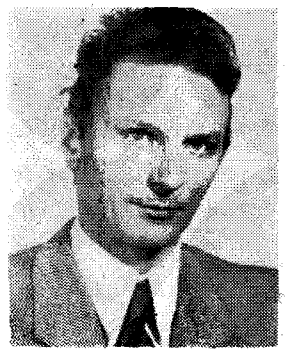
Layout visszafejtő program cellás tervezésű integrált áramkörökhöz

DR. SZÉKELY VLADIMÍR—BAJI PÁL—DR. MASSZI FERENC
KERECSENNÉ DR. RENCZ MÁRTA
BME Elektronikus Eszközök Tanszék
KÓNYA ILONA
BME Villamoskari Matematika Tanszék
DR. KOLTAI MIHÁLY
BME Elméleti Villamosságtan Tanszék



ÖSSZEFOGLALÁS

A cikkben bemutatott program — a CELLINEX — cellás tervezésű integrált áramkörök layout-jából megállapítja a cellák között megvalósított összeköttetéseket. Ennek alapján generálja az áramkör logikai leírását és ellenőrzi, hogy a megvalósított összeköttetések megengedettek-e, illetve nincsenek-e triviális hiányok stb. A cikk ismerteti a program jellemzőit és a visszafejtés legfontosabb algoritmusait. Bemutat néhány eredményközlési formát és útmutatást ad a program használatára vonatkozóan.

DR. SZÉKELY VLADIMÍR

A BME Villamosmérnöki Karán kitüntetéssel szerzett oklevelet 1964-ben. Egyetemi doktori disszertációját 1970-ben védte meg. Kandidátusi fokozatot 1978-ban szerzett, az integrált áramkörök elektro-termikus jelenségei modellezésének témakörében. 1964 óta a BME Elektronikus Eszközök Tanszék oktatója; jelenleg docens, tanszékvezető-helyettes. Fő szakterületei: félvezetőeszközök működésének fizikája, számítógépes szimuláció, integrált áramkörök számítógéppel segített tervezése.

1. Bevezetés

Az integrált áramköri maszkok előállítása — tudjuk — igen költséges és időigényes feladat. Ezért elsőrendű fontosságú kérdés, hogy a maszkok lehetőség szerint már első változatukban jók legyenek — hiszen az esetleg szükséges megismétlés mind költségkihatásai miatt, mind az átfutási idő növekedése folytán kellemetlenül rontja a tervezés hatékonyságát. Különösen áll ez a berendezésorientált áramkörökre, amelyek életképességének alapfeltétele a redukált tervezési, gyártáselőkészítési költség és a vonzóan kis átfutási idő. Ezért minden lehetőséget meg kell ragadni a maszktervek menet közbeni ellenőrzésére, hogy tényleges előállításra csak a már nagy valószínűséggel jó maszkok kerüljenek.

Mit jelent közelebbről a maszktervek (layout-tervek) ellenőrzése? — A tervezés általában számítógéppel segített módon történik (grafikus editorral, digitalizálóval stb.), az eredményként előálló maszktervek tulajdonképpen számítógépi adatfile-ok, amelyek a maszkok geometriáját, a rajtuk levő geometriai objektumok, alakzatok összességét definiálják. Ebből az adathalmazból teljesen automatizált módon áll elő a gyártást szolgáló maszk. A lépések:

- az adatokból a mestermaszk előállítása számítógép-vezérelt ábragenerátorral,
- a mestermaszk végleges méretre kicsinyítése és léptetése,
- a gyártási maszkok másolása.

Idő és költség szempontjából egyaránt szűk keresztmetszet az ábragenerálás. (Jelenleg két ábragenerátor dolgozik az országban; a művelet sebességére az jellemző, hogy hetenként 1—2 *LSI IC* maszkjai állítható elő velük.) Ez mindenképpen azt jelenti, hogy még az ábragenerálás előtt, tehát a maszkokat leíró számítógépi adatfile-okon kell annyi ellenőrzést elvégezni, amennyit csak tudunk. Ez még extrém nagy gépidő felhasználás (akár néhányszor 10 órás CPU idő) mellett is gazdaságos megoldás.

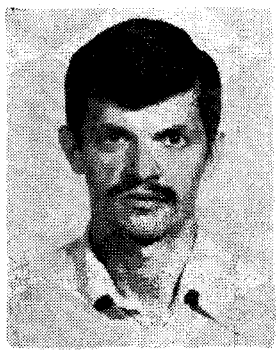
Beérkezett: 1984. I. 26. (⅄)

Ha osztályozni próbáljuk a maszkadatok ellenőrzésének lehetőségeit, azokat két nagy csoportba sorolhatjuk.

- *Szabályellenőrzés jellegű műveletek.* Nevezhetnénk ezeket szintaktikus ellenőrzésnek is. Léteznek ugyanis a maszkalakzatokra olyan alapvető, a technológia által meghatározott szabályok, amelyeket semmiképpen nem szabad áthágni, amelyek szinte a maszkok „geometriai helyesírását", szintaxisát adják. Ilyenek például egy vezetőcsík minimális szélességére, két alakzat minimális távolságára vonatkozó megkötések. E szabályok összességét nevezzük *tervezési szabályoknak*, a rájuk vonatkozó ellenőrzést végzik a tervezési szabály ellenőrző (design-rule check, DRC) programok.
- *Visszafejtés jellegű művelétek.* Most azt ellenőrizzük, hogy a szintaktikusan hibátlan layout azt ábrázolja-e, amit kell — szemantikusan helyes-e tehát. „Visszafejtjük" a maszkrajzolatok alapján, hogy hol milyen alkatrész jön majd létre, és azok hogyan kapcsolódnak egymáshoz — vagyis visszafejtjük a layout által képviselt áramkört.

A visszafejtés mélysége különböző lehet aszerint, hogy mi az áramkör felismert legkisebb alkateleme. Ha ez a legkisebb elem az *alkatrész* (ellenállás, tranzisztor stb.), akkor beszélünk *alkatrészszintű visszafejtésről*. Ha nem alkatrészmélységig, hanem csak egyes részegységekig (tipikusan a cellás tervezésű áramkör celláiig) fejtünk vissza, akkor ez a *cellaszintű visszafejtés*. Utóbbi esetben a cellákon túl csak azok összekötő vezetékeit kell felismerni — ezért sok esetben konnexió-feltárás néven is említik a műveletet.

Cikkünkben a cellaszintű visszafejtés egy lehetséges megvalósításával, az általunk 1983 tavaszán kidolgozott *CELLINEX* (CELL INterconnection EXtraction) programmal foglalkozunk.



BAJI PÁL

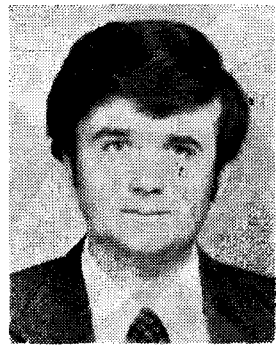
A BME Villamosmérnöki Karán 1970-ben szerzett diplomát. A BME Elektronikus Eszközök Tanszékén 1971—1973 között ösztöndíjasként, Schottky dióda technológiával foglalkozoit. 1973-tól tanársegédként, 1984-től adjunktusként dolgozik a BME Elektronikus Eszközök Tanszéken. Érdeklődési területe: félvezető eszközök számítógépes modellezése, integrált áramkörök számítógépes tervezése.

KERECSENNÉ DR. RENCZ MÁRTA

1973-ban végzett a BME Villamosmérnöki Karának Műszer és Irányítástechnika Szakán. Azóta a BME Elektronikus Eszközök Tanszékén dolgozik, jelenleg adjunktusként. Egyetemi doktori disszertációját 1979-ben védte meg. Szakterülete: félvezető eszközök számítógépes modellezése, integrált áramkörök számítógéppel segített tervezése.

KÓNYA ILONA

1970-ben szerzett oklevelet a Budapesti Műszaki Egyetem Villamosmérnöki Karának Híradástechnikai szakán, majd 1973-ban kitüntetéses villamosmérnök-matematikus szakmérnöki oklevelet. 1970-től a Villamoskari Matematika Tanszéken dolgozik, ahol elsősorban numerikus módszerekkel és számítástechnikával foglalkozik.



DR. MASSZI FERENC

1976-ban szerzett kitüntetéses oklevelet a BME Villamosmérnöki Karán. 1978-ban megvédett műszaki egyetemi doktori disszertációjának témája a vélvezető memóriaelemek számítógépes modellezése volt. Oklevelének megszerzése óta a BME Elektronikus Eszközök Tanszékén dolgozik, jelenleg adjunktusként. Kutatási területe: félvezető struktúrák modellezése, számítógépes tervezés.

2. A **CELLINEX**-program jellemzői

A **CELLINEX** program cellás tervezésű digitális **LSI** áramkörök layout-ellenőrzésére szolgál [5]. Kidolgozása része annak a fejlesztő munkának, amely a Mikroelektronikai Vállalatnál a berendezésorientált integrált áramkörök tervezése software eszköztárának létrehozását célozza. Szorosan kapcsolódik a tanszékünk által korábban kidolgozott **CELLIB** cellakönyvtár-kezelő programhoz, amiről a közelmúltban adtunk ismertetést ugyanezen folyóiratban [4].

A program könyvtári cellákból és összeköttetéseikből álló layout visszafejtésére alkalmas. A layout adatait a Mikroelektronikai Vállalatnál általánosan használt grafikus leíró nyelven olvassa be. A kezelhető layout-méret megfelel a berendezésorientált tervezés által néhány év távlatában igényeltnek: 400 cella, 2000 jelvezeték, 5000 összeköttetésfunkciójú alakzat, mint maximum. Az összeköttetésrétegek száma a kezelhető 20 maszksíkon belül tetszőleges: egyrétegű és többrétegű fémezéssel készülő áramkörök egyaránt visszafejthetők. A program igen hatékony particionálási technikával és gondosan tervezett algoritmusokkal biztosítja a layout-feldolgozási feladatoknál szokatlanul kis számítási időt (4—5 perc a legnagyobb feldolgozható hálózatra).

A program eredményközlése az ellenőrzés változatos formáit szolgálhatja; úgy alakítottuk ki azokat, hogy lehetőleg minden tervezői igényt kielégítsenek. A visszafejtett áramkör például automatikusan továbbítható logikai szimulációra, vagy grafikus képernyőn tanulmányozhatjuk a layout-rajzon a logikai jelek terjedési útjait — hogy csak a lehetőségek szélsőségeit említsük.

DR. KOLTAI MIHÁLY

1973-ban szerzett diplomát a BME Villamosmérnöki Karán. 1976-ban megvédett műszaki egyetemi doktori értekezésének témája az általa kifejlesztett elektromágneses térszámító programrendszer volt. Oklevelének megszerzése óta a BME Elméleti Villamosságtan Tanszékén dolgozik, jelenleg adjunktusként. Kutatási területe: elektromágneses térszámítás, félvezető eszközök és technológia számítógépes modellezése, számítógépes grafika.

3. Algoritmus-kérdések

Első ránézésre úgy tűnhet, hogy az összeköttetésfelderítés nem vet fel komolyabb algoritmus-problémákat. Poligonok érintkezését-átfedését, poligonoknak ablakokkal való találkozását kell feltárni és nyilvántartani. Első benyomásunk az lehet, hogy eléggé triviális analitikus geometriai problémák programozásában merül ki a feladat megoldása.

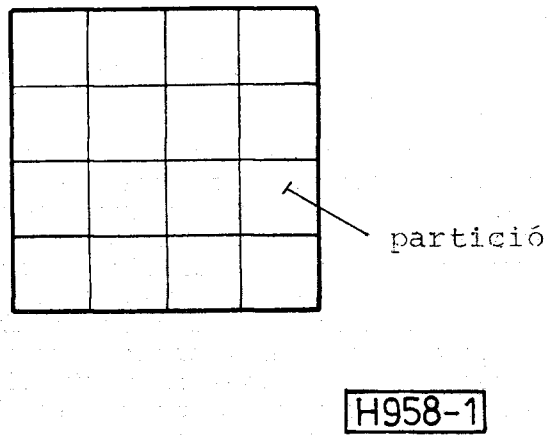
Aki már találkozott az **LSI** integrált áramkörök maszkinformációinak gépi feldolgozási problémáival, az tudja, hogy ez nincs egészen így. A maszkok nagy összetettsége miatt a feldolgozandó információmennyiség már olyan hatalmas, hogy a „straightforward" programozási mód már nem hozhat megoldást. Egyrészt a tárolóhelyigény nő meg alaposan. Nem járható az az út, hogy a számítás során

minden adatot mindig a központi memóriában tartsunk. Ehelyett a háttértárolón őrzött nagy adatfileokon kell a programnak dolgoznia (egyiket a másikba transzformálni stb.). A **CELLINEX** program által kezelt maximális layout-bonyolultságnál ezek a fileok egyenként 0,5—1 megabyte-ig is elmehetnek. Másrészt (és ez a tárolóhelygondoknál is súlyosabb) az úgynevezett kétparaméteres geometriai műveletek időigénye nő meg tűrhetetlenül. Ilyen művelet például az alakzatok érintkezéseinek, átfedéseinek megkeresése. Ezekhez minden alakzatot minden másikkal össze kell vetni. Ha a vizsgált maszksíkon n alakzat van,

$$\binom{n}{2}=\frac{n(n-1)}{2}\approx\frac{n^2}{2} \qquad \text{(i)}$$

összevetést kell tennünk — a műveletszám tehát n^2-nel arányos. 2000 alakzattal számolva és egy összevetést 200 μs-mal véve számításba, egyetlen ilyen művelet 6-7 perc **CPU** időt igényelne, ami a teljes program futását órás nagyságrendbe emelné.

A vázolt nehézségek indokolttá teszik, hogy az algoritmus kérdésekről részletesen szóljunk. Ezt tesszük az alábbiakban, mondanivalónkat néhány lényeges téma köré csoportosítva.



1. ábra. A layout particionálása

Particionálási technika. Az alakzatok nagy számából eredő problémákat nagyságrendekkel enyhítheti a layout-terv részekre szabdalása, particionálása, és a műveletek partíciónkénti külön elvégzése [1]. Tegyük fel például, hogy a layout-ot az 1. ábra szerint mindkét irányban felosztva, k db partícióra vágjuk. Legyen az alakzatok eloszlása a maszkon egyenletes és egyelőre tekintsünk el a partícióhatárok által elvágott alakzatok problémájától. Ekkor egy partícióba n/k alakzat esik és a végzendő műveletek számá összesen

$$k\left(\frac{n}{k}\right)^2\cdot\frac{1}{2}=\frac{n^2}{2k}, \qquad (2)$$

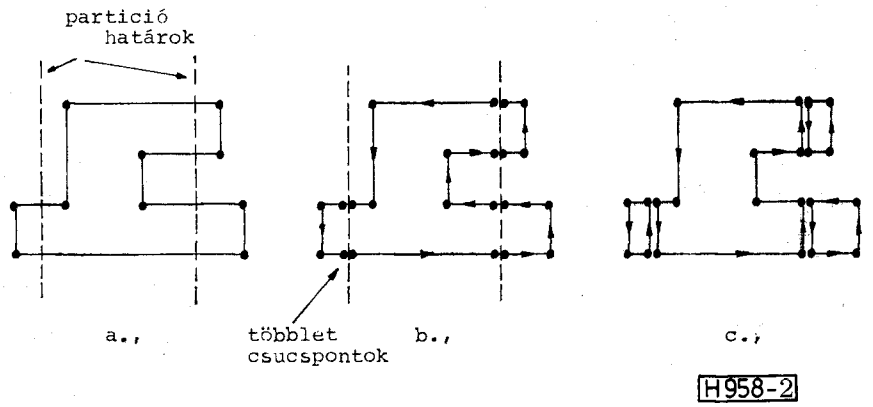
vagyis k-szor kisebb, mint a particionálás nélküli esetben. A nyereség nyilvánvaló; 100 partícióval dolgozva például, az előbbi, 6-7 perces műveletidő néhány másodpercre csökken.

Elhamarkodott volna természetesen az a következtetés, hogy a partíciók számát határtalanul növelni volna célszerű. Ha egy-egy partíció mérete az átlagos alakzatméret alá csökken, egyre több alakzatot vágnak két vagy több részre a partícióhatárok, és ezzel az alakzatok teljes száma növekedni kezd. A sok partíció maga is jelentős többletadminisztrációt, tehát időt igényel. Ezért egy határon túl már nem érdemes folytatni a layout felosztását; van egy optimális mértéke a particionálásnak [2]. (Erre vonatkozó kísérleti tapasztalatokról a következő szakaszban számolunk be.)

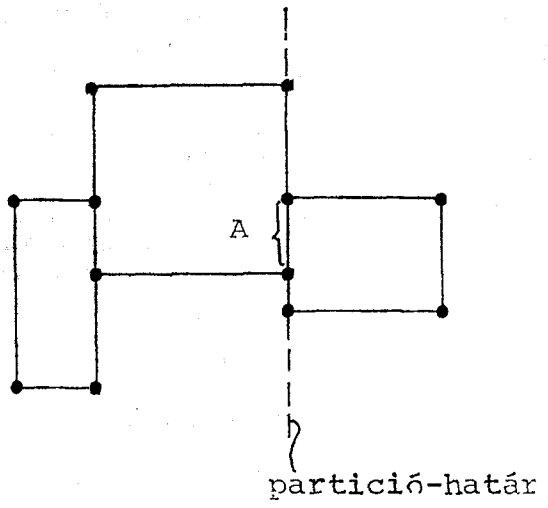
A particionálás gyakorlati kérdései. A **CELLINEX**-programban egy- vagy kétdimenziós, rögzített határú, elvágó particionálási technikát alkalmazunk. A partíciók számát a felhasználó adhatja meg. Ez mind x, mind y irányban maximálisan 16, tehát összesen maximálisan 256 partíció lehet. Rögzített határon azt értjük, hogy a partícióhatárok helye állandó és független a maszk alakzatainak elhelyezkedésétől. Az „elvágó" particionálás annyit jelent, hogy a partícióhatárok részekre választják szét azokat az alakzatokat, amelyeket átmetszenek.

A particionálás algoritmusát a 2. ábra segítségével mutatjuk be. A műveletet először a függőleges partícióhatárokra végezzük el. A lépések a következők:

— mindazon pontokban, ahol poligonél partíció-



2. ábra. A particionálási algoritmus magyarázatához. A többlet csúcspontok párosával azonos helyre esnek, csak a szemléltetés kedvéért ábrázoltuk páronként külön azokat



H958-3

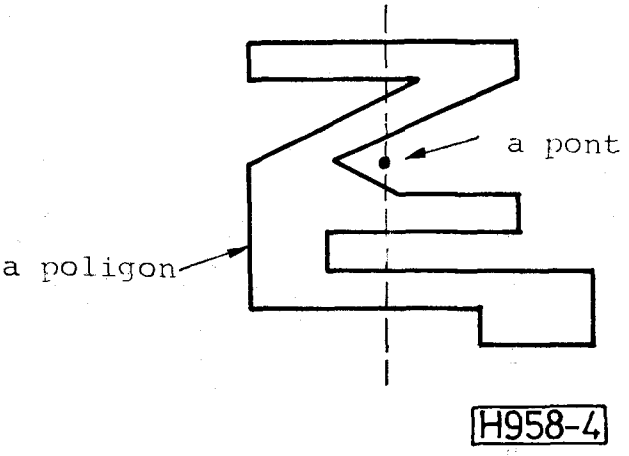
3. ábra. Ha partíción belül vizsgáljuk csak az alakzatok érintkezését, az *A*-val jelölt kontaktus feltáratlan marad

határt metsz, két-két többlet poligon csúcspontot veszünk fel (ugyanazon koordináta-adatokkal, de egyiket az egyik, másikat a másik határos partícióhoz tartozóan; 2*b* ábra),
- valamely körüljárási irány szerint pointerláncra fűzzük a poligon csúcspontjait,
- megállapítjuk minden egyes partícióhatárra a felvett többlet csúcspontok *y* irányú sorrendjét,
- az e sorrendben szomszédos többlet csúcspontok közül a körüljárás szerint „hátsók" továbbmutató pointereit felcseréljük (2c ábra).

A vízszintes partícióhatárokra a művelet hasonló módon végezhető el.

A particionálásnál külön gondot okoznak azok az alakzatok, amelyek csúcspontjai, élei éppen partícióhatárra esnek. Ezek kezelése, ha megoldhatatlan problémát nem is jelent, de a particionáló algoritmust meglehetősen bonyolítja. Ennél is súlyosabb nehézség viszont (ami miatt végül is más megoldást kellett választanunk), hogy ha megengedünk partícióhatárra eső alakzatéleket, akkor az érintkező alakzatok feltárása nem lesz partíciónként függetlenül kezelhető probléma (lásd a 3. ábra ellenpéldáját).

A vázolt gondok elkerülésére a következő megoldást választottuk. Az alakzat-koordináták minden esetben eleve egész értékűek (a maszk-leíró nyelvek ugyanis egy kötött rácson, „grid-értékekben" teszik lehetővé a koordináták megadását). A **CELLINEX**-program belső adatábrázolásában kétszer ilyen sűrű rácsot használ, a maszk minden koordinátáját tehát kettővel szorozzuk. Így az alakzatok csúcspont-koordinátái feltétlenül páros számok. Ez után elegendő a partícióhatárokat páratlan koordináta-értékűre venni, s elkerüljük a partícióhatárra eső csúcspontokkal, élekkel együttjáró gondokat.



H958-4

4. ábra. A „pont a poligonban van-e" algoritmus magyarázatához

Rendezési műveletek. Célszerű elvégezni — és a programban meg is tesszük — a particionált alakzatok megfelelő rendezését. A rendezés partíciók szerint történik. Végeredménye a „rendezett, particionált alakzat-file", amelynek minden egyes rekordja egy-egy partíció alakzatait, kontaktusablakait tartalmazza. A partíciónkénti feldolgozás során e file rekordjait olvassuk egy nagy méretű (80 kbyte) bufferbe. Az egy partíciónyi adathalmazon a további műveletek előtt az egyes maszksíkokat összejelölő pointérláncokat hozunk létre.

A „pont a poligonban van-e" algoritmus. Ez az egyik legfontosabb algoritmusa a programnak. Feladata: megállapítani, hogy egy pont a poligonon belülre (határára) esik-e vagy azon kívül. Az algoritmus jelentősége kettős. Egyfelől az ablakok közvetítésével létrejövő kontaktusok feltárásának kulcsa. Az ablakokat a program középpontjukkal veszi számításba: az a két poligon érintkezhet egy ablakon át, melyek mindegyikébe beleesik az ablak középpontja. Másfelől a poligonok azonosítóval (címkével) való ellátásában van szerepe: egy címkeszöveg azon poligonhoz tartozik, amelyre a kezdőpontja esik.

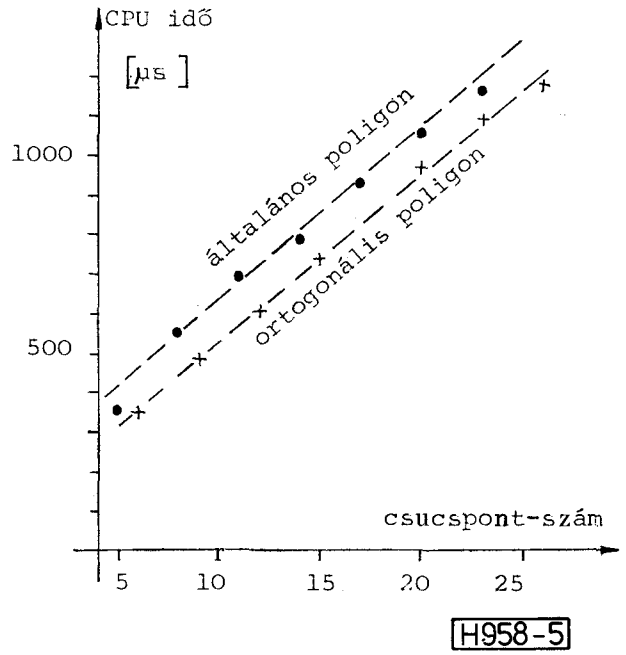
Az algoritmus lényegét a 4. ábrán szemléltetjük. Lépései:
- a kérdéses ponton át képzeletben egy (pl. *y* irányú) egyenest húzunk,
- meghatározzuk ezen egyenes metszéspontjait az alakzat kontúrjával,
- ha (*y* irányban értve) páratlan számú ilyen metszéspont van a kérdéses pont „alatt", akkor a pont benne van az alakzatban, egyébként rajta kívül.

(Az alakzathatárra eső pont esetét kezelő külön lépéseket itt nem részletezzük.)

Az 5. ábrán futásidő statisztikát mutatunk be a fenti algoritmust realizáló szubrutinról.* A statisztikából elhagytuk a triviálisan számolható eseteket; ezeknél a gépidő ≈ 50 μs.

A műveletek egyszerűsítési lehetőségei. A gyorsabb működés érdekében a program mindenütt kihasználja

* Ez és minden további futásidő adat a MEV Fóti úti telephelyén, a gépi tervezési osztályon végzett futtatásokból származik és **CPU** időben értendő.



5. ábra. A „pont a poligonban van-e" algoritmus futásidői

azokat a lehetőségeket, amelyek speciális esetekben a számítás egyszerűsítésére adnak módot vagy triviális esetekben a számítás teljes mellőzését teszik lehetővé. Ennek érdekében a poligonok belső tárolásának része

- a poligont befoglaló téglalap négy koordináta adata,
- kód, amely azt jelzi, hogy a poligon általános-e vagy valamely speciális, könnyebben kezelhető kategóriához tartozik (téglalap, ortogonális poligon).

Megéri, hogy tároljunk-e néhány redundáns adatot, mert segítségükkel a számítás sok esetben egyszerűsíthető. Például a „pont a poligonban van-e" vagy „két poligon érintkezik-e" típusú döntéseknél először a befoglaló téglalap adatait veszi tekintetbe a program. A negatív döntés (a pont *nincs* a poligonban, a két poligon *nincs* érintkezésben) sok esetben már ennek alapján meghozható. A poligon kategóriája alapján háromfelé ágazik a további számítás. Téglalap esetén nincs is szükség további vizsgálatra (hiszen ilyenkor az alakzat azonos a már megvizsgált befoglaló téglalappal). Ortogonális poligon esetén egyszerűbb úton folytatódhat a számítás, és csak az általános poligon esetén „veti be" a program a ferde oldaléleket is kezelő, s így leghosszadalmasabb algoritmust.

Az összeköttetés-felderítés folyamata. A program először partíciónként végigtekinti, hogy mely poligonok között áll fenn vezető összeköttetés. Ez az ablak közvetítésével létrejövő összeköttetéseknél a „pont a poligonban van-e" algoritmus segítségével történik, azonos maszk-sík érintkező alakzatainál a „két poligon érintkezik-e" algoritmussal. Valamennyi felfedett érintkezést az ún. érintkezés-file-on jegyzi be a program; e file-on minden ilyen érintkezést két-két adat, a két érintkező poligon sorszáma reprezentálja.

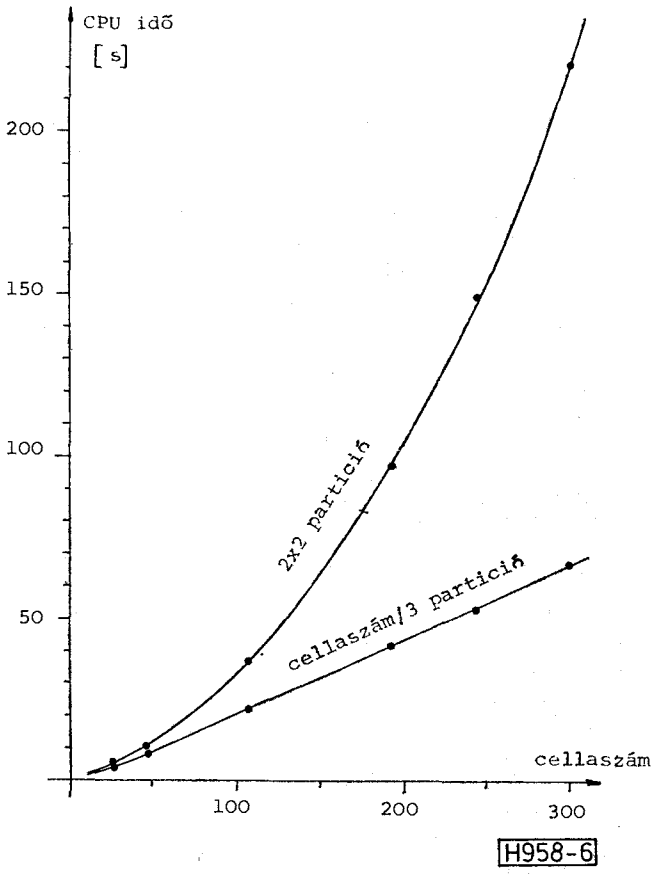
A következő lépés: a páronként felfedett érintkezések alapján megkeresni az alakzatok ekvipotenciális csoportjait. Ennek algoritmusa:

- minden alakzathoz egy pointert rendelünk, ami az ekvipotenciális csoport egy következő alakzatára mutat,
- e pointerek kezdeti beállítása olyan, hogy mindegyik alakzat önmagára mutat (tehát minden csoport egyetlen alakzatból áll),
- sorra vesszük az érintkezés-file-on bejegyzett alakzatpárosokat és minden egyes ilyen páros pointereit felcseréljük. (A többszörös összeköttetéseket, hurkokat tartalmazó ekvipotenciális csoportok miatt itt még néhány többletlépésre is szükség van; ezeket nem részletezzük.)

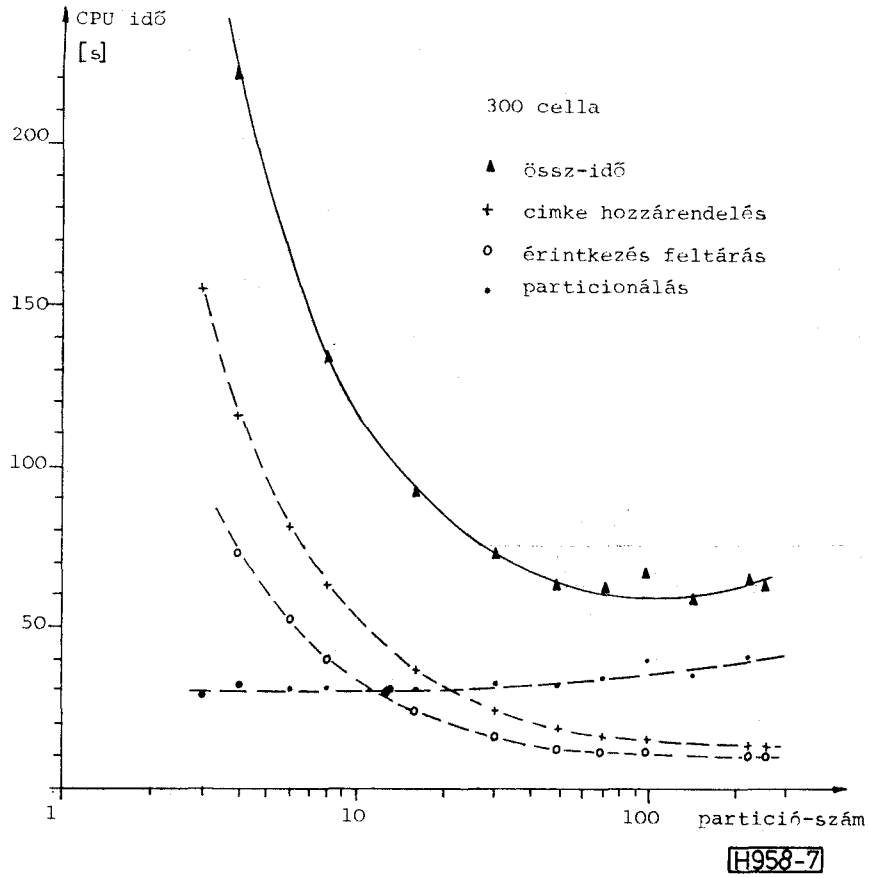
Mire a fenti eljárást végigvittük, a pointerek az ekvipotenciális alakzatok csoportjait láncolják össze. Az összeköttetés-feltárás tehát megtörtént; ami hátra van, az az eredményközlés.

4. Particionálás és futásidő

A program próbaüzeme során kísérleti vizsgálatokat végeztünk arra vonatkozóan, hogy milyen összefüggés van hálózatméret és futásidő között, valamint hogy a particionálás mértéke hogyan befolyásolja a program egyes rész-algoritmusainak időigényét.



6. ábra. A számítás **CPU** ideje a cellaszám függvényében (a „merge" művelet nélkül)



7. ábra. Az egyes rész-algoritmusok futásideje a partíciószám függvényében

A vizsgált mintahálózat alapeleme egy három cellából álló áramkörrészlet volt, 24 összeköttetés-alakzattal, 7 kontaktusablakkal, 20 alakzatazonosítóval. Ezt sokszoroztuk meg mátrixszerűen, a különböző méretű layoutok vizsgálatához.

Először a teljes **CPU** időt mértük különböző nagyságú hálózatokra, a partíciószámot vagy állandó értéken tartva vagy a hálózatmérettel arányosan növelve. Az eredményeket a 6. ábrán mutatjuk be. Rögzített számú partíciónál a várakozásnak megfelelően, nagyjából négyzetesen emelkedik a futásidő a layout mérettel, növekvő számú partíciónál cca lineárisan.

A következő vizsgálatunk egy nagyméretű, 300 cellából álló hálózatra vonatkozik. A partíciószámot egyre növelve vizsgáltuk az egyes rész-algoritmusok futásidejét. Az eredményeket a 7. ábrán közöljük. Látható, hogy a címke-hozzárendelés és az érintkezések feltárása a partíciószám növelésével monoton csökkenő időt igényel. Ezt részben lerontja magának a particionálásnak a növekvő időfelhasználása. Az összidőnek láthatóan minimuma, optimuma van a 100 körüli partíciószámnál.

Megjegyzendő, hogy az sem lényegtelen, hogy egy adott partíciószám hogyan oszlik meg az x és az y irányú particionálás között. A legjobb eredményt akkor értük el, ha a két irányban cca egyforma számú partíciót alkalmaztunk (egybehangzóan a kérdésre vonatkozó elméleti vizsgálatokkal [3]).

5. Az eredmények dokumentálása

A visszafejtés eredményei igen változatos formában használhatók fel a layout ellenőrzésére. A **CELLINEX**-program eredményközlési lehetőségeivel igyekszik támogatni minden szóba jövő ellenőrzési módszert.

Hibajelzések, figyelmeztető üzenetek. A visszafejtés során feltárásra kerülhetnek a layout olyan vonásai, amelyek feltétlenül tervezési hibára utalnak. Ezekről a program hibajelzést nyomtat. Más esetben csak valószínűsíthető a hiba jelenléte — ezekből képződnek a figyelmeztető üzenetek. Tipikus, a program által jelzett rendellenességek az alábbiak:

- kihasználatlan ablak,
- bekötetlen cellakivezetés,
- hurkot tartalmazó ekvipotenciális vezeték,
- zárlatok (tápfeszültség, cellakimenet stb.),
- funkcionális rendellenesség a cellák összekötésében (nem csatlakozik cellabemenet egy jelvezetékhez, több, nem háromállapotú kimenet össze van kötve stb.),
- rendeltetés nélküli részletek a layout-on (vezeték, amely csak egy cellához csatlakozik vagy egyhez sem).

Listázás a cellák szerint. A nyomtatási képet a 8. ábra mutatja, egyetlen cellára vonatkozóan. A program kilistázza a cella kivezetéseit, címkéjükkel és a funkciójukra utaló karakterkóddal (**INP**, **OUT** stb.)

```
                      CELL LISTING
                      ============

             DESIGN FILE NAME=  CELL1

CELL NO.=   1,   NAME=WAND2   ,   RELATED INTERCONNECTIONS=

              CELL NODES            I    INTERCONNECTION
          NO.   TYPE   IDENTIFIER   I    NO.  IDENTIFIER
          ------------------------------------------------
          1.    INP*      BEI       I     1.   LAAB
                                    I          MASODIK
          2.    INP*      BE2       I     2.   ELSO.BEK
          3.    OUT*      OUTP      I     3.   -
          4.    UGG-      TAPF      I    15.   -
          5.    UDD-      TAPF      I    16.   -
          6.    GND-      FOLD      I    17.   -
          ------------------------------------------------
                                                  H958-8
```

8. ábra. A cellák szerinti eredményközlés részlete

```
                    INTERCONNECTION LISTING
                    =======================

          DESIGN FILE NAME=  CELL1

INTERCONNECTION NO.*   1   IDENTIFIER=LAAB     ,
                                ALIAS=MASODIK ,  CONNECTED CELLS=

                  C E L L      I          N O D E
              NO.     NAPE     I   NO.     TYPE    IDENTIFIER
              --------------------------------------------
              1.    WAND2      I   1.      INP*      BE1
              2.    WNOR2      I   2.      INP*      BE2
              --------------------------------------------

      ****  ERROR  ****    NO OUTPUTS OR FIXED LEVELS (SUFPLY,GND)
                           ARE CONNECTED TO THIS TREE
      **** WARNING ****    MULTIPLE IDENTIFIER
                                                      H958-9
```

9. ábra. Az összeköttetések (jelvezetékek) szerinti eredményközlés részlete

ellátva kiírja, hogy az illető cellakivezetés hányas sorszámú és milyen azonosítójú összeköttetéshez csatlakozik.

Listázás az összeköttetések szerint. A 9. ábrán látjuk ennek egy összeköttetésre vonatkozó részét. Tulajdonképpen az előbbi lista megfordításáról van szó: most azt listázzuk, hogy az egyes összeköttetésekhez mely cellák, mely kivezetésükkel csatlakoznak.

Jelfolyam-felderítés. A felhasználónak lehetősége van arra, hogy egy logikai jel útját nyomon kövesse az egymáshoz kapcsolódó cellákon át. Ehhez meg kell adnia

— a kiinduló cellát,
— a felderítés mélységét (hogy tehát az hány cellán át történjen meg).

A dokumentáló szegmens megkeresi a kiinduló cella kimeneteit, az azokhoz bemenetükkel csatlakozó további cellákat, ez utóbbiak kimeneteit ... stb., a kívánt mélységig. A jelfolyam felderítés eredményét a program a 10. ábrán látható, szemléletes formában közli. A műveletet nem csak a jelterjedés irányában, hanem azzal ellentétesen, visszafelé is kérhetjük.

Grafikus-interaktív jelfolyam-felderítés. Az előbbi művelet grafikus display képernyőn is végezhető. Az ernyőn megjelenik a layout vagy annak kiválasztott részlete, de kizárólag a cellák körvonalainak ábrázolásával. Ha kijelölünk egy cellát és kérjük a jelfolyam-felderítést, kirajzolódnak mindazon összeköttetés-alakzatok, amelyek az illető cella kimeneteihez kapcsolódnak és vonalkázás emeli ki a bemenetekkel csatlakozó további cellákat (11. ábra). A művelet ismétlésével tetszőleges jel-út nyomon követhető. Mint az előbb, itt is haladhatunk a jelterjedéssel ellentétes irányban is.

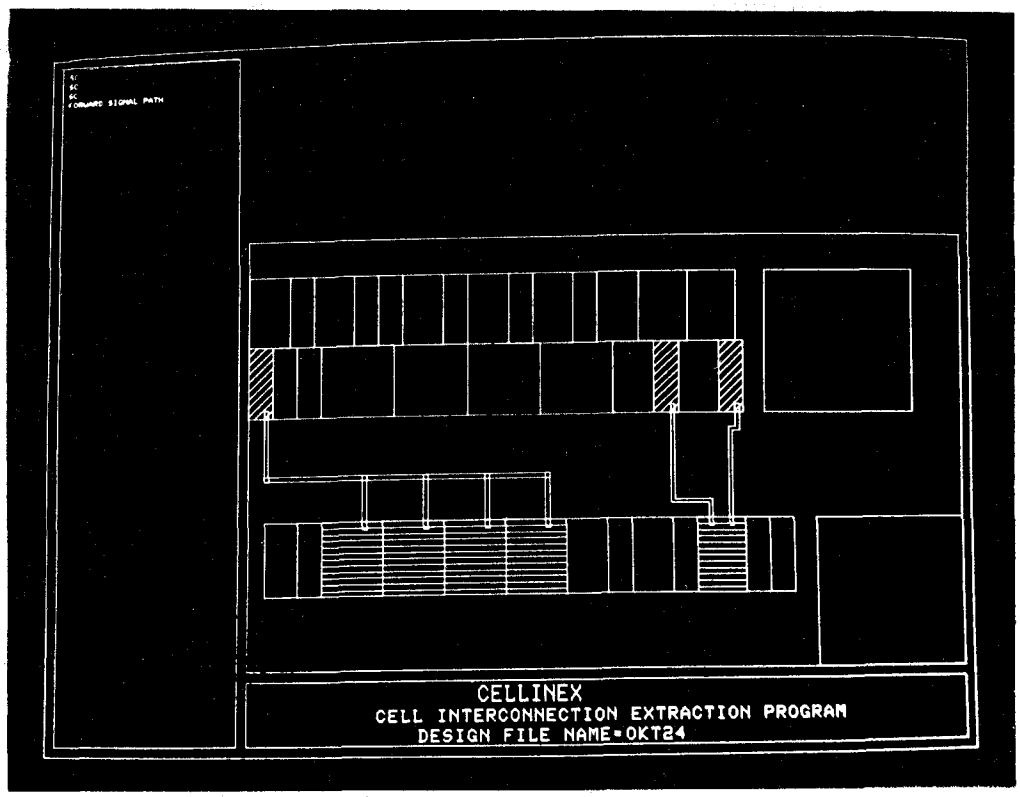
Logikai hálózatleírás generálása. A program jelenleg a **LOBSTER** logikai szimulációs program [6] bemeneti nyelvén állítja elő a layout által képviselt hálózat logikai leírását, a cellák belső logikai leírását a **CELLIB**-program [4] által szervezett cellakönyvtárból emelve ki. (A későbbiekben valószínűleg a MEV most kialakítás alatt álló integrált tervezőrendszerének **ETALON** egységes topológiai leírónyelvére térünk itt át.) A logikai leírás

— összevethető a felhasználó által megadottal,
— ellenőrző logikai szimulációnak vethető alá.

Layout-statisztikák készítése. Nem közvetlenül a visszafejtéshez tartozó, de hasznos szolgáltatása a programnak, hogy a layout lényeges statisztikai jellemzőit közli. Ilyenek:

```
1-JUL-83          ,CELL INTERCONNECTION EXTRACTION PROGRAM          PAGE=   5
================================================================================

                     SIGNAL-FLOW PATH DISCOVERING
                     ============================

            DESIGN FILE= CELL

            DIRECTION= FORWARD                   DEPTH= 5 STAGE

PRINTCUT CODE= --(IIII)222/AAAAAA         WHERE  IIII= CELL INPUT NODE LABEL
                 (OCOO)---                       OCOC= CELL OUTPUT NCOE LABEL
                 (                               AAAAAA= CFLL NAME
                 (OCCO)---                         222= CELL NO.

INITIAL CELL       1ST DEPTH           2ND DEPTH          3RD DEPTH          4TH DEPTH          5TH DEPTH
---------------------------------------------------------------------------------------------------------

 1/WAND2
  COUTP)-----(BE2 )  3/WNOR2
                   (OUTF)-----(BE1 ) 20/WNOF2
                          I          (OUTP)-----(BE1 ) 21/WNOR2
                          I                          (OUTP)-----(BEI ) 33/WNOR2
                          I                                 I          COUTP)-----(BE1 ) 39/WNOR2
                          I                                 I
                          I                                 ---(BE2 ) 37/WAND2
                          I                                            COUTP)-----(BE2 ) 39/WNOR2
                          I
                          ---(6E2 ) 19/WAND2
                                     (OUTP)-----(BE2 ) 21/WNOR2
                                                     (OLTP)-----(BE1 ) 38/WNOR2
                                                            I          CCUTP)-----CBE1 ) 39/WNCR2
                                                            I
                                                            ---(BE2 ) 37/WAND2
                                                                       (CUTP)-----CBE2 ) 39/WNOR2
                                                                                        H958-10
```

10. ábra. A jelfolyam-felderítés eredményközlése sornyomtatón



11. ábra. Jelfolyam-felderítés a grafikus képernyőn

— alakzatok, ablakok, azonosítók, jelvezetékek száma,
— cellák és cellafajták száma,
— alakzatok átlagos és maximális csúcspontszáma,
— x és y irányú átlagos és maximális mérete,
— alakzatok megoszlása a partíciók között stb.

6. Felhasználás, továbbfejlesztés

A **CELLINEX**-program 1983 júliusában került használatba a Mikroelektronikai Vállalatnál. Rendeltetése a cellás tervezésű berendezésorientált áramkörök layoutjának ellenőrzése. A használati tapasztalatokról korai ma még szólni; ezekre 1-2 év után érdemes majd visszatérni. A továbbfejlesztés lehetőségei közül kettőt látunk már ma meglehetősen világosan. Ezek:

— A program „feldeklarálása" nagyobb hálózatméretre. A programkészítés és az eddig végzett futtatások tapasztalatai alapján nyilvánvaló, hogy a program nagyobb nehézségek nélkül alkalmassá tehető a 400-nál több cellát tartalmazó layout-ok vizsgálatára — előre láthatóan kb. 1000 celláig.
— A parazita kapacitások számítása. A program csekély kiegészítéssel alkalmas lesz arra, hogy az összeköttetések által képviselt szórt kapacitást számolni tudja. Ez igen lényeges lehet a pontosabb logikai szimuláció, az időzítések pontos ellenőrzése szempontjából.

Várható természetesen, hogy a használati tapasztalatok még egyéb továbbfejlesztési igényeket is felvetnek majd.

IRODALOM

[1] *Dr. Székely V., Baji P., dr. Rencz M., dr. Koltai M.:* IC maszkok gépi tervezése, tanulmány, BME Elektronikus Eszközök Tanszéke, 1981.
[2] *Baird:* Fast algorithms for LSI artwork analysis, IEEE DATC 1977, pp. 303—311.
[3] *Farkas G.:* Integrált áramkörök helyettesítő képének meghatározása a topológiai elrendezés alapján, szakmérnöki diplomaterv, BME 1980.
[4] *Dr. Székely V., Baji P., Kerecsenné dr. Rencz M., Kónya I., dr. Masszi F.: CELLIB* — cellakönyvtár kezelő program a mikroelektronikai tervezés céljára, Híradástechnika, 1983. nov.
[5] CELLINEX cellaszintű layout-visszafejtő program, használati utasítás, Budapest, 1983.
[6] *Jávor A., Benkő T-né:* LOBSTER programrendszer felhasználói leírás, KFKI kiadvány.