BOHUS MIKLÓS—
CSOPAKI GYULA—FILP ANDRÁS
BME Híradástechnikai Elektronika Intézet
HINSENKAMP ALFRÉD SZKI
MÁTÉ LEVENTE MTA SZTAKI

Digitális berendezések szintézisének számítógépes támogatása

A hardwarefejlesztés folyamatában a megtervezett egység, berendezés működésének számítógépi segítséggel történő funkcionális ellenőrzése jelentős mértékben növelheti a fejlesztési hatékonyságot a berendezés megépítése előtti korai hibafelismerés és egyéb közvetett előnyök révén.

Szimulációs módszerek, programok már hosszabb idő óta ismertek, általános elterjedésüket néhány gyakorlati nehézség akadályozta:

— a kötött szintekkel való megközelítés (alkatrész, kapu, logikai elem, gépi struktúra, gépi architektúra, software implementáció, nyelvek [7], amely nem követi a tervezett eszköz konstrukciós struktúráját, és a szintenkénti automatikus (gépi) átmenet általában nem biztosított;
— vezérlés és adatforgalom különválasztása [7];
— a működtetés bármely magasabb szinten is végül elem (kapu) szintre való visszavezetéssel lehetséges csak, ami már egyszerűbb esetekben is sebesség- és tárkapacitás-problémákra vezet;
— az egyes szinteken kezelt fogalmak, elemek a hardware-fejlesztés fogalomkörétől idegenek, a szimulációt a hardwarefejlesztő csak speciális szakképzettséget igénylő és jelentős mennyiségű előkészítő munka után tudja használni;
— a szimuláció csak tesztadatokkal való működtetésre ad lehetőséget, és általánosságban nincs mód arra, hogy egy alacsonyabb szintű leírást (struktúra) egy magasabb szintűvel (funkcionális specifikáció) összevessünk megfelelésre.

Az itt ismertetett módszer kiküszöböli ezeket a hiányosságokat, és az ismert módszerekhez képest a gyakorlatban jobban, hatékonyabban használható tervezési segédeszközt ad.

A módszer lényege, hogy a szimuláció tárgyát képező egység leírását két oldalról közelíti meg:

— lehetőséget ad a felhasználónak, hogy a szimuláció tárgyát képező egységet fekete dobozként, a be- és kimenetek segítségével specifikálja;
— az egység struktúráját, az alacsonyabb szintű részegységek fentiekkel megegyező funkcionális specifikációja, és ezen részegységek közötti kapcsolat leírása formájában kezeli.

Bár ez a megoldás sem mentesít az eredendő specifikációs hibáktól, de az eddigi módszerekkel szemben az alábbi előnyökkel jár:

— a kétirányú megközelítést a tervezés minden szintjén, a legmagasabb (rendszer) szinttől alkatrészszintig alkalmazva, konzisztens, összefüggő rendszert alkotó ellenőrzésre van mód, miközben a szintenként a fejlesztő által végzett specifikációs szűkítések biztosítják, hogy végül is az egész rendszerre végzett szimuláció csak a tényleg szükséges adatokat kezelje,
— az egység külső specifikálásának szabályozott formája rákényszeríti a fejlesztőt a kiindulási és a közbenső specifikációs szinteken a pontos, ellentmondásmentes és hiánytalan specifikálásra, egyben ezt dokumentálja is.

Követelmények a tervezőrendszerrel szemben a tervező szempontjából

Az elvi kidolgozás fázisában a funkcionális specifikáció szabályozott tartalma és formája segítse a fejlesztőt a pontos specifikálásban.

A formai előírásoknak eleget tevő specifikáció gépi eszközökkel ellenőrizhető legyen nemcsak szintaktikusan, de teljességre és ellentmondás-mentességre is (nem maradtak-e specifikálatlan és nem is tiltott bemenő kombinációk, minden specifikált inputhoz egyértelműen van-e output rendelve, a kapcsolódó felületek összekapcsolhatók-e.

A funkcionális vagy strukturális leírás a szimuláció alapját képezze, egyben az egység dokumentációja is legyen.

A funkcionális specifikáció szabályai változatlan formában legyenek alkalmazhatók a fejlesztés minden szintjén: rendszerszinttől az alkatrészszintig, és legyenek függetlenek a szintek számától.

A szintekre bontást a fejlesztő a konkrét körülményektől (pl. a tervezett eszköz struktúrájától) függően határozhassa meg. Minden olyan esetben, amikor a funkcionális szint megegyezik a konstrukciós szinttel, a strktúraleírásnak összevethetőnek kell lennie a megfelelő szintű konstrukciós leírással (a realizálás szabályainak figyelembevételével).

Támogatnia kell a top-down tervezést és a strukturálhatóságot.

Bármely szinten leírt modell működtethető legyen.

A cikkben leírtak elhangzottak a Budapesti Műszaki Egyetem Villamosmérnöki Karának 1980. jan. 28—29-én megrendezett tudományos ülésszakán.

Beérkezett: 1980. VI. 7.

A szimuláció idehelyesen történjék.

A szimulációs eredmények könnyen kiértékelhetők legyenek [2].

A tervezés folyamata

Egy digitális berendezés tervezése során először a teljes berendezés kívánt működését, viselkedését legáltalánosabban leíró F_1^0 külső funkcionális specifikáció készíthető el [2, 3].

Az F^0 leírás működésének ellenőrzésére a tervező által meghatározott D^0 bemeneti adatok segítségével végrehajtjuk a $Z(D^0, F_1^0)$ szimulációt, ami a P^0 eredményeket szolgáltatja. Ezen eredmények kiértékelésével a tervező megállapítja, hogy az F_1^0 funkcionális specifikáció helyes-e.

Általában a legmagasabb szintű funkcionális modellekből több lépésben végzett finomítással jutunk el a tényleges megvalósításig. A finomítás miként jét, tehát az alacsonyabb szintű tervezési lépés konkrét tartalmát, módját nyilvánvalóan a tervező határozza meg, ezért lényeges, hogy egy-egy egység (részegység) külső funkcionális specifikációját annak későbbi konkrét megvalósítási módjától függetlenül, csak a külső jellemzők felhasználásával (F_1^0) írjuk le. Mihelyt a tervező megtervezte eggyel alacsonyabb szinten ezt az egységet, tehát alacsonyabb szintű összetevőket struktúrába rendezett, az összetett funkcionális pecifikációt is finomítani tudja. Ezt úgy végzi, hogy leírja a T_i típusokkal realizált E_j-elemek egy olyan S összekapcsolódását, struktúráját, amely az F^0 funkcionális leírással egyenértékű, azaz

$$F_1^0 \Rightarrow S_1^0\{E_j^1\},$$

ahol

S_1^0 a nulladik szintű struktúra leírása,

E_j^1 a nulladik szintű struktúra j-edik eleme, amelyre igaz, hogy

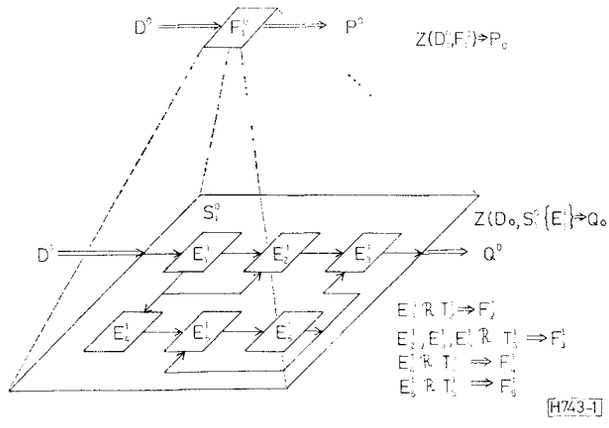
$$E_j^1 \mathcal{R} T_i^1.$$

$\mathcal{R}$ a realizáció jele (azaz a T_i^1 típus realizálja E_j^1-et).

Az $S^0\{E_j^1\}$ strukturális leíráson a D^0 bemeneti adatokkal elvégezve a

$Z(D^0, S_1^0\{E_j^1\})$ szimulációt, a

kapott Q^0 eredményeket össze kell vetni a P^0 eredményekkel.



1. ábra. A tervezés folyamata

A nulladik szintű struktúra E_j^1 elemeit realizáló T_i^1 típusokat először önmagukban specifikáljuk azok F_i^1, önálló külső specifikációjával. Az F_i^1-ek az F_1^0-nál eggyel alacsonyabb szintű funkcionális specifikációk halmazát képezik. Ha ezek mindegyikén alkalmazzuk az F^0-ra leírtakat, láthatóan rekurzív tervezési eljárásra jutottunk, ahol az egymás utáni

$$F^x \Rightarrow S^x\{E^{x+1}\} \Rightarrow T^{x+1} \Rightarrow F^{x+1}$$

szintváltások addig folytatódnak, ameddig olyan T^y típusokhoz jutunk, amelyeknek már van ismert áramköri megvalósításuk. A tervezés ilyen módja csupán azt kívánja, hogy T^y megvalósítása már ismert legyen, nem tételez fel semmit annak összetettségéről (lehet integrált áramkör, szerelt kártya vagy akár mikroszámítógép is).

Könnyen belátható, hogy azzal, hogy egy-egy lépés során nem szükséges két szintnél [a vizsgált egység (F^x) és az azt megvalósító struktúraelemek szintjénél (E^{x+1})-nél] többet kezelni, kiindulási követelményeink jelentős része egyszerűen teljesíthető. A lényegesebb következmények:

- lehetővé válik a top-down megközelítés, mert a struktúraelemek specifikációja az azoktól elvárt, nem pedig a realizáció eredményeként bizonyított működés. Ennek következtében
- a gépi leírás csak a tényleg szükséges (elvárt, specifikált) funkciókat tartalmazza explicit formában, és nem kezeli az össze lehetséges funkció kiértékeléséhez szükséges adatokat. Ez lényeges tárterület- és sebesség-nyereséget hoz, viszont így
- nincsen közvetlen lehetőség annak megválaszolására, hogy hogyan viselkedik az egység új, korábban nem specifikált körülmények között. Ez értelemszerűen már nem top-down, hanem egy bottom-up megközelítést jelent.

Rövid irodalmi áttekintés

A digitális rendszerek működésének szimulációjára, illetve tervezésük verifikálására vonatkozó irodalom igen bőséges. Közülük csupán néhány olyan jelentősebbet vizsgálunk röviden, amelyek egy-egy sajátos vonásukkal kapcsolódnak az előzőekben ismertetett koncepcióhoz.

Az első szimulációs nyelvek általában a regiszterek közötti adatmozgatás szintjén írták be a szimulálandó objektumokat. Legjellegzetesebb képviselőjük a Y. Chu által definiált CDL nyelv [5, 6], illetve ennek R. Hartenstein által továbbfejlesztett változata [7]. Bár ez utóbbi törekszik a strukturált leírásra, de ezt eléggé megnehezíti azzal, hogy a nyelv alapoperátorai igen erősen kötődnek egy adott funkcionális elemkészlethez (regiszter, dekóder, kapcsoló stb.), és minden struktúrát ezekre kell visszavezetni. Nem segíti elő a hatékony strukturálást az sem, hogy a CDL/KA időkezelése eléggé leegyszerűsített, nem értelmez eredményeket. Előnyös sajátossága viszont, hogy igen gazdag az a műveletkészlet, amit az adatokon értelmez. Ezek alapján a CDL/KA igen kényelmes regiszter transzfer szintű leírást és szimulációt enged meg.

A legnagyobb működő rendszerek egyike a Bell Laboratories-ban kifejlesztett LAMP rendszer [8]. A LAMP leíró nyelve az LSI—LOCAL már sokkal inkább hardware-leírás orientált, mint a CDL. Az LSI—LOCAL nyelv szintén a regiszter transzfer szintű leírás eszköze. A LAMP rendszer nagyon sokféle módon segíti a digitális berendezések tervezőit és üzemeltetőit, ezek közül a szimuláció csupán egy lehetőség. A CDL/KA-hoz hasonlóan kevés a berendezés többszintes, strukturált leírásához nyújtott támogatása.

A Kaliforniai Egyetemen kifejlesztett SARA rendszer két független eszközt biztosít a többszintes modellezéshez [9, 10]. Külön formalizmus használatos a struktúra leírására és a viselkedés vagy funkció megadására. Ez egyben a rendszer egyik kedvezőtlen sajátossága is. A struktúra leírására három fogalom: a modul, az összekapcsolás és a tok vagy elem szolgál. A funkcionális leírás alapvető eszköze két gráf: a vezérlési gráf és az adatgráf. Az adatgráf csomópontjai adathalmazok vagy rajtuk műveleteket végző processzorok lehetnek, a gráf élei a hozzáférési lehetőséget mutatják. A vezérlési gráf határozza meg, hogy az adatgráf processzorai milyen sorrendben válnak aktívvá. A gráfok leírása egy PL/1-szerű nyelven történik.

A rendszerben a modellek leírása több szinten lehetséges oly módon, hogy a magasabb szint elemei az alacsonyabb szint összetett moduljaival helyettesíthetők. Megtehető ez a funkcionális leírás gráfjával is, ahol egy csomópont helyettesíthető az alacsonyabb szint egy teljes gráfjával.

Az általunk kialakított koncepcióhoz legközelebb álló többszintes szimulációs rendszer a Stanford Egyetemen létrehozott SABLE [11]. A SARA rendszerhez hasonlóan itt is külön eszköz szolgál a struktúra leírására, (ez a struktúra tervező nyelv, SDL [12]) és egy másik, algoritmikus nyelv (az ADLIB) teremti meg a funkcionális leírás lehetőségét. Az ADLIB nyelv a PASCAL-hoz igen hasonló, típusos nyelv, tömör áttekinthető leírások készítésére alkalmas. A funkcionális leírás legfontosabb eszköze az összetevőtípus és a kapcsolódástípus. Az összetevőtípus a modell egyes összetevőinek viselkedését, működését írja le. A leírás nem más, mint a bemeneteken történő változásokra való reagálás megadása. A kapcsolódástípus adja meg azokat az adat struktúrákat (pl. byte, ASCII karakter, BCD szám), amelyek az összetevőtípusok be- és kimenetein értelmezettek.

A különböző szintű összetevőtípusok általában más adatstruktúrákat használnak fel. Az adatstruktúrák egymásba alakítására transzlátorok szolgálnak, ezeket néhány egyszerű eset kivételével a felhasználónak magának kell megírnia. A transzlátorokat a rendszer illeszti be a megfelelő adatsturktúrák közé.

Az irodalomban ismertetett rendszerek felépítési elveit az általunk kidolgozott koncepcióval egybevetve, a következő jelentősebb eltérések fedezhetők fel:

— vagy nem adnak a vizsgált rendszerek módot külön funkcionális és srukturális leírásra, vagy ezt külön-külön formalizmussal teszik lehetővé;
— egyetlen rendszer sem hangsúlyozza olyan erősen az eseményorientáltságot, mint azt koncepciónk teszi;
— egyetlen rendszer sem biztosítja események strukturálását magasabb szintű eseményekké;
— általában gazdagabb operátorkészletet biztosítanak, ami elsősorban az algoritmikus, regisztertranszfer szintű szimulációt teszi igen egyszerűvé.

A leíró nyelv koncepciója

A nyelv felépítése során az alábbi követelményeket kell egyidejűleg kielégíteni:

— egységes formalizmus használatával többszintűség megvalósítása a struktúra leírásában, az időbeli működés nyomonkövetésében, valamint a működtetéshez szükséges adatok felépítésében;
— a struktúrától teljesen függetlenül megadott külső funkcionális specifikáció és a struktúra működésének összevetése;
— a legalacsonyabb szintű struktúra leírás felhasználható legyen egy konstrukciós tervező rendszer bemenő adataként, illetve összevethető legyen egy meglevő konstrukciós leírással;
— legyen mentes feleslegesen korlátozó megkötésektől.

A nyelv elemei [2]

A nyelv a leírt egység egy-egy szintjén belül maga is hierarchikus felépítésű. Legalacsonyabb szintű elemei a JELEK és az ezekre értelmezhető elemi ESEMÉNYEK. Ezekből szükség szerint magasabb szintű, összetett események is képezhetők.

Jelek és események rendezett halmazaként előáll

— egy-egy konkrét részegység (struktúraelem) külső specifikációja, a TÍPUS. Lényeges rögzíteni, hogy a TÍPUS a részegység saját jellemzője, függetlenül attól, hogy milyen struktúra részét képezi és milyen beépítési körülmények között;
— a struktúraelemek összekapcsolásával előáll a struktúra és annak gépi leírása a MODELL.

JEL

A jel a legalapvetőbb nyelvi objektum, amelyet

— neve (azonosítója),
— hossza (bitszáma),
— reprezentációja (számrendszere) és
— értékkészlete

definiál.

A funkcionális leírásban szereplő jelek és azok konstrukciós megvalósítása közötti kapcsolat általában triviális.

ESEMÉNY

Eseményen adott jelek meghatározott időviszonyokkal jellemzett, meghatározott értékváltozását értjük. A könnyebb kezelés és eligazodás érdekében megkülönböztetünk elemi és összetett eseményt.

Az elemi esemény egy jel meghatározott értékváltozását jelenti. Az elemi eseményt leírja *(2. ábra)* az elemi esemény neve, a jel neve, a jel értékváltozása (hatása), az értékváltozás időpontja (az elemi esemény időpontja) és az értékváltozás feltétele.

Az összetett esemény elemi események rendezett halmaza. A rendezettség mértéke és jellege esetenként erősen eltérő lehet.

Az összetett eseményt az összetett esemény neve, az összetevő események és a közöttük levő reláció és az összetett esemény bekövetkezésének időpontja írja le.

Miután definíciónk tudatosan összetevő eseményeket és nem összetevő elemi eseményeket tartalmaz, látható, hogy az események tetszőleges szintszámmal strukturában is lehetnek összetettek.

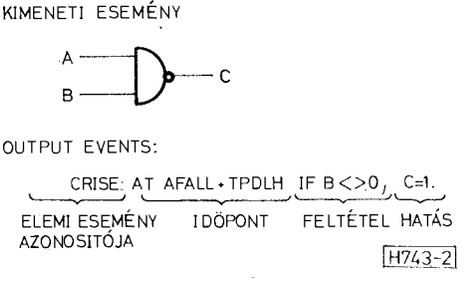
A teljesség igénye nélkül felsorolunk néhány relációt elemi események között:

— események sorozata (lazán vagy szigorúan kötött sorrenddel, ismétlődéssel vagy anélkül stb.);
— egyidejű események logikai kapcsolatban;
— soros és párhuzamos események vegyes halmaza.

TÍPUS

A típus egy áramköri egység megadása ki- és bemeneti neveivel, valamint a bemenetein értelmezett bemeneti események és a hatásukra a kimenetek megváltozását létrehozó kimeneti események leírásával. A típus tehát egy összetett nyelvi eszköz, amely a tervezés alatt álló T_i objektum leírását adja.

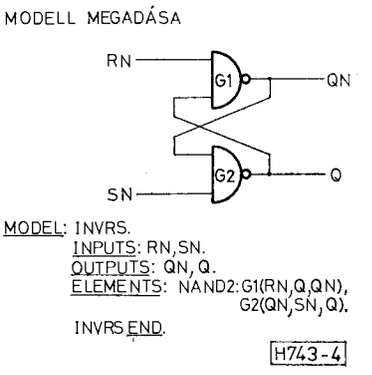
A típus megadása során a tervező függetlenítheti



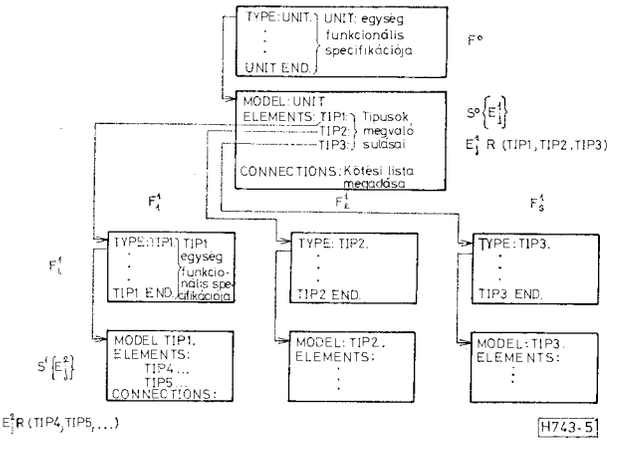
2. ábra. Az elemi esemény

```
TYPE: NAND2 A,B,C.
INPUTS: A,B.
OUTPUTS: C.
INPUT EVENTS: INRISE: ANY INPUTS CHANGE TO 1;
              INFALL: ANY INPUTS CHANGE TO 0.
OUTPUT EVENTS:
        OUTRISE: AT INFALL + 22 NS,
                 IF NO INPUT = 0, C=1;
        OUTFALL: AT INRISE + 15 NS,
                 IF ALL INPUTS=1, C = 0.
NAND2 END.                                  H743-3
```

3. ábra. Típus megadása



4. ábra. Modell



5. ábra. Többszintű leírás

magát attól a struktúrától, amellyel a funkcionális specifikációt meg kívánja valósítani. Típusként tehát tetszőleges bonyolultságú egység szerepelhet *(3. ábra)*. Lehetővé teszi a nyelv, hogy már meglevő típusok felhasználásával újabbakat hozzunk létre, így a típus a strukturálás egyik eszköze.

MŰKÖDÉS

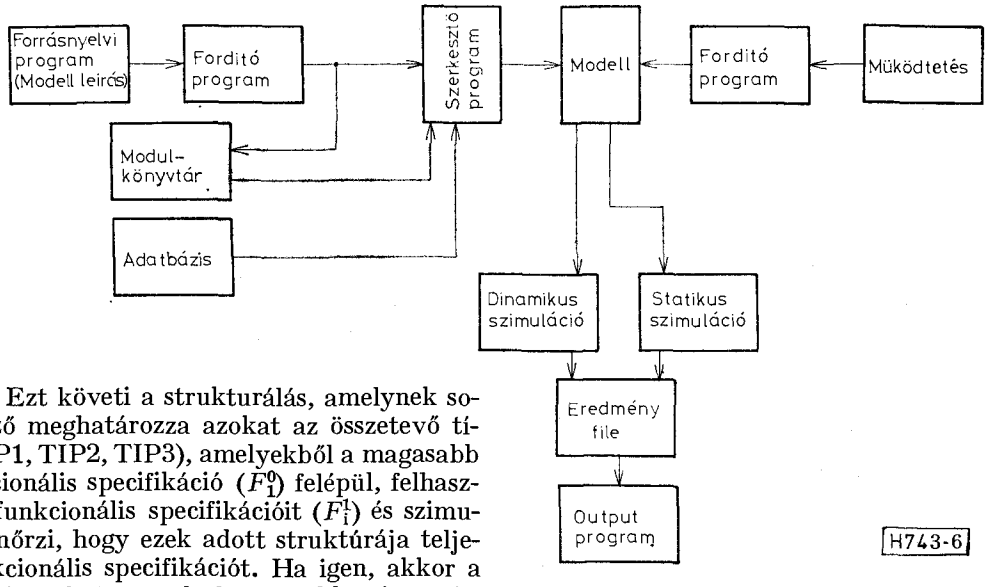
A típusokban az elemi eseményeken kívül egy bonyolultabb nyelvi elem, a működés is felhasználható. A működés az elemi eseményektől eltérően nem időpontot, hanem időtartamot jelez. A működés leírásához meg kell adni kezdetének időpontját, feltételét (ha van) és befejeződésének idejét és hatását. Működésnél megadható olyan leállító feltétel, amelynek teljesülése félbeszakítja a működés folyamatát. A működés ideje alatt a benne levő jelek értéke határozatlan, és csak a működési idő eltelte után van értékük.

Modell

A modell a struktúra gépi képe, leírása *(4. ábra)*. A modell leírásához megadandók: a bemenőjelek, a kimenőjelek, a modellbeli típusok és azok realizációi, a kötési lista.

Többszintű leírás

Egy adott berendezés felülről lefelé való tervezése során különböző szintű leírások keletkeznek *(5. ábra)*. A tervező először rendszerint a berendezés működését a legmagasabb szintű funkcionális specifikációval



6. ábra. A szimulációs rendszer felépítése

írja le. (F^0). Ezt követi a strukturálás, amelynek során a tervező meghatározza azokat az összetevő típusokat (TIP1, TIP2, TIP3), amelyekből a magasabb szintű funkcionális specifikáció (F_1^0) felépül, felhasználva azok funkcionális specifikációit (F_i^1) és szimulációval ellenőrzi, hogy ezek adott struktúrája teljesíti-e a funkcionális specifikációt. Ha igen, akkor a felhasznált típusokat eggyel alacsonyabb szinten ismét struktúraként írja le, amelyek szintén típusokból állnak. Ez a folyamat addig folytatódik, amíg olyan egyszerű típusokhoz jutunk, amelyeket már célszerűtlen tovább osztani, mivel egyetlen konstrukciós egységként (pl. IC, szerelt kártya stb.) rendelkezésre állnak (5. ábra). A tervezési eljárást támogató nyelvet a cikk folytatásában ismertetjük.

A szimuláció folyamata [1]

A szimulációs nyelven leírt modellek, amennyiben szintaktikailag helyesek és specifikációik ellentmondásmentesek, bekerülnek a modulkönyvtárba (*6. ábra*).

A fordítóprogram tevékenységei:

- szintaktikai ellenőrzés;
- vizsgálat ellentmondásmentességre a típuson önmagán belül, valamint a külső funkcionális specifikáció és a struktúra leírása között;
- konzisztencia-vizsgálat a funkcionális specifikáció leírására;
- belső ábrázolási forma létrehozása;
- működtető jelsorozatok szintaktikai ellenőrzése.

A szimulálandó rendszer összeállítását a szerkesztőprogram végzi.

A szerkesztőprogram tevékenységei:

- összeszerkeszti a modell-leírás elemeit;
- az adatbázisban szereplő elemek adatait beépíti a modellbe;
- futtatható állapotú modellt hoz létre a szimulátorok számára.

A szimulációs rendszer szolgáltatásai:

- dinamikus szimuláció (a működtetendő rendszer elemeinek időhelyes működtetését végzi);
- az alacsonyabb szintű típusokból álló struktúra működtetésének a magasabb szintű funkcionális leírással történő összevetése (ehhez az elemi események szekvenciáiban fel kell ismernie a magasabb szintű összetett eseményeket, működéseket);
- bonyolultabb adatszerkezetek összeállítása az alacsonyabb szint egyszerűbb szerkezeteiből;
- statikus szimuláció (a modellbeli elemek működési idejét egységnyinek tekinti, így csak a funkcionális működés ellenőrzésére szolgál);
- output program (a szimulátorok által létrehozott eredmény file adatait írja ki a felhasználó által kívánt formátumban).

IRODALOM

[1] *Bohus M.—Csopaki Gy.—Filp A.—Gruber G.—Hinsenkamp A.—Jagudits L.—Tagányi Gy.—Theisz P.:* Számítógépek és részegységeinek szimulációja. Tervtanulmány az SZKI számára, Budapest, 1979. június

[2] *Bohus M.—Csopaki Gy.—Filp A.—Jagudits L.—Tagányi Gy.:* Digitális rendszerek többszintes, időhelyes szimulációjának rendszerterve. Leíró nyelv. Tervtanulmány az SZKI számára, Budapest, 1979. december

[3] *Máté L.:* APGS kidolgozása nem egyenlő leírási szintű digitális rendszerek szimulációjára Bp. 1976. Műszaki Terv az AMT AT—3 számára

[4] *Hinsenkamp A.:* Hardware tervezés gépi segítése a szimuláció fogalomkörébe tartozó módszerekkel Bp. 1978. SZKI Hardware Laboratórium. Kézirat

[5] *Chu, Yaohan:* An ALGOL-Like Computer Design Language Comm. of ACM. October, 1965. pp. 607—615

[6] *Chu, Yaohan:* Computer Organization and Microprogramming. Prentice Hall Inc. New Yersey, USA

[7] *Hartenstein, R.:* Fundamentals of Structured Hardware Design

[8] *Chang, H. Y.—Smith, G. W.—Walford. I. R. B.:* LAMP: System Description. BSTJ. vol. 53. pp. 1431—1449, October, 1974

[9] *Gardner, R. I.:* Multi-level Modeling in Sara Proc. of Symposium on Design Automation and Microprocessors Palo Alto, Feb. 24—25. 1977. pp. 63—67

[10] *Razouk, R. R.:* The Graph Model of Behavior Simulator. Proc. of Symposium on Design Automation and Microprocessors, Palo Alto, Feb. 24—25. 1977. pp. 63—67

[11] *Hill, D.—van Cleemput, W.:* SABLE a Tool for Generating Structured Multi-Level Simulations. Proc. of 16th Design Automation Conference, San Diego, 1979. pp. 272—279

[12] *van Cleemput, W.:* A Hierarchical Language for the Structural Description of Digital Systems. Proc. of 14th Design Automation Conference, New Orleans, 1977. pp. 377—385.