DÉNES GYÖRGY
INFELOR

Logikai terv leírása és kódolása, valamint részekre bontása az ESZTER számítógépes programrendszerben

ETO 681.32.06 ESZTER

Kétségtelen, hogy a digitális készülékek tervezését és gyártását segítő programrendszer kidolgozásában legizgalmasabb az ún. algoritmikus problémák (részekre bontás, elhelyezés, fóliatervezés) megoldása. Ezen témákról a szaklapokban és konferenciákon számos publikáció látott napvilágot, míg az általunk realizálásnak nevezett programról csak elvétve hallhattunk. Realizáló programunk elsődleges szerepe tulajdonképpen egy olyan kapcsolat megteremtése a tervező és a programrendszer között, mely mindkettő számára előnyös, azaz lehetővé kell tenni a megvalósítandó logikai terv egy egyszerű, könnyen kezelhető leírását, majd ezt olyan kódolt formában kell tárolni, hogy a későbbi tervező programok kiindulási adatbázisát szolgáltassa.

Munkánk során rá kellett jönnünk arra, hogy ha programrendszerünket el akarjuk fogadtatni a digitális készülékek tervezőivel, a logikai terv leírási szabályainak megállapításakor alkalmazkodnunk kell a kézi tervezési módszerekhez. Ezért a következő 4 alapelvet fogadtuk el:

1. A realizálás

1.1 A logikai terv leírása

1. Feltételezzük, hogy egy nagyobb digitális készülék logikai terveit — a rendszertervek elkészülte után — több tervezőből álló csoport tagjai különkülön készítik, funkcionális egységenként. A tervezők egymástól függetlenül szeretnének meggyőződni leírásuk formai és funkcionális helyességéről. A teljes rendszer összeállítására akkor kerül sor, ha már mindenegyes funkcionális egység hibátlan leírása elkészült.

2. A logikai tervező a funkcionális egységét egymással kapcsolódó ún. fekete dobozok segítségével adja meg, melyek mindegyike az áramkörtervezők által tervezett elektronikus áramköröket tartalmaz. Előfordul, hogy bizonyos feketedoboz-kombinációk többször szerepelnek a tervben, célszerű megengedni ezek egyetlen fekete dobozzá történő összevonását.

3. A fekete dobozokat összekötő vonalakat a tervező mnemonikus azonosítókkal látja el. Lehetővé kell tenni, hogy a leírás ezekkel a mnemonikus azonosítókkal történjék, fekete dobozként, megkülönböztetve a be-, ill. kimenetre kapcsolódó vonalakat.

4. A fekete dobozok egy része kapuáramkör jellegű — azaz bemenetei felcserélhetők — míg másik része fel nem cserélhető bemeneteit a tervező mnemonikus jelekkel különbözteti meg.

A felsorolt négy elv megvalósítása, ha el is tekintünk a sok apró problémától, komoly gondot okozott. Felismertük viszont azt, hogy az ICL—System 4/50 Program Trials System (program próba rendszer) rutinjait nagyon jól használhatjuk céljaink eléréséhez, ha olyan makrokönyvtárat hozunk létre, melynek elemei megfelelnek a logikai terv fekete dobozainak, és amelyek a USERCODE translator által történő kifejtés és fordítás után a további feldolgozás számára alkalmas adatstruktúrát szolgáltatnak. Egy funkcionális egység leírása tehát makroutasítások sorozata lesz, amelyekben aktuális paraméterek a tervező által adott vonalazonosítók. A makro neve az általa reprezentált fekete doboz mnemonikus azonosítója. (Pl. egy kétbemenetű ÉS kapu makrojának neve: AND2.)

A funkcionális egység — s ezzel együtt a logikai terv — megadását jelentő makroutasításokat USERCODE kódlapon írjuk, a szimbolikus gépi kódú nyelv szabályainak megfelelően:

- a makroutasítás szimbólummezejébe a fekete doboz által generált vonal azonosítóját (betűvel kezdődő, maximálisan 8 jelből álló alfanumerikus jelsorozat);
- a műveleti kód helyére a fekete doboz, ill. a makro nevét;
- majd az operandusmezőbe a bemenetekre kapcsolódó vonalak azonosítóit kell írni. Kaputípusú fekete doboz esetén a makrot pozícionálisnak, a fel nem cserélhető bemenetűeket kulcsszavasnak definiáljuk. Az előbbi esetben a benetekhez kapcsolódó vonalak azonosítóit egymás után írjuk vesszővel elválasztva, míg az utóbbiban ezek a kulcsszó utáni egyenlőségjelet követik.

Két speciális makro operandusmezejében a funkcionális egységen kívül generált (INPUT) ill. a funkcionális egységéből elmenő (OUTPUT) vonalakat soroljuk fel, egy harmadik makroban pedig a funkcionális egységet azonosítjuk (FUNKE).

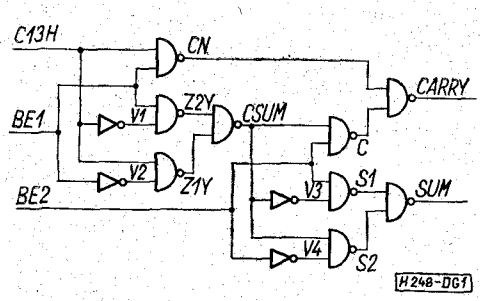
Egy vonal akkor definiált, ha azonosítója vagy az INPUT-makro operandus mezejében vagy valamely fekete doboz makro szimbólummezejében megjelenik. Ha az előbbiekben ismertetett szabályok szerint leírt utasítássorozatot kiegészítjük egy START és egy END fordítóprogram-utasítással, akkor a USERCODE translator számára értelmes modult kapunk. Egy kisméretű logikai terv és annak leírása az 1. ábrán látható.

1.2 A program-próba rendszer kihasználása

A rendszer fő funkciói: a módosítás, fordítás és az összekapcsolás.

A *módosító* (AMENDER) rutin a tervezők által leírt és lyukkártyára lelyukasztott funkcionális egységeket külön-külön elhelyezi az erre a célra fenntar-

Bérkezett: 1973. X. 1.




```
FULAD   START
        FUNKE 1
        INPUT  C13H, BE1, BE2
        OUTP   CARRY, SUM
CN      NAND2  C13H, BE1
V1      NOT1   C13H
V2      NOT1   BE1
Z2Y     NAND2  BE1, V1
Z1Y     NAND2  C13H, V2
CSUM    NAND2  Z2Y, Z1Y
V3      NOT1   CSUM
V4      NOT1   BE2
C       NAND2  CSUM, BE2
S1      NAND2  BE2, V3
S2      NAND2  CSUM, V4
SUM     NAND2  S1, S2
CARRY   NAND2  CN, C
        END
```

1. ábra. Logikai terv leírása az ESZTER programrendszerben

tott mágneslemez ún. forráskönyvtárában, az utasításokat sorszámozza és sornyomtatón kiírja. A mágneslemezen levő leírás módosítását vagy leírását ugyanez a rutin végzi.

A *fordító rutin* (USERCODE TRANSLATOR) elvégzi a makro kifejtését (generálását) azaz a leírás kódolását. A leírásban előforduló hibákat és azok jellegét kiírja, és létrehozza az előbbi mágneslemez ún. tárgykönyvtárában a funkcionális egység kódolt változatát.

Végül az *összekapcsoló-rutin* (COMPOSER) összeállítja a realizáló programot ugyanazon mágneslemez ún. ideiglenes futtatási könyvtárában úgy, hogy a funkcionális egységek a programnak szegmensei lesznek.

1.2.1 A realizáló program futtatása

A realizáló program futtatása során megtörténik a funkcionális egységek közötti kapcsolatok elemzése, és — a teljes logikai tervet tekintve — kialakul az elsődleges be-, ill. kimenő vonalak listája. Sornyomtatón kerül kiírásra a tervben előforduló fizikai elemek típusonkénti és funkcionális egységenkénti darabszáma, valamint minden egyes vonal terhelési állapota. Ezen utóbbi információ alapján kitűnik az esetleges túlterhelés, és a szükséges módosítás (pl. egy elem duplikálása) elvégezhető. A tervező számára kiírt adatokon kívül a program mágneslemezen előállítja a logikai terv alapadattömbjét (AAT).

1.2.2 A makro-definiáló nyelv adta lehetőségek

A realizáló program működésének alapját képező makrokönyvtár elemeit a makro — definiáló nyelv segítségével írhatjuk le. A makrokönyvtár bázisát valamely fizikai építőelem-készlet (esetünkben TEXAS SN74 sorozat) alapelemei képezik. A háromszoros mélységű makrogenerálási lehetőséget kihasználva, további makrók építhetők be, melyek a tervező szempontjából első, ill. második szintű funkcionális elemek lesznek. A funkcionális elemek ún. belső vonalainak azonosítói a makrogenerálás során jönnek létre és minden esetben egy előre meghatározott speciális karakterrel kezdődnek.

Problémát jelentett a több kimenettel rendelkező fekete dobozok leírása, mivel a makroutasítás szimbolummezejében generált vonalként csak egyetlen vonalazonosító írható. Ezen fekete dobozok makroiból az egyes kimenetek által generált vonalak egyedi azonosítói makrogenerálás kor állnak elő, mikor is a szimbólummezőbe írt jelsorozathoz előre meghatározott karaktereket adunk.

A makrokönyvtár karbantartását (bővítés, törlés, katalóguskiírás stb.) rendszerprogramokkal végeztetjük.

2. Szétosztás kártyákra

2.1 *A kártyárabontás feladata*

A realizáló program futtatása után az AAT-ben mágneslemezen található a digitális készülék logikai tervének egy kódolt és hibáktól mentes leírása. A kártyárabontó program feladata a leírásban szereplő alapelemek szétosztása kisebb egységekbe — esetünkben nyomtatott áramkörű kártyákra — úgy, hogy közben figyelembe vegyünk, ill. kielégítsünk bizonyos követelményeket, úgymint:

a) Az egy egységben levő elemek halmaza megadott számú *(T)* tokkal lefedhető legyen;
b) Az egy egységet jelentő kapcsolás ki/bemeneti csatlakozóigénye ne legyen nagyobb a megengedett felső korlátnál *(C)*;
c) Az egységek közötti összeköttetések száma legyen minimális;
d) Az egységek száma legyen minimális;
e) Minden alapelem csak egy egységben szerepeljen;
f) Funkcionálisan összetartozó alapelemek ugyanabba az egységbe kerüljenek (tesztelés és karbantartás megkönnyítése).

Ezen követelmények egyidejű megvalósítása egy számítógépes programrendszerben rendkívül nehézkes, különös tekintettel arra, hogy ellentmondásokat is tartalmaznak. Valószínűleg ez az oka, hogy az irodalomban minden igényt kielégítő kártyárabontó eljárást ismertető leírással nem találkoztunk. Az általunk megvalósított program sem akar a kézi tervezési módszerek helyére lépni, jelentősége akkor mérhető fel, ha számítógépes tervezési rendszerünk tagjaként szemléljük.

2.2 Kártyárabontó algoritmusok

Lawler [1] az elektronikus építőelemek feladatát egy klasszikus kapcsoláselméleti problémához való hasonlóság alapján kívánja megoldani, a logikai függ-

vények minimális minterm-reprezentációjának előállításához hasonlóan. Ugyancsak Lawler talált egy egyszerű algoritmust faszerkezetű gráfok olyan szétosztására, amelyben a csoportok közötti összeköttetések okozta maximális késleltetés minimális [2].

A kártyárabontási módszerek nagy része a „maximális összekötöttség, minimális vágás" elvét alkalmazza, valamilyen kritériumfüggvényen keresztül [3, 4].

A kártyárabontó programunk magját képező eljárás a — sok más optimalizálási probléma megoldására is használt — páronkénti cserék módszerén alapul. Kernighan és Lin [5] gráfok szétosztására alkalmazta ezt a módszert, egy olyan heurisztikus eljáráson belül, mely elég hatékony egy optimális eredmény elérésére, és elég gyors ahhoz, hogy segítségével nagyméretű feladatok is megoldhatók legyenek.

Eljárásuk egy 2n csúcsú gráfot két független n csúcsú $\mathbf{A}=\{a_1, a_2, \ldots, a_n\}$ és $\mathbf{B}=\{b_1, b_2, \ldots, b_n\}$ algráfra oszt szét úgy, hogy A és **B** csúcsai között az összeköttetések száma minimális legyen.

Eljárásuk azon a tényen alapul, hogy ha adott egy tetszőleges A és **B** felosztás, akkor létezik A-nak egy olyan A' és **B**-nek egy olyan **B'** algráfja, melyek felcserélésével optimális szétosztást kapunk. Kiindulási alapjuk tehát egy tetszőleges A és **B** felosztás lesz, melyek között az összeköttetések számát *(S)* igyekszenek lecsökkenteni. Az első lépés egy olyan $a_1' \in \mathbf{A}$ és $b_1' \in \mathbf{B}$ elempár megkeresése, melyek felcserélésével *S*-ben a legnagyobb mértékű csökkenés — jelöljük ezt g_1-gyel — érhető el. Az *a'* és *b'* párok keresését *n*-szer végezzük el úgy, hogy minden elemet csak egyszer választunk ki s eredményül az

$$a_1', a_2', \ldots, a_n'; \; b_1', b_2', \ldots, b_n', \text{ ill. } g_1, g_2, \ldots, g_n$$

sorozatot kapjuk. A g_i értékek negatívok is lehetnek, és

$$\sum_{i=1}^{n} g_i = 0,$$

mivel ha A és **B** minden elemét felcseréljük, c értéke nem fog változni.

Keressük meg azt a $k<n$ értéket, melyre

$$\sum_{i=1}^{k} g_i = \text{maximum}.$$

Ha most a tetszőleges kezdeti A, **B** felosztásból adódott

$$\mathbf{A'}=\{a_1', a_2', \ldots, a_k'\} \quad \text{és} \quad \mathbf{B'}=\{b_1', b_2', \ldots, b_k'\}$$

halmazokat felcseréljük, *S* értékének csökkenése maximális lesz. Az így létrejött szétosztást kiindulásnak tekintve újabb *a'* és *b'* kiválasztási sorozatot indítunk.

Az eljárás akkor fejeződik be, ha a maximális csökkenésre 0 vagy negatív értéket kapunk, amikor is egy helyi optimumhoz jutottunk.

2.3 Az ESZTER20-ban felhasznált eljárás

Az ismertetett eljárás alkalmazható a kártyárabontási feladat megoldására is, ha a szétosztandó logikai tervet olyan gráfnak tekintjük, melynek csúcsai az integrált áramköri tokok. Mivel a logikai terv leírása alapelemekkel történik, a szétosztás előtt az alapelemeket tokokba kell válogatni a köztük levő kapcsolatok vagy a tervező előírásai alapján. Ezek után az N számú tokot olyan csoportokba kell szétosztani, hogy egyik csoport se tartalmazzon *T*-nél több tokot, és egyik csoportnak se legyen *C*-nél nagyobb csatlakozóigénye. A megoldást a következő lépéseken keresztül érjük el:

1. Legyen $n=N$ és $t=T$:
2. Veszünk egy tetszőleges
   $\mathbf{A}=\{a_1, a_2, \ldots, a_{n-t}\}$ $\mathbf{B}=\{b_1, b_2, \ldots, b_t\}$
   felosztást;
3. Páronkénti cserével képezzük az **A** és **B** optimális felosztást;
4. Meghatározzuk a **B** csoport csatlakozóigényét, *S*-el;
5. Ha $S>C$ akkor $t=t-1$, $n=n+1$ helyettesítéssel és **B** valamelyik elemének A-ba történő áthelyezésével visszatérünk a 3. lépéshez;
6. Ha $S<C$, egy kártyát kialakítóttunk, folytatás a 7. lépéssel;
7. Ha $n>t$, akkor $n=n-t$, $t=T$ helyettesítéssel visszatérünk a 2. lépéshez;
8. Ha $n>0$, az A halmaz elemeit **B**-be tesszük át, majd $t=n$ és $n=0$ helyettesítéssel visszatérünk a 4. lépéshez;
9. ha $n=0$ az eljárás befejeződött.

A 3. lépésben jelzett optimális szétosztás keresése egy — súlyozással felállított — összeköttetési mátrix alapján történik úgy, hogy a legerősebb kapcsolatot az ugyanazon funkcionális elemhez tartozó alapelemek közötti összeköttetés jelenti. Ennél kisebb a „húzóerő", ha a vonal azonos funkcionális egységbe tartozó alapelemeket köt össze. Végül a legkisebb súlyozást a funkcionális egységek közötti kapcsolatoknak adjuk.

2.3.1 A tervezői beavatkozás lehetőségei

Automatikus kártyárabontó eljárásunkat nem tekintjük tökéletes megoldásnak, ezért programunkban minél több lehetőséget kívántunk adni a tervezők beavatkozására. Az alábbiakban ezeket röviden összefoglaljuk:

a) Az automatikus kártyárabontás előtt a logikai tervből kijelölhetők olyan alapelemek, melyeket a tervező egy tokban, ill. egy csoportban kíván tartani;

b) előre kijelölhetők teljes, ún. szabványos kártyák;

c) a futási idő csökkentése érdekében a kártyárabontás funkcionális egységenként is elvégeztethető;

d) a kártyárabontás eredményei módosíthatók tokok és/vagy alapelemek kártyák közötti cserélgetésével.

3. Összefoglalás

A cikkben ismertetésre kerültek az ESZTER20 programcsomagot alkotó

— logikai terv leírását értelmező és kódolását végző, valamint a
— kártyárabontást elvégző programok.

A programok lefuttatásakor létrejön az az adathalmaz, amelynek alapján a kártyatervező programok elhelyezhetik az építőelemeket, megtervezhetik a fólia-utakat, és lyukszalagokat állíthatnak elő a gyártás vezérléséhez. Programjainkkal nem adunk a prob-

lémákra automatikus megoldási lehetőséget. Eredményeik csak akkor közelítik meg vagy érik el a hagyományos tervezési módszerek eredményességét, ha a tervező végig aktívan résztvesz a munkában, és a számára nyújtott irányítási és beavatkozási lehetőségeket célszerűen kihasználja.

IRODALOM

[1] E. *L. Lawler:* Electrical Assemblies with a Minimum Number of Interconnections. IEEE Transactions on Electronic Computers, Vol. EC—11, February 1962, pp. 86—88.

[2] *E. L. Lawler—K. N. Levitt—*J. Turner: Module Clustering to Minimize Delay in Digital Networks. IEEE Transactions on Computers, vol. C—18, January, 1969. pp. 47—57.

[3] *A. J. Stone:* Partitioning of Logic into Physical Entities. Proceedings of SHARE Design Antomation Workshop. 1966.

[4] *H. A. Nidecker—W. F. Simon:* Logic Partitioning — Component Assignment. Proceeding of ACM National Conferenee, 1968.

[5] *B. W. Kernighan—S. Lin:* An Efficient Heuristic Procedure for Partitioning Graphs. The Bell System Technical Journal, February, 1970.