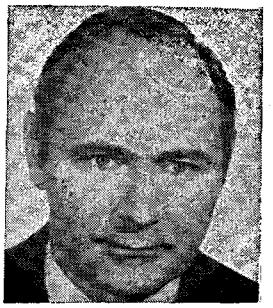
μSIM—logikai szimulátor mikroszámítógépre

DR. GÄRTNER PÉTER
BME Elektronikus Eszközök Tanszék



DR. GÄRTNER PÉTER

A BME Villamosmérnöki Kar Híradástechnika Szakon 1960-ban végzett, majd kitüntetéses diplomát szerzett 1961-ben. 1963-ig az Elektromechanikai Válldlatnál dolgozott, mint fejlesztőmérnök, tv-adók antennarendszereinek kutatás-fejlesztési és megvalósítási munkáján. 1963 óta a BME Elektronikus Eszközök Tanszékén dolgozik. 1968-ban antennák és tápvonalak témakörben egyetemi doktori címet szerzett. Jelenleg elektronikus eszközök méréstechnikájával foglalkozik, különösen a nagybonyolultságú integrált áramkörök tesztelési kérdéseivel.

ÖSSZEFOGLALÁS

Az előadás bemutatja a Z80 alapú mikroszámítógépre készült μSIM logikai szimulátorprogramot. A program néhányszáz kapu-funkcióig terjedő hálózatok szimulációját végzi kapu-szinten, egységnyi kapukésleltetéses módszerrel. Az input szekvenciák leírására hatékony magas szintű nyelvvel rendelkezik és egyszerű hibaszimulációra is képes. Használata megkönnyíti a berendezés-orientált áramkörök logikai és teszt tervezését.

1. Célkitűzés

A 70-es években a logikai szimuláció területén a VLSI áramkörök megjelenése következtében a funkcionális szintű modellezés és szimulációs technika került az előtérbe. A közelmúltban azonban a figyelem ismét a kapu-szintű modellezés felé fordult. Milliós kapuszámú hálózatok kapu-szintű szimulációjára alkalmas célszámítógépeket (ill. többprocesszoros rendszereket) fejlesztettek ki [1, 2]. Ennek az a magyarázata, hogy a logikai szintű szimulációnál az áramkörök működésének finomabb részleteit kapu-szintű modellezéssel lehet a legjobban visszaadni.

A jelen cikkben ismertetett szimulátor program is kapu-szintű, de mikroszámítógépre készült. Kidolgozásánál figyelembe vettük a mikroszámítógépek elterjedtségét, valamint azt, hogy az áramkörtervezői gyakorlat a berendezésorientált áramkörök tervezése felé tolódik el. Célunk olyan szimulátor kidolgozása volt, amely ezt a logikai és teszt-tervező tevékenységet a mikroszámítógép lehetőségei mellett maximálisan támogatja.

2, Bevezetés

A μSIM programot a BME Elektronikus Eszközök Tanszékén dolgoztuk ki egy saját fejlesztésű Z80 alapú mikroszámítógépre, amelyben katódsugárcsöves display, floppy-disk és mátrixnyomtató áll rendelkezésre. A program terjedelme jelenleg 11 KByte és 41 KByte-ot használ adatok tárolására.

A program megoldóalgoritmusa egységnyi kapukésleltetésű szelektív nyomkövetés (unit-delay table-driven selective tracing) [3]. A hálózat-leírás szintaxisát a SEMCON rendszerben futó SIMUL program mintájára dolgoztuk ki. A hálózatot meghajtó jelek leírására azonban egy teljesen új, önálló bemeneti nyelvet alakítottunk ki. A következőkben röviden ismertetjük a program főbb tulajdonságait.

Beérkezett: 1985. X. 25. (H)

3. A csomópontok kezelése

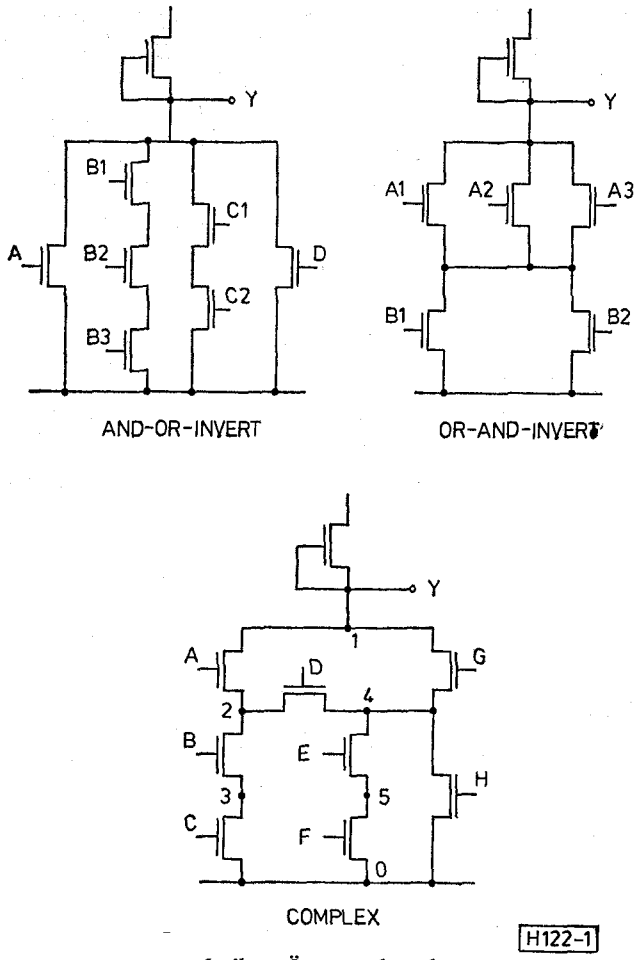
A primer bemenetek, valamint az egyes kapuk kimeneteivel definiált csomópontok maximum hatkarakteres alfanumerikus névvel vannak ellátva. Megvan a lehetőség a csomópontok egy csoportját (sorozatát) vektorként deklarálni, és akkor a továbbiakban az egész csoportra, valamint annak egyes összefüggő rész-sorozataira közösen is lehet hivatkozni. Formailag a rendszer az integrált áramkörök lábainak a jelölési módját követi. A belső kezelés természetesen sorszámok alapján történik, amelyeket a program egy-egy byte-ban tárol. Mivel a kétbyte-os szavakban való tárolás jelentősen komplikálta volna a programot, ezért a szimbólumok számát 256-ban korlátoztuk.

4. Kapuszintű szimuláció

A program a logikai hálózatokat alapvetően kapuszinten szimulálja. A rendelkezésre álló kapukészlet azonban a szokásos egyszerű kapuk mellett (ezek: INVERTER, AND, NAND, OR, NOR, XOR, XNOR) az NMOS IC technológiához illeszkedve összetett formákat is tartalmaz: AND—OR—INVERT, OR—AND—INVERT, COMPLEX. (1. ábra).

Az összetett kapuk elvileg korlátlan, gyakorlatilag egynél jóval több, 3...10 elemi kapufunkciót tudnak megvalósítani, és így a szimulálható maximális hálózat nagysága a 250 elemi kapu-funkciót jelentősen meghaladhatja. A programban, az egész rendszer minél homogénebb struktúráját megőrzendő, funkcionális egységeket, mint pl. flip-flop, dekóder stb. nem modelleztünk.

A hierarchikus hálózatfelépítést kapuszinten modellezett funkcionális részegységeknek makrók formájában való deklarálása támogatja. A makrók felépítésére nincs megkötés, tetszőleges számban behívhatók és négyszeres mélységben egymásba skatulyázhatók. Lehetőség van makrokönyvtár kialakítására, amelyből mindenkor az éppen szükségeseket hívjuk le. Szükség esetén minden egyes belső csomópont egyértelműen azonosítható és hozzáférhető.



1. ábra. Összetett kapuk

5. Időkezelés és megoldóalgoritmus

A hálózat logikai megoldása szimuláció közben sokszorosan ismétlódó, iteratív folyamat. Itt kell a futásidő kérdésekre a leginkább koncentrálni. Az egységnyi kapukésleltetés az időkezelés legegyszerűbb formája, az időnek aránylag durva kvantálását jelenti. A kapuszintű modellezéssel együtt azonban így is meglehetősen jó eredményeket szolgáltat a szimulált hálózatról.

Az alkalmazott logikai szintek számát is a minimumon tartottuk. A szokásos Ø és 1 mellett csak az indítási tranziensek alkalmas lebonyolításához elengedhetetlenül szükséges u (undefined) állapotot használjuk.

A szelektív nyomkövetés szintén a legegyszerűbb és egyben a leggyorsabb hálózatmegoldó módszer, hiszen mindig csak azokat a kapukat értékeli ki, amelyek bemenetén változás történt. Az elsődleges bemeneteken bekövetkezett változások hatását minden egyes lépésben iterative végigkövetjük, és ha a hálózat stabilizálódott, kiadjuk a végeredményt. Ez a módszer kombinációs és szinkron szekvenciális hálózatoknál elvileg is jó eredményt ad, amennyiben feltételezzük, hogy két lépés között a hálózatnak van ideje stabilizálódni. A stabilizálódáshoz szükséges időt azonban csak kapukésleltetési időegységekben kaphatjuk meg, ami ugyan elég durva, de ezért mértékadó becslésnek tekinthető.

Aszinkron hálózatoknál a lényeges hazárdok fellépésének a lehetősége és veszélye miatt a kapott eredményeket kritikának kell alávetni. A program sem a visszacsatolási hurkokat, sem a hazárdokat nem ellenőrzi, ezek a feladatok a felhasználóra maradnak. Megadja viszont azt a lehetőséget, hogy kritikus esetekben az egy időlépéshez tartozó összes iterációs lépést dokumentáljuk, és így az eseményeket a hálózatban a legapróbb részletekig nyomon lehet követni. Ellenőrzi azonban a program azt, hogy az iteráció nem kerül-e egyszerű alternáló ciklusba, (gerjedés két kapukésleltetésnyi periódussal), valamint, hogy a stabilizáció adott (programozható) számú iterációs lépés alatt bekövetkezik-e.

6. Az input szekvencia leírása

A modellezett hálózat működésének a szimulációjához az elsődleges bemenetek mindegyikén meg kell adni a jelek időfüggését. Alkalmas magas szintű leíró nyelv nélkül ez meglehetősen körülményes, lassú és hibaveszélyes. Ezért a magas szintű bemeneti nyelv kialakítása kulcskérdés. A bemenetek szimbolikus kezelése természetes. Emellett azonban két, egymástól lényegesen különböző rendszertechnikai lehetőség kínálkozik.

1. A működtető (teszt) programot az operatív tárban tárolva a szokásos számítógép-programokhoz hasonlóan szubrutinokat és ciklusokat lehet szervezni.
2. Ha a működtető programot a háttérben tartjuk, akkor csak lineáris végrehajtásra van lehetőség, viszont az esetleg hosszú program tárolása az operatív tárat nem terheli.

A μSIM számára a második változatot választottuk, de a program leírásának egyszerűsítése és tömörítése végett többszörösen végrehajtható utasításokat is alakítottunk ki. Ez azt jelenti, hogy algoritmikusan leírható jelek számára generátorokat lehet a bemeneteken deklarálni. Ezek a beépített algoritmus szerint lépésenként automatikusan előállítják a hozzájuk rendelt bemeneteken a jelsorozatot. Így csak azokon a bemeneteken kell a jeleket értékadó utasításokkal leírni, ahol arra alkalmas generátor nem áll rendelkezésre. Ha pedig ilyenre nincs szükség, akkor ismételten végrehajtott „üres" utasítást adunk, és közben a generátorok dolgoznak. Ilyen módon kedvező esetben néhány utasítás-sorban sok-száz lépésből álló tesztprogramot lehet leírni. Az értékadó utasítás maga is igen tömör lehet, ha vektorokról van szó.

Az alábbiakban röviden bemutatjuk a rendelkezésre álló generátorokat. A szintaxisban NAME mindenütt az érintett bemeneti vektort jelöli, „begin" a deklaráláskor adott kezdőértéket jelenti, „entry" pedig azt, hogy a generátor a deklarálást követően hányadik lépésben válik aktívvá.

GEN NAME begin entry t1 t2

Impulzusgenerátor. Itt NAME kivételesen egyetlen bemeneti pontra vonatkozik, amelyen az aktívvá válás után felváltva t1, ill. t2 lépés elteltével történik jelváltás.

CNTU
CNTD } NAME begin entry stay delta

Számláló generátor. Felfelé, ill. lefelé számlál. A belépéstől „stay" lépésenként növekszik vagy csökken az értéke „deltá"-val. A számlálás moduló 2^n rendszerben történik, ahol n a vektor szélessége.

ROR, ROL } NAME begin entry stay delta

Ciklikusan léptető generátor. A kezdőértéket ciklikusan jobbra vagy balra lépteti, a kilépő bitek a másik végen belépnek. A generátor „stay" lépésenként működik, és esetenként „delta" számú léptetést végez.

MARCHR, MARCHL } NAME begin entry stay

„Menetelő" generátor. A ciklikusan léptető generátortól annyiban különbözik, hogy mindig csak egyet léptet, és az egyik széléről kilépő bitet a másik szélén invertálva lépteti be.

Így pl. nulla kezdőérték esetén a memória tesztelésből ismert menetelő minta áll elő.

WALKR, WALKL } NAME begin entry stay

„Sétáló" generátor. Ez a generátor is a ciklikus léptetésen alapul. A legelső működéskor az egyik szélen kilépő bitet a másik szélen invertálva lépteti be, a továbbiakban azonban nem invertál. Így például nulla kezdőérték mellett egy darab 1 sétál végig a vektor elemein. Szemben az eddigiekben ismertetett generátorokkal, amelyek folyamatos működésűek és külön utasítással kell őket leállítani, ez a generátor csak annyi lépést tesz, amennyi a vektor szélessége, és azután automatikusan leáll.

LOGPAT NAME begin entry stay

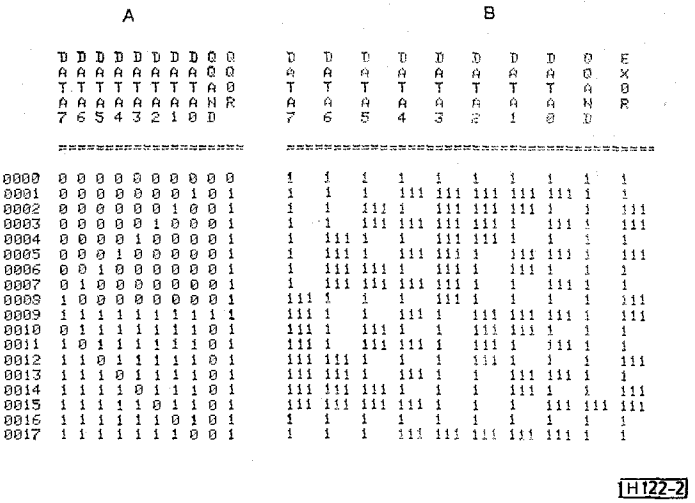
„Sakktábla" generátor. Ez a generátor 1, 2, 4... stb. szélességű sakktábla mintát (1Ø1Ø, 11ØØ, 11110000), valamint az inverzeiket állítja elő kettesével sorba egymás után, mindaddig, amíg ez a szélesség el nem éri a vektor szélességét, és akkor automatikusan leáll. Ezzel a mintával a vektor két tetszőleges eleme közötti egyszeres zárlat detektálható.

RNDS, RNDL } NAME begin entry stay delta

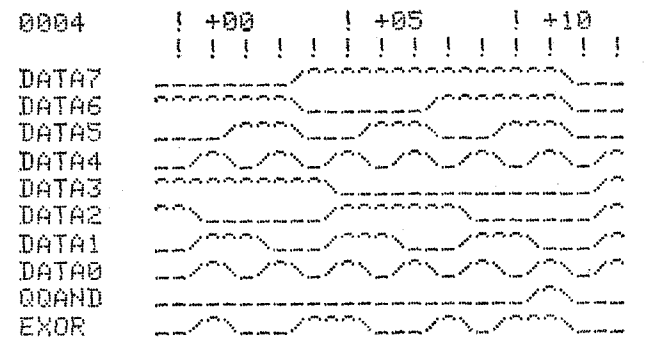
Véletlenszám generátor. Egy byte (short), illetve két byte (long) hosszúságú visszacsatolt léptetőregiszter, amely „stay" lépésenként működik és minden működés alkalmával „delta" számú léptetést végez. NAME szélessége lehet kisebb, mint a generátoré, az összerendelés a legkisebb helyiértéktől indul.

7. Az eredmények dokumentálása

A szimulátor a tesztprogram eredményeit a program futásával párhuzamosan azonnal dokumentálni is tudja. Ilyenkor megjelennek az utasítás-sorok és ezeket követik a generált tesztlépések. Ugyanakkor az eredményeket a program az operatív tárban tárolja is önálló dokumentálás számára. A kimeneti periféria szabadon választható. A dokumentálás bitenként, igazságtábla vagy hullámforma képében történik.



2. ábra. Dokumentációs formák: *a)* igazságtábla; *b)* hullámforma



3. ábra. „Oszcilloszkóp" típusú ábrázolás a képernyőn

(2. ábra) Jelen kiépítésében a program automatikusan a deklarált elsődleges bemeneteket valamint a kimeneti csomópontokat dokumentálja, de lehetőség van egy-egy belső csomópont állapotának az esetenkénti lekérdezésére is.

Lehetőség van továbbá a display képernyőjén „oszcilloszkóp-jellegű" vízszintes időtengelyű ábrázolásra is, max. 2Ø csomópont erejéig. A jelformát alkalmas karakterek segítségével lehet a képernyőre vinni (3. ábra). A képernyőn egyidejűleg egy 3Ø időlépésnyi „ablak" látható, amely a cursor-mozgató billentyűkkel jobbra-balra léptethető.

8. Hibaszimuláció és áramkörmódosítás

A teszt-tervezés megkönnyítésére lehetőség van leragadási (stuck-at) hibáknak az áramkörbe való menetközbeni beiktatására. Leragadási hibák beépíthetők a kapuk be- és kimeneteire, egyidejűleg max. 8 darab. A hibák módosíthatók és külön-külön vagy közös paranccsal megszüntethetők. Hasonló módon lehetőség van a szimulált áramkörnek menetközben való korlátozott mértékű módosítására is, anélkül, hogy az áramkörleírást tartalmazó file módosítására és újbóli

beolvasására szükség lenne. Ez azt jelenti, hogy a hálózat struktúráját leíró táblázat egyes elemeinek az értékét megváltoztathatjuk, de magának a táblázatnak a struktúráját nem. Az egyik ilyen lehetőség az, hogy egy kapu bemenetét át lehet kötni az eredetileg megadott csomópontról egy másikra, a másik pedig az egyszerű kapuk funkciójának a módosítása.

9. Egyéb szolgáltatások

Az interaktív áramkör- és teszt-tervező tevékenységet a program külön szolgáltatásokkal támogatja. Ezeket a lehetőségeket főként konzol bemenetről lehet jól kihasználni. Lehet egyenként visszafelé lépkedni az áramkör megelőző állapotaihoz (Backstep). Adott pillanatban az áramkör állapotát el lehet menteni (Save), majd később, ismételten is, ide vissza lehet térni (Restart). A program kívánságra minden lépésben dokumentálja az iterációs ciklusok számát is. Ha az iterációs ciklusok száma elérte a megengedett maximumot, akkor a program ezt kijelzi.

A bemenő adatok — áramkörleírás és teszt-szekvencia — kötetlen formátumban írhatók le, üres sorokkal és kommentárokkal tagoltan. Beolvasáskor a program alapos szintaktikai vizsgálatot végez, az esetleges hiba körülbelüli helyét kijelzi és hibaüzenetet ad. Kérésre az áramkörleírás alapján fan-out listát is ad, ahol külön megjelöli a nem definiált szimbólumokat, valamint a nem használt (nulla terhelésű) belső csomópontokat.

10. Eddigi alkalmazási tapasztalatok

A program Z80 Assembler nyelven van megírva és egy igen egyszerű disk-kezelő rendszer felügyelete alatt fut. Számos egyszerű áramkörön végzett próbafuttatás már eddig is igazolta a könnyű kezelhetőséget. Felhasználtuk továbbá az „Integrált áramkörök méréstechnikája" c. tantárgy laboratóriumi gyakorlataiban. Sikerrel alkalmaztuk U400 típusú gate-array egyes részleteinek, valamint teszt-szekvenciájának tervezésében.

Végezetül álljon itt egy példa a szimulációs futási idők illusztrálására. Egy főként aszinkron osztólláncokból álló U400-as gate-array némileg egyszerűsített változatát szimuláltuk. 15 bemenete volt, és 206 kaput tartalmazott, jelentős részben mátrix kapukat. Az áramkör minden részét alaposan megmozgató 315 lépésből álló tesztprogram 85 sec alatt futott le (display output mellett). Figyelembe véve a mikroszámítógépektől elvárható teljesítményeket, ezt az eredményt nagyon jónak értékeljük.

Foglalkozunk a továbbfejlesztés gondolatával is. Szándékunkban áll a háromállapotú és a transzfer kapuk modelljét is beépíteni, továbbá megvalósítani, hogy a szimulációs programban a várt eredményeket is meg lehessen adni és a program ellenőrizze ezek megvalósulását. Folyamatban van egy, a programhoz csatlakozó deduktív hibaszimulátor kísérleti kidolgozása.

IRODALOM

[1] *Denneau:* The Yorktown Simulation Engine. IEEE 19th Design Automation Conference, 1982, pp. 55—59.

[2] *Abramovici, Levendel, Menon:* A Logic Simulation Machine. IEEE Trans. Vol. CAD-2, No. 2, 1983. Apr. pp. 82—93.

[3] *Breuer, Friedman:* Diagnosis and Reliable Design of Digital Systems. Computer Science Press, Woodland Hills, USA, 1976, pp. 207—210.