet olyan értelemben, hogy nem szükséges a függvényt a normál alakra visszavezetni, mint pl. a Quine-McCluskey numerikus eljárás esetében [4].

Közleményünk ZX Speetrum (48 K) személyi számítógépre kidolgozott minimalizálási programot ismertetet, ami Melvin A. Breuer módszerén alapul. Az eredményül kapott irredundáns lefedés kétszintű ÉS-VAGY, ill. NÉS-NÉS kapus hálózattal valósítható meg.

Beérkezett: 1988. VI. 6. (H)

KÁLDI TIBOR

A Jedlik Ányos Közlekedésgépészeti Technikumot 1958-ban végezte el. 8 évig az EIVRT-ben, a TUNGSRAM RT jogelődjénél dolgozott a tranzisztorgyártás szaklén. 1967-től az OMH Optikai osztályán radiometriai méréstechnikával és fotodetektorok vizsgálataival foglalkozik.

SZENTIDAY KLÁRA

Okl. fizikus, okl. elektronika szakmérnök, egy. doktor. Korábban a Híradástechnikai Ipari Kutató Intézet (a MEV jogelődje) tud. fómunkatársa, jelenleg a KKVMF KATI docense, szakcsoportvezető. Szakmai tevékenysége: tranzisztor strukturavizsgálatok, optoelektronikai félvezető eszközök méréstechnikája, minőségellenőrzés. Oktatási területe a félvezetők konstrukciós számításai, digitális áramkörök.

2. Az algebrai formalizmus

Írjuk fel a logikai függvény kiindulást képező alakját szorzattagok (mintermek, P tagok) összegeként. Ha a függvényt algebrai úton egyszerűsítjük, olyan szorzattagok is előfordulhatnak, amelyekből bizonyos változók kiestek. Az ilyen csonka P tagot az egyszerűbb algoritmizálhatóság érdekében J. P. Roth [5] ún. *kocka* (cube) alakjában fejezte ki. Legyen pl. a $c=\bar{A}_1 A_3$ szorzattag egy háromváltozós függvény egyik tagja, amit a hiányzó A_2 változóval $\bar{A}_1 A_2 A_3 + \bar{A}_1 \bar{A}_2 A_3$ szerint egészíthetünk ki. Az A_2 közömbös változót x-szel, a ponált változót 1-gyel és a negáltat 0-val jelölve, a c kocka

$$c=1\times 0$$

formában írható fel.

A kocka-specifikáció lényege tehát, hogy az n változós Boole-függvényt n dimenziós vektorok (kockák) halmazával adjuk meg. A c vektor összetevői $c=\{0, 1, x\}$ értékűek, és az f függvényt az

$$f=\{c_1, c_2, \ldots, c_m\}$$

kockahalmazzal definiáljuk.

A gépi algoritmusok számos operátort és műveletet értelmeznek a kockák és a kockahalmazok között. Ezek közül a metszet ós az éles szorzat ismertetésére térünk ki.

1. táblázat

A metszet koordináta-táblája

	b_i			
$\cap$	0	1	x
a_i	.0	0	Ω	0
	1	Ω	1	1
	x	0	1	x

2. táblázat

Az éles szorzat koordináta-táblája

	b_i			
#	0	1	x
a_i	0	z	y	z
	1	y	z	z
	x	1	0	z

Metszet. A $c_x \cap c_y$ művelet az 1. táblázat segítségével végezhető el. $c_x = (a_1 a_2 \ldots a_i \ldots a_n)$, $c_y = (b_1 b_2 \ldots b_i \ldots b_n)$ és az eredményül kapott új kocka $c_{xy} = (d_1 d_2 \ldots d_i \ldots d_n)$. Legyen pl. $c_x = 10x1$ és $c_x = x011$. Az azonos pozícióban lévő elemek között elvégezve a kijelölt műveletet, $c_{xy} = 1011$ lesz az eredmény. Legyen most $c_x = 10x0$, ekkor a negyedik elemnél Ω olvasható ki a táblázatból. Ha bármely pozícióban Ω adódik, az eredményül kapott c_{xy} kocka üres halmaz lesz, amit ψ-vel jelölünk. Nem nehéz felismerni, hogy a metszet nem más, mint a kockáknak megfelelő, csonka P tagok szorzata.

Éles szorzat. A P tagos alaknál maradva, a c_x-nek a c_y-nal való éles szorzata

$$c_x \# c_y = P(a) \cdot \overline{P(b)}$$

alapján értelmezhető, ahol a $c_x = P(a)$ és a $c_y = P(b)$ megfeleltetéssel élünk. A 2. táblázat alapján nyerhető C függvény kockahalmaz, ami a következőképpen definiálható:

$C = \psi$ (üres halmaz), ha $a_i \# b_i = z$ minden i-re; $c_x \# c_y = c_x$. ha bármely i-re y adódik.

Egyébként:

$$c_x \# c_y = \bigcup_i (a_1 a_2 \ldots a_{i-1} \alpha_i a_{i+1} \ldots a_n)$$

ahol $\alpha_i = a_i \# b_i$, és ha $\alpha_i = z$, az i-edik elemet törölni kell. Ha α_i 1-gyel, vagy 0-val egyenlő, a kocka ezekkel az értékekkel szerepel az eredményben. $\cup$ a halmazok egyesítésének művelete (unio). Az éles szorzat ez utóbbi változatát világítsuk meg egy példával. Legyen $c_x = 1x0x$ és $c_y = x1x1$. Az eredmény a $C = \{100x,\ 1x00\}$ halmaz lesz. Ezt igazolhatjuk, ha szorzattag alakban írjuk fel a kockákat és a DeMorgan szabályt alkalmazva

elvégezzük a $P(a) \cdot \overline{P}(\overline{b})$ műveletet.
$c_x = P(a) = A_1 \overline{A_3}$ és $c_y = P(b) = A_2 A_4$ megfeleltetéssel:

$$c_x \# c_y = A_2 \overline{A_3}(\overline{A_2}\,\overline{A_4}) = A_1 \overline{A_3}(\overline{A_2} + \overline{A_4}) = A_1 \overline{A_2} \overline{A_3} + A_1 \overline{A_3} \overline{A_4}$$

Ez utóbbi eredmény éppen a C halmaz P tagos alakja. Ellentétben a metszettel, az éles szorzat nem kommutatív művelet, a szorzótényezők sorrendje tehát nem cserélhető fel.

Halmazműveletek. A metszet és az éles szorzat kockahalmazokra is kiterjeszthető. Legyen C_i és C_j két n-dimenziós halmaz. Ekkor

$$C_i \cap C_j = \bigcup_{c' \in C_i} [\bigcup_{c \in C_j} (c' \cap c)] \text{ és}$$

$$C_i \# C_j = \bigcup_{c' \in C_i} [c' \# C_j]$$

ahol $c' \# C_j = (\ldots (c' \# c_1) \# c_2) \# c_3 \ldots) \# c_n)$
itt $C_j = (c_1, c_2, \ldots, c_n)$ értékű.
($c' \in C_i$ jelentése: c' eleme a C_i halmaznak.)

3. Az irredundáns lefedés alforitmusai

Az irredundáns lefedés algoritmusait több, egymást követő lépésre bontva ismertetjük. Az algoritmusok levezethetők a Boole-algebra alaptételeiből és szemléltethetők a Karnaugh táblán.

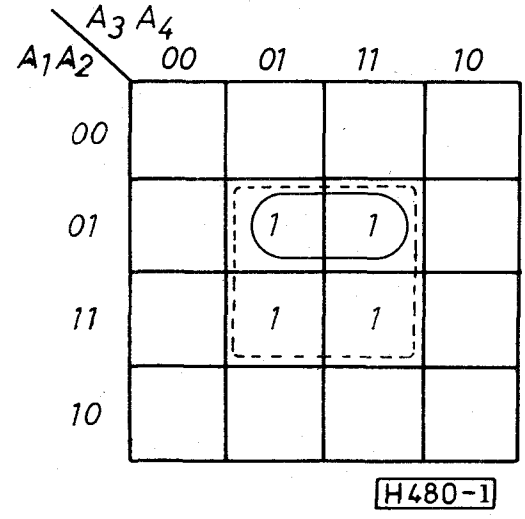
1. lépés: implikáció. Nézzük az 1. ábra példáját A $c_1 = 01x1$ hurok benne foglaltatik a $c_2 = x1x1$ hurokban, vagyis c_1 implikálja c_2-t $(c_1 \to c_2)$. Amennyiben $c_1 \to c_2$ fennáll, $c_1 \# c_2 = \psi$ (üres halmaz) adódik és c_1 elhagyható.

2. lépés: kockatörlés. A 2. ábra kezdeti lefedését tekintve, belátható, hogy a c_5 kocka felesleges, hiszen a többi négy rendre lefedi annak egy-egy celláját. Valamely c kocka akkor törölhető, ha kiemelve azt a C halmazból, a $c \# (C - c)$ művelet eredménye ψ-vel lesz egyenlő. A példánál maradva belátható, hogy

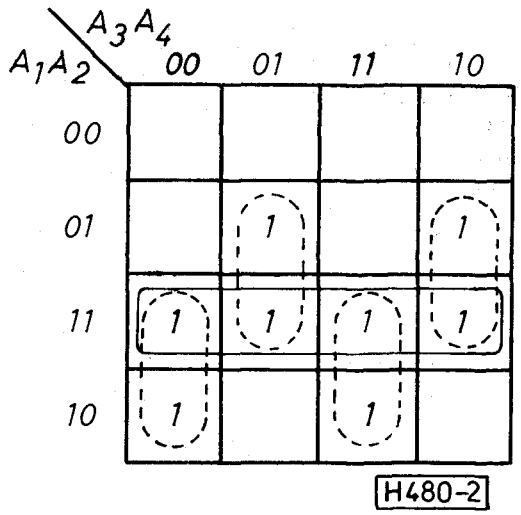
$$(((c_5 \# c_1) \# c_2) \# c_3) \# c_4 = \psi.$$

Megjegyezzük, hogy a vizsgált c kockát követő $(C\text{-}c)$ halmaz kockáinak sorrendje az éles szorzatok végzésekor tetszőleges lehet.

3. lépés: elemtörlés. A 3. ábrán látható példa alapján ezt a lépést inkább a „hurokbővítés" névvel illethetnénk. Az elemtörlésnél tehát azt a folyamatot algoritmizáljuk, amikor a kezdeti lefedést nagyobb, több cellát befogó hurokkal helyettesítjük



1. ábra. Implikáció: a $c_1 = 01x1$ kockának megfelelő hurok benne foglaltatik a $c_2 = x1x1$ kockának megfelelő hurokban



2. ábra. Kockatörlés: A c_5=11xx kockának megfelelő hurok teljesen fedve van a c_1=1x00, c_2=x101, c_3=1x11 és c_4=x110 kockáknak megfelelő hurok egy-egy cellája által

a Karnaugh táblán. Ezáltal a kockákban a {0, 1} elemek x-re változnak. Az algoritmus lépései a következők.

Vesszük az első c kockát és annak {0, 1} elemeit sorszámmal látjuk el. Az $i=1$ elemet negáljuk.

Az így nyert $\overline{c(i=1)}$ kockát kiemelve a C halmazból, éles szorzat-sorozatot végzünk. Amennyiben

$$\overline{c(i=1)} \# (C-c) = \psi$$

adódik, a c kocka $i=1$ elemét x-re írjuk át. A c kocka ezután átírt formájában szerepel a C halmazban. Az eljárást valamennyi i-re ($i \leq n$) elvégezzük, és a C halmaz valamennyi kockáját megvizsgáljuk a bemutatott módon.

Visszatérve a 3. ábra példájához, az elemnegálás azt jelenti, mintha a kiválasztott kockát reprezentáló hurkot valamelyik irányba eltoltuk volna. Ha a léptetett hurok által befogott cellákat a többi hurok (kocka) lefedi, a bővítés elvégezhető.

4. lépés: közös implikánsok keresése. A k kimenetű kombinációs hálózat k darab logikai függvényből álló függvényrendszerrel jellemezhető. A hálózat tervezésének egyik lehetősége, hogy az egyes kimenetekhez tartozó logikai függvényeket különkülön, egymástól függetlenül minimalizáljuk. Gazdaságosabb megoldást kaphatunk azonban, ha az egymástól független minimalizálást követően megkeressük azokat az implikánsokat, amelyek két vagy több függvényben közösek.

Nézzünk egy példát a közös implikánsokra.

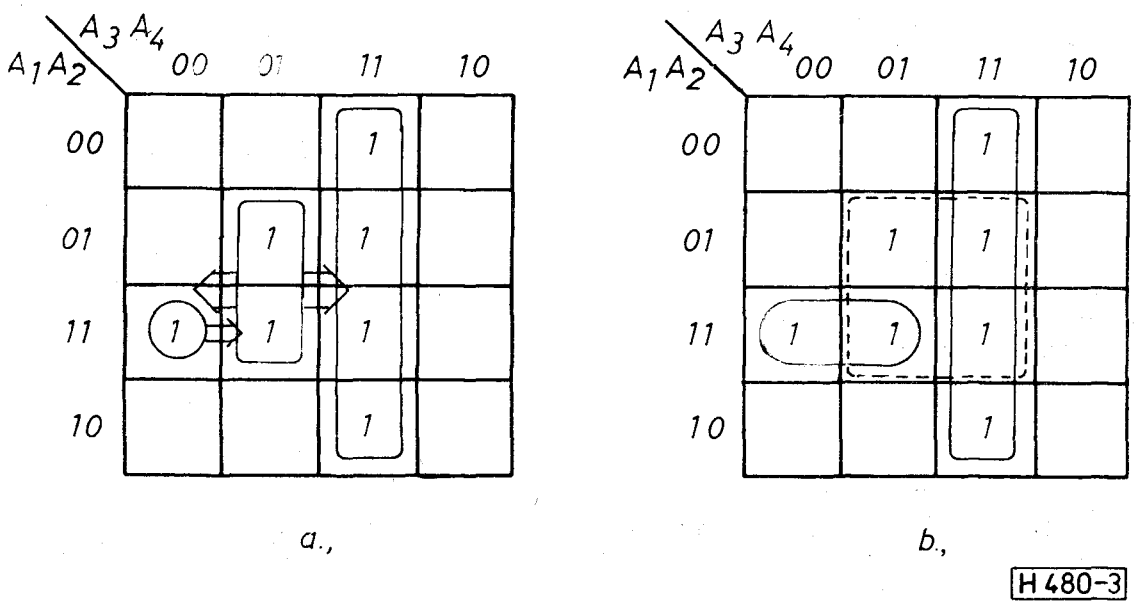
Legyen $f^1 = A_2A_3 + \overline{A_1}A_2$ és $f^2 = A_1\overline{A_3} + A_1A_2$. Az f'-hez tartozó kezdeti lefedést a 4a. ábra, míg f^2 kezdeti lefedését a 4*b*. ábra szemlélteti. Kockákkal kifejezve: O_1={x11, 01x}; C_2={1x0, 11x} f^1-et tehát a C_1 és f^2-t a C_2 halmaz reprezentálja. Képezzük e két halmaz metszetét a $C_1 \cap C_2$ művelettel. Az 1. koordináta-táblázatot használva:

$$C_1 \cap C_2 = \text{x11} \cap \text{11x} = \text{111}$$

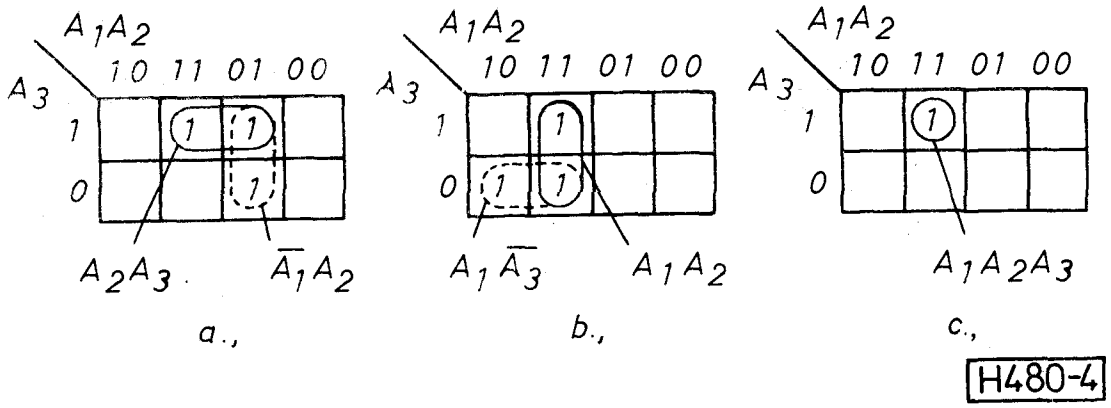
adódik, a többi párosítás ψ-t eredményez. A metszet eredménye, ami nem más, mint az f^1 és f^2 függvények szorzata, a 4. c. ábrán látható. Mind f^1-be, mint f^2-be beírható a közös implikáns, és cserébe elhagyható f^1-ből az A_2A_3 és f^2-ből az A_1A_2 szorzattag, mivel ezek második celláját mind f^1, mind f^2 esetében a másik hurok lefedi. Végeredményben

$$C_1 = \{111, 01x\} \text{ és } C_2 = \{111, 1x0\}$$

adódik, ami egyszerűsíti a realizálást, hiszen az 111 kocka mint közös kapu, mindkét kimenethez csatlakoztatható.



3. ábra. Elemtörlés: A c_2=x101 kocka i=2 elemét negálva $c_2(i=2)$=x111 adódik, amit a c_3=xx11 kockának megfelelő hurok lefed. Hasonlóképpen követhető a c_1=1100 kocka elemcseréje is.



4. ábra. Példa közös implikáns keresésére:
a) az f^1 függvény kezdeti lefedése
b) az f^2 függvény kezdeti lefedése

Az algoritmizálás a következőképpen történik. Meg kell vizsgálni a kezdeti lefedést jelentő, k számú halmaz egymással való összes lehetséges metszetét. Ha pl. a C_i halmazban szereplő c_x kockának a C_j halmazban lévő c_y kockával való metszete ψ-től különböző, $c_x \cap c_y = c_{xy}$ új kocka keletkezik. Ezt követően meg kell vizsgálni, hogy a C_i halmazban a c_x és a C_j halmazban a c_y helyettesíthető-e a c_{xy}-nal. Ha mindkét halmazra teljesülnek a lefedési feltételek, c_x és c_y törlendő és mind C_i-be, mind C_j-be c_{xy} írandó. A halmazok egymással való metszetének műveletét mindaddig meg kell ismételni, amíg új kocka keletkezik.

Valamely kocka elhagyásának vizsgálatához elő kell állítani a metszettől különböző összes variánst. Hogy azokat lefedi-e a halmaz többi eleme, arról a 2. lépésnek megfelelő éles szorzat műveletsor ad felvilágosítást.

Költségfüggvények. A logikai függvények megvalósításának költségeit korábban a szükséges kapuk száma alapján határozták meg. A mikroelektronikában, ahol egyetlen félvezető lapkán helyezik el az egész áramkört, nagyobb jelentősége van a kapubemenetszám és a FAN OUT csökkentésének. Általában többféle költségfüggvényt definiálnak, amelyek a fenti jellemzőket tartalmazzák.

4. Programszerkesztés

Kiindulási feltételek:

— a minimalizálandó logikai függvényt szorzatok összegeként kell megadni;
— tetszőleges kezdeti lefedés választható a csonka P tagokat kockákkal kifejezve;
— egyszerűsített lefedésből indulunk ki $C \rightarrow C_0 \cup$ DC alapján, ahol C_0 az igaz (care) és DC a közömbös (don't care) halmaz;
— a program többkimenetű függvények minimalizálására készült, amely — értelemszerűen — magában foglalja az egykimenetű kombinációs függvények esetét is.

A program bemenet, kimenet ós interaktív tömbjét BASIC nyelven írtuk meg, míg a feladatmegoldó rész ASSEMBLY nyelven készült.

Bemeneti jellemzők

Az egyes kockák elemeinek n száma maximálisan 20 lehet azzal a megkötéssel, hogy a közömbös változók (az x-értékek) száma nem lehet nagyobb 16-nál. A függvénykimenetek k száma maximálisan 10 lehet. Egy-egy kimenethez max. 64 darab kocka tartozhat, azonban a kockák összes száma nem lépheti túl a 255-öt. Az adatbevitel kockánként történik a {0, 1, x} szimbólumok bevitelével, majd feltüntetjük, hogy a beírt kocka mely kimenetekhez tartozik, ill. nem tartozik: az aktív kimenetet 1-gyel, és az inaktív kimenetet x-szel jelölve.

Kimeneti adatok

A program első lépésben a kimenetenkénti minimalizálást végzi el. Amennyiben egykimenetű logikai függvényt vizsgálunk, a feladatmegoldás véget is ért. Többkimenetű függvény esetén a program második lépésben elvégzi a közös implikánsok megkeresésén alapuló minimalizálási eljárást is.

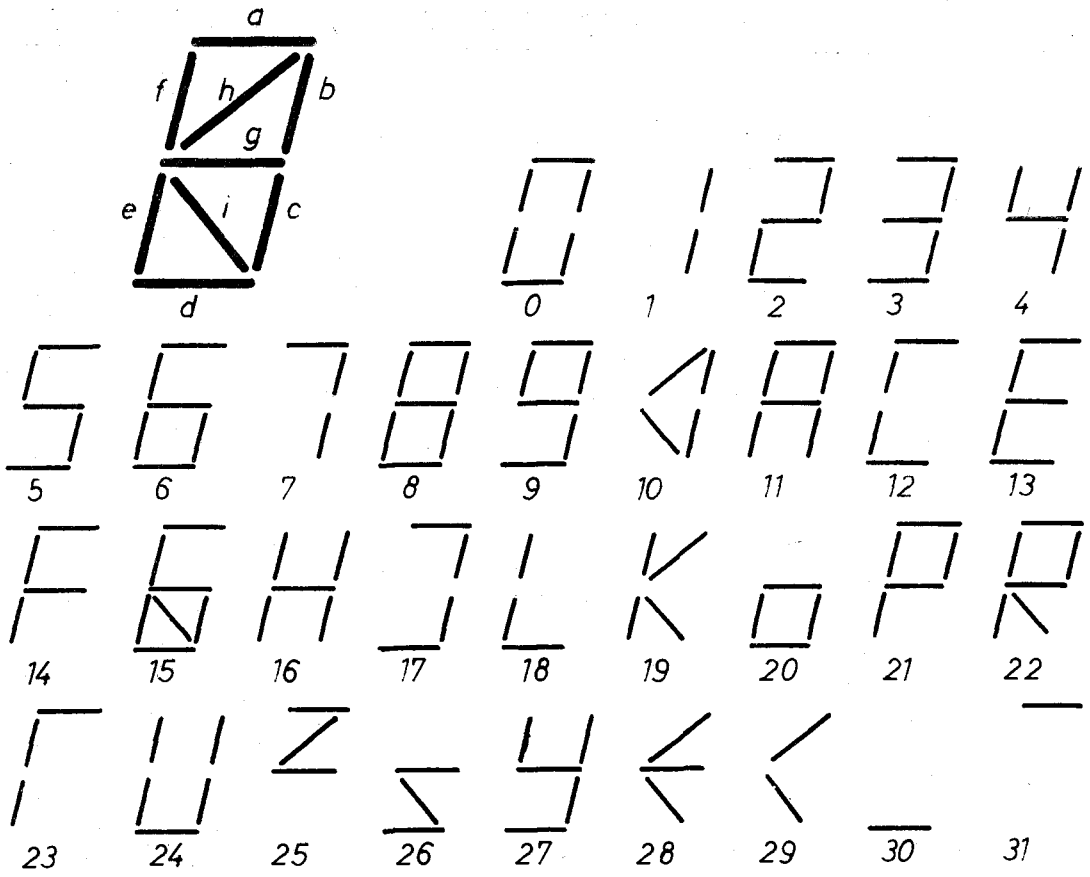
A program kiszámítja az alábbi költségfüggvényeket:

Kapubemenetek száma
Kapuk száma
Max. FAN IN
Max. FAN OUT

ÉS-VAGY realizáció esetében a Max. FAN IN megadja, hogy maximálisan hány bemenetű ÉS kapura van szükség. A Max. FAN OUT azt jelenti, hogy egy ÉS kapu kimenetre maximálisan hány darab VAGY kapubemenet csatlakozik.

Perifériák

A program mind magnetofonkazettás, mind hajlékony mágnes lemezes változatban elkészült. Ez azt jelenti, hogy mind a bevitt adatok, mind pedig a részeredményként és végeredményként kapott kockahalmazok és költségfüggvények háttértárra kivihetők, ill. onnan betölthetők. A ZX Spectrum géphez Seikosha GP 50S (ill. 100S) printer illeszthető és az eredmények oldalanként kinyomtathatók.



5. ábra. **9-szegmenses kijelző képe és a megjeleníteni kívánt karakterkészlet**

5. Tervezési példa

Szintézisprogramunkat 9-szegmenses kijelző dekóder tervezési példájával illusztráljuk. A 9-szegmenses karaktermegjelenítő képe és szegmenseinek jelölése az 5. ábrán látható. Az ábra feltünteti az általunk példaként választott karakterkészletet is, amelyet a kijelző megjelenít, ha a dekóder bemenetére bináris alakban az egyes karaktereknél feltüntetett számnak megfelelő jelsorozatot adjuk. Pl. a 7-es számjegy a 00111 jelsorozattal gyújtható ki, ahol a 0 a logikai L-szintet és az 1 a logikai H-szintet jelöli. A 9-szegmenses kijelzővel már csaknem valamennyi betű, és sokféle szimbólum is megjeleníthető, míg az ismertebb 7-szegmenses változatot inkább csak a számjegyek kiíratására alkalmazzák [6]. A teljesség kedvéért a dekódert engedélyező bemenettel (EN) és lámpavizsgáló bemenettel (LT) is elláttuk. Ha az EN bemenet L-szintet kap, a kijelzés letiltott, míg H-szint

3. táblázat

A dekóder logikai függvényének kezdeti lefedése

Bemenetek								Kimenetek							
EN	LT	E	D	C	B	A	i	h	g	f	e	d	c	b	a
0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1	0	x	x	x	x	x	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	x	x	x	x	1	1	1	1	1
1	1	0	0	0	0	1	x	x	x	x	x	x	1	1	x
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
														.	.

4. táblázat

Az irredundáns lefedés végső eredménye

Bemenet	Kimenet	Bemenet	Kimenet
1x01111	1xxxx1xxx	1x01x11	xxxx1xxxx
1x10011	11xx1xxxx	1x011xx	xxxx1xxxx
1x10110	1x1xxxx11	1x101xx	xxxx1xxxx
1xx1010	1xxxxxxxx	1xx0x10	xxxx1xxxx
1x1110x	1xxxxxxxx	1x10001	xxxxx1xxx
10xxxxx	111111111	1x10100	xxxxx11xx
1x11100	x11xxxxxx	1x0x000	xxxxx1xx1
1x01010	x1xxxxxxx	1x0x101	xxxxx1xxx
1x11x01	x1xxxxxxx	1xx1000	xxxxx11xx
1x00x10	xx1xx1xx1	1x1x010	xxxxx1xxx
1x00011	xx1xx1xxx	1x11x10	xxxxx1xxx
1x0100x	xx1xxxxxx	1x01x0x	xxxxx1xx1
1x0x110	xx1xxxxx1	1x001xx	xxxxxx1xx
1x10x00	xx1xxxxxx	1x010xx	xxxxxx1xx
1x1101x	xx1xxxxxx	1x0xx11	xxxxxx1x1
1x01xx1	xx11xxxxx	1xx000x	xxxxxx1xx
1xx010x	xx1xxxxxx	1x00x00	xxxxxxx1x
1xx10x1	xx1xxxxxx	1x00x11	xxxxxxx1x
1xx0101	xxx1xxxx1	1x10x01	xxxxxxx1x
1x11011	xxx1x111x	1x0x0xx	xxxxxxx1x
1x0x1x0	xxx1xxxxx	1x1x001	xxxxxxxx1
1x10x1x	xxx1xxxxx	1xxx111	xxxxxxxx1
1xxx000	xxx11xx1x		

5. táblázat

Költségfüggvények

	Kezdeti lefedés	Kimenetenkénti min.	Végső eredmény
Kapubemenetszám	371	321	290
Bemeneti kapuk sz.	33	53	45
Kapuk száma	42	62	54
Max. FAN IN	7	6	6
Max. FAN OUT	9	9	9

esetén engedélyezett. Ha az LT bemenet L-szintet kap, a kijelző valamennyi szegmense aktivizálódik. A logikai függvény kezdeti lefedésének néhány sorát a 3. táblázat tartalmazza. A többi sor az 5. ábra alapján egyszerűen meghatározható.

A minimalizálás végső eredményét a 4. táblázat tünteti fel. A dekóder működését egyszerű BASIC nyelvű program segítségével szimuláltuk, ellenőrizve a 4. táblázatban foglalt eredményeket.

Az 5. táblázat a költségfüggvényeket adja meg a kezdeti lefedés, a kimenetenkónti minimalizálás ós a közös implikánsok megkeresése után. Mint a 4. és 5. táblázatból látható, a végső eredményben lényegesen lecsökkent a kapubemenetek száma a kezdeti lefedéshez képest, a kapuszám azonban megnőtt. A program teljes futási ideje kb. 15 s nagyságú volt.

A programot számos példával teszteltük és tapasztaltuk, hogy a futási idő az adatstruktúrától is lényegesen függ. Vizsgálataink és becsléseink arra utalnak, hogy maximális terjedelmű adatbevitellel a legkedvezőtlenebb esetben mintegy félórás futási időre lehet számítani.

6. Köszönetnyilvánítás

Szerzők köszönetet mondanak Keresztes Péter főiskolai docensnek, amiért a probléma érdekességére felhívta figyelmüket.

IRODALOM

[1] Automatikus struktúra szintézis a Hierarchikus Tervező Rendszerhez. Tanulmány. Készült a Mikroelektronikai Kormánybiztos ós a MEV megbízásából 1985-ben. Szerzők: *Keresztes Péter, Ágotai István stb.*

[2] Design Automation of Digital Systems. Chapter One, R. J. Preiss. 1972 by Prentice-Hall, Inc.

[3] Design Automation of Digital Systems. Chapter Two, *Melvin A. Breuer.* 1972 by Prentice-Hall, Inc.

[4] *Janovics-Tóth:* A logikai tervezés módszerei 2., javított kiadás. Műszaki Könyvkiadó, Bp. 1976. 4. fejezet.

[5] *J. P. Roth:* „Algebraic Topological Methods for the Synthesis of Switching Systems. I." Trans. Amer. Math. Soc., Vol. 88 (1958), pp. 301—326.

[6] *Kis Halas—Mészáros—Szentiday:* Optoelektronikai kijelzők és megjelenítők. Műszaki Könyvkiadó' Bp. 1984. 1. fejezet.