Hardware leíró nyelv digitális berendezések többszintű leírására

CSOPAKI GYULA
BME Híradástechnikai Elektronika Intézet

ÖSSZEFOGLALÁS

A cikk áttekinti a digitális berendezések többszintű leírására szolgáló hardware leíró nyelvekkel szemben támasztott követelményeket. Az ismertetésre kerülő hardware leíró nyelv a CARS (Computer Aid for Recoursive Synthesis) rendszer számára került kifejlesztésre. A nyelv egyik legfontosabb jellegzetessége, hogy a tervező ugyanazon nyelvi elemeket használhatja mind a funkcionális, mind a strukturális specifikációra, a tervezés bármely szintjén.

CSOPAKI GYULA

Villamosmérnök oklevelét 1969-ben a Budapesti Műszaki Egyetem Villamosmérnöki Karán szerezte meg. 1969-től a BME Híradástechnikai Elektronika Intézetében az MTA Informatikai és Elektronikai Tanszéki Kutatócsoportjának tudományos munkatársa; szakterülete digitális berendezések számítógéppel segített tervezése.

Bevezetés

Digitális berendezések tervezése során a tervezési hatékonyság nagymértékben növelhető a megtervezett egységek számítógéppel történő ellenőrzése által. A számítógépes ellenőrző programok számára a berendezés működését hardware leíró nyelvek segítségével specifikáljuk. A tervezési folyamat során a felülről lefelé történő megközelítés történik, kiindulva a rendszerszinttől, eljutva az alkatrészszintig. A jelenlegi tervezőrendszerek általában nem biztosítják azt, hogy az egyes tervezési szinteken a tervezendő egység működését azonos nyelvi eszközökkel specifikáljuk. A továbbiakban ismertetjük egy ilyen hardware leíró nyelvvel szemben támasztott követelményeket, majd tárgyaljuk a nyelv alapelemeit, az általuk leírható funkciókat és struktúrákat.

A többszintű hierarchikus tervezés folyamata

A leíró nyelv alapvetően a felülről lefelé való tervezési folyamatot támogatja, azáltal, hogy lehetővé teszi a magasszinten definiált funkcionális egység egyértelmű működésének megadását mint külső specifikációt. A hardware leíró nyelv támogatja és egyben kényszeríti is a tervezőt a pontos és egyértelmű specifikációra. A funkcionális egység helyes működését számítógépes szimulációval ellenőrzi a tervező a bemenetre kerülő jelszekvenciákra a kimeneten kapott jelsorozatokkal. Ezt követőleg a tervező felépíti a funkcionális egységet mint struktúrát alacsonyabb szintű funkcionális egységek összekapcsolásából.

Az így felépített struktúrát ugyanazzal a bemenő jelsorozattal működtetve, mint a funkcionális egységet, hasonlóképpen egy kimeneti jelsorozatot kapunk. A két eredmény összevetéséből megállapítható, hogy a struktúra megvalósítja-e a kívánt funkciót. Ha igen, akkor a struktúrát megvalósító funkcionális részegységeket hasonlóképpen felépítjük struktúraként eggyel alacsonyabb szinten specifikált és ellenőrzött részegységekből és ellenőrizzük a struktúrát a fentieknek megfelelő módon. A tervezési folyamat akkor ér véget, ha a struktúrát megvalósító részegységek már előzőleg megtervezett funkcionális egységek vagy katalóguselemek [3].

A tervezési folyamat során a funkcionális egységek és a struktúrák működésének specifikációja hardware leíró nyelv segítségével történik [2].

A hardware leíró nyelvvel szemben támasztott követelmények:

- a tervezési folyamat minden szintjén azonos nyelvi eszközökkel biztosítsa a funkcionális működés és a struktúra leírását;
- tegye lehetővé a működéseknek és a működések feltételének az időben helyes leírását;
- biztosítsa az időbeli teljes finomszerkezet leírásán kívül az általánosabb időbeli leírást;
- a nyelv eszközei és elemei olvasható és öndokumentáló szintaktikai szerkezetek létrehozását tegyék lehetővé;
- a nyelv szintaktikai szerkezete olyan legyen, hogy a struktúráit programozás eszközeivel elemezhető legyen.

A hardware leíró nyelv elemei [1, 5]

A hardware leíró nyelvnek a következő fő építőelemei vannak:

- specifikálni kell a funkcionális egység vagy a struktúra bemenő-, kimenő- és buszjeleit. Ezekre a jelekre vonatkozóan meg kell adni a jel bitbeli méretét, típusát és a megengedett értékek halmazát;
- nyelvi eszközökkel biztosítani kell a funkcionális egység bemenetein történő elemi változások leírását elemi bemeneti eseményekként;
- lehetővé kell tenni a bemeneten zajló jelszekvenciák leírását összetett bemeneti eseményekként;
- a funkcionális egység bemenetén történő válto-

Beérkezett: 1984. IX. 15. (#).

zások hatására a kimeneteken kimeneti eseményekkel írhatjuk le a változásokat.

A kimeneti események a bemeneti változásokhoz viszonyítva adott késleltetéssel történnek meg. A kimeneti eseményeknek ebben az adott időpontban való bekövetkezése lehet feltételhez rendelve. A feltételrész vonatkozhat jelek értéke logikai kapcsolatának igaz voltára, de biztosítja a feltételrész adott jelértékek időtartamban való vizsgálatát.

- a nyelvi elemeknek biztosítani kell az időzítési viszonyok általánosabb kezelését, amikor is kimeneti jelek értékváltozásait nem konkrét időpontokhoz rendeljük, hanem adott időintervallumhoz;
- biztosítani kell a struktúrák felépítéséhez a kötési listák leírását;
- lehetővé kell tenni a funkcionális egységeket és struktúrákat működtető bemeneti jelsorozatok leírását.

A letrásokon végezhető ellenőrzések

A funkcionális egységek vagy struktúrák leírása működtethető bemeneti jelsorozatokkal, ezáltal az egység működése ellenőrizhető, hogy végrehajtja-e a kívánt funkciót. Ezt a szimulációs úton történő ellenőrzést következetesen végrehajtva a funkcionális egységekre és az őt megvalósító struktúrára a tervezési folyamat nagymértékben támogatható [4].

A funkcionális egység és az őt megvalósító struktúra által ugyanarra a bemenő jelsorozatra adott kimeneti jelsorozat összevethető, az eltérések az összehasonlító program által kijelezhetők.

Az egyes funkcionális leírások vagy struktúrák ellenőrizhetők teljességre és ellentmondásmentességre. Teljességre történő ellenőrzés során megvizsgálható, hogy az adott bemeneti változások mindegyikéhez definiálva lett-e kimeneti esemény. Megengedhető, hogy a bemeneti események ne fedjék le az összes lehetséges változások halmazát, azonban az szükséges, hogy a definiált értékkészletet teljesen meghatározzuk. Az ellentmondásra történő ellenőrzés azt vizsgálja meg, hogy adott bemeneti jelváltozásokhoz nincs-e két különböző egymásnak ellentmondó kimeneti jelváltozás rendelve.

A leíró nyelv elemei

Külső jelek specifikálása

A tervezés során a specifikálandó áramköri egység bemeneteit, kimeneteit és buszjeleit egyértelműen definiálni kell.

A bemeneti jeleket a következő attributumok határozzák meg:

- bitbeli szélessége;
- számrendszere;
- értékkészlete.

A fenti attributumokon kívül azonosítóval kell ellátni a bemenetet, hogy a bemenetre a leírásban hivatkozni lehessen.

A kimeneti jeleket meghatározó attributumok:

- a jel bitbeli szélessége;
- számrendszere;
- értékkészlete;
- a kimenet típusa.

A be- és kimeneti jeleknél a bitbeli szélesség megadása decimális számmal történik, melyet a BITS alapszó követ. A számrendszer megadásakor definiálni kell, hogy a jel bitjeit hány darab oktális, decimális vagy hexadecimális számjegyként értelmezzük. Bináris értelmezés esetén a számrendszer megadása elmarad.

A bemeneti és kimeneti jelek értékkészletét azáltal specifikáljuk, hogy megadjuk mely értékeket vagy értéktartományokat nem veheti fel a jel. A meg nem engedett értékeket az EXCEPT alapszót követően zárójelek között kell felsorolni.

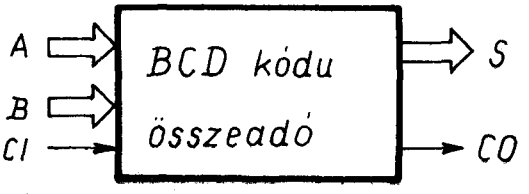
A kimenet típusa lehet ellenütemű vezérlésű totem-pole, nyitott kollektoros vagy háromállapotú. Alapértelmezésben a kimenet típusa ellenütemű vezérlésű, a nyitott kollektoros kimenet jelölése OC-vel, a háromállapotú kimenet jelölése TS-sel történik.

A síneket meghatározó attributumok:

- bitbeli szélessége;
- számrendszere;
- értékkészlete;
- a sín típusa;
- a jelterjedés iránya.

A sínen a jelterjedés lehet kétirányú vagy kimeneti. Azonosításukra a BIDIRECTIONAL és az OUTPUT alapszavak szolgálnak.

Mintaként definiáljuk egy egydekádos BCD összeadó bemeneti és kimeneti jeleit (1. ábra).



1. ábra. BCD kódú összeadó be- és kimenő jelei

```
INPUTS: A,B 4 BITS, 1 DEC DIGIT; CI.
OUTPUTS: S 4 BITS, 1 DEC DIGIT; CO.
```

A bemeneti jelek közül az A és B négy bites, a CI egy bites, a kimeneti jelek közül S négy bites, a CO egy bites. Az A,B és S jelek decimálisként vannak értelmezve.

Bemeneti események

A funkcionális egységek bemenetén lezajló változások bemeneti eseményekként írhatók le. Elemi jelek egyszerű változását elemi bemeneti eseményeknek nevezzük. A következő jellegű változások specifikálhatók bemeneti eseményként (zárójelben a nyelvbeli megfelelő alapszavak):

— adott bemeneti jel megváltozása:
jel CHANGES;
— adott bemeneti jel határozott értékű megváltozása:
jel CHANGES_TO érték;
— bármely bemeneti jel megváltozik:
ANY_INPUTS CHANGES;
— bármely bemeneti jel adott értékű megváltozása:
ANY_INPUTS CHANGES_TO érték;
— az összes bemeneti jel megváltozása:
ALL_INPUTS CHANGE;
— az összes bemeneti jel határozott értékű megváltozása:
ALL_INPUTS CHANGE_TO érték.

A fenti BCD kódú összeadó bemenetén ha valamilyen változás történik, akkor ki kell értékelni a kimenetet. Az ilyen jellegű változások leírására a bármely bemenet megváltozása típusú esemény alkalmas. A bemeneti eseményeket azonosítóval is el kell látni, hogy az eseményekre hivatkozni lehessen. Legyen a szóban forgó bemeneti esemény azonosítója INCHG.

A bemeneti események definícióját mindig az INPUT_EVENTS alapszó vezeti be. Az esemény leírása ezután:

INPUT_EVENTS: INCHG: ANY_INPUTS CHANGES.

A fenti elemi bemenő eseményekkel bármilyen bemeneti változás leírható. Az elemi bemeneti eseményekre adott válaszfüggvények kimeneti eseményekkel definiálhatók.

Abban az esetben, ha a kimeneten akkor történik változás, ha a bemeneten adott jelszekvencia érkezett, elemi bemeneti változásokkal nem lehet egyszerűen leírni. Bemeneti jelszekvenciák képzése és leírása a bemeneti összetett eseményekkel történik. Bemeneti összetett eseményként definiálható elemi bemeneti események adott sorrendű előfordulása kötött időzítésű sorrenddel. Ilyen esetben definiálni kell, hogy az összetevő eseményeknek milyen sorrendben és milyen időzítéssel kell egymást követniük.

Összetett bemeneti eseményként definiálható továbbá adott elemi bemeneti események nem kötött sorrendű előfordulása adott időtartamon belül. Összetett bemeneti eseményként definiálható az összes felsorolt elemi bemeneti esemény egyidejű előfordulása, vagy csak az egyik, vagy bármennyi egyidejű előfordulása. Ilyen esetekben az összetett esemény bekövetkezését vagy meghiúsulását kijelzi a rendszer. Ezáltal a specifikáció megvalósulását vagy meghiúsulását a futtató rendszer közli a tervezővel.

A fenti összetett esemény-konstrukciók igen nagy mértékben támogatják a tervezőt a bemeneten történő változások egzakt időhelyes leírásában.

Kimeneti események

A funkcionális egység bemenetén történő változások hatására kimeneti változások történnek. A kimeneti változásokat kimeneti eseményeknek nevezzük. A kimeneti eseményeket a következő attributumok határozzák meg:

— bekövetkezési időpont;
— bekövetkezés esetleges feltétele;
— hatásrész.

A bekövetkezés időpontját meghatározza a kimeneti eseményt kiváltó bemeneti esemény és az esetleges késleltetési idő.

A bekövetkezésnek lehetnek feltételei, melyeket logikai kifejezésekként írhatunk le. A logikai kifejezés operandusai jelek és konstansok lehetnek, logikai operátorként logikai és, logikai vagy/és negációs operátorok szerepelhetnek. Mivel a kimeneti események alapvetően a többszintű időhelyes leírást támogatják, biztosítani kell nyelvi eszközökkel a beállási és tartási idő jellegű feltételek leírását, továbbá jelek adott időtartambeli konstans vagy nem konstans értékének vizsgálatát. A vizsgált időponthoz képest múltbeni és jövőbeni időtartam specifikálására szolgáló szerkezet: FROM ⟨TIME1⟩ TO ⟨TIME2⟩; ahol TIME1 az intervallum alsó, a TIME2 az intervallum felső határát specifikálja. Az intervallum alsó és felső határát eseménynévvel és egy relatív késleltetési idővel lehet megadni. Ha az intervallum felső határa a referenciaidőpont akkor az intervallum specifikációja: SINCE ⟨TIME1⟩ ahol a TIME1 eseménynévvel és relatív késleltetési időponttal specifikálható.

A kimeneti események a funkcionális egységekbeli történések teljes időbeli és funkcióbeli finomszerkezetének leírását adják.

A hatásrészben leírható történések logikai kifejezések vagy memóriaműveletek lehetnek. A logikai kifejezésekkel jelek új értékét definiálhatjuk. Memóriaműveletként memóriába való írást és memóriából való olvasást specifikálhatunk. A memóriákat nem kell külön specifikálni, hivatkozni memóriaelemekre a címregiszter és az adatregiszter segítségével lehet.

Ha a teljes időbeni finomszerkezet leírására nincs mód, mert olyan bonyolultságú funkcionális elemeket alkalmazunk, amelyekre vonatkozólag nincsenek egzakt időpontok, a jelváltozásokra akkor úgynevezett működéseket alkalmazunk.

A működések esetén definiálandó a működés kezdetének időpontja, a működés befejezésének időpontja vagy feltétele és a funkció. A kezdeti időpont megadására egy referencia esemény és az ehhez viszonyított relatív késleltetés szolgál. Az indításhoz ezenkívül feltétel is rendelhető. A feltételes kifejezés jelek értékeire hivatkozhat időparaméterekkel.

A működés befejezésének megadására vagy egy referenciaesemény és az esetlegesen szereplő relatív késleltetési idő vagy a befejeződési feltétel szolgál. A befejeződést megszakítás jellegű feltételek bekövetkezése idézheti elő, tehát bizonyos jelek adott értékre történő megváltozása.

A hatásrészben jelek értékét lehet definiálni aritmetikai vagy logikai kifejezésekkel. Az értékadó utasítások lehetnek feltételes vagy feltétel nélküliek. A hatásrész tartalmazhat továbbá memóriaműveleteket. Memóriaműveletek (írás, olvasás) esetén a memória azonosítása cím- és adatregiszterével történik, a memóriákat sem deklarálni, sem névvel el-

látni nem kell. A memóriarekeszek deklarációja előfordulásuk által történik.

A működések indítási és befejeződési időpontja között a működés során értéket kapó jelek értéke elérhetetlenné válik, így a rájuk való hivatkozás nincs megengedve.

A tervezés során ez előnyös, mert nem engedi meg a tervezőnek, hogy olyan jelre vagy jelekre hivatkozzon, amelyeknek az értéke nem egyértelműen definiált.

Az alábbiakban az eseményekre és a működésekre vonatkozólag megadjuk az SN7474-es D tároló és az SN7483-as összeadó funkcionális leírását.

```
TYPE: DFLOP (D,CP,CL,PR,Q,NQ; SN7474).
INPUTS: D,CP,CL,PR.
OUTPUTS: Q,NQ.
INPUT_EVENTS:
  CPLH: CP CHANGES_TO 1 BIN;
  CLHL: CL CHANGES_TO Ø BIN;
  PRHL: PR CHANGES_TO Ø BIN.
OUTPUT_EVENTS: SET:
  AT CPLH+33 NSEC,IF D STEADY
  FROM CPLH−2Ø NSEC TO CPLH+5 NSEC
  AND CL=1 BIN AND PR=1 BIN,
  Q=D, NQ=NOT D;
CLEAR: AT CLHL+33 NSEC,
Q=Ø BIN,NQ=1 BIN;
PRESET: AT PRHL+33 NSEC,
Q=1 BIN, NQ=Ø BIN.
DFLOP END.
```

Bemeneti eseményként a CP órajel felfutását, a CL törlő és a PR beíró bemenetek lefutását definiáltuk.

A SET kimeneti esemény, mely beállítja a Q és NQ kimeneteket a D bemeneteknek megfelelően a CPLH bemeneti esemény bekövetkezését 33 ns-mal követőleg történik meg. A SET kimeneti esemény bekövetkezésének az a feltétele, hogy a D bemeneten levő jelérték állandó legyen a beállási és tartási idő alatt, valamint az aszinkron beíró és törlő bemenetek értéke logikai egyes szintű legyen. A CLEAR és a PRESET kimeneti eseményeknek nincs feltételrésze, az események a CL vagy PR bemenetek $1 \rightarrow 0$ átmenetekor megtörténnek 33 ns késleltetési idővel.

Az SN7483-as négybites bináris összeadó leírása:

```
TYPE: SUM4 (A,B,CI,CO,S; SN7483).
INPUTS: A,B 4 BITS, 4 BIN DIGITS; CL
OUTPUTS: S 4 BITS; CO.
INPUT_EVENTS: CHG:
  ANY_INPUTS CHANGE.
OPERATIONS: SUM: STARTS_AT CHG
TERMINATES_AT CHG+24;
RESULT: S=A+B+CI.
SUM END.
CRY: STARTS_AT CHG
TERMINATES_AT CHG+16;
RESULT: IF(A+B+CI)>=10ØØØ BIN
        THEN CO=1 BIN;
        IF(A+B+CI)<10000 BIN
        THEN CO=0 BIN.
CRY END.
OPERATIONS_END.
SUM4 END.
```

A típus bemeneti jelei bármelyikének megváltozása hatással van a kimenetre, így bemeneti eseményként az ANY_INPUTS CHANGE változást definiáljuk. A bemeneti esemény hatására a kimeneten megjelenik a bemeneti jelek összege és beállítódik az átvitel jel értéke.

Kimeneti események esetén a kimeneti jelek értéke mindig meghatározott, a jelváltás időpontjáig a régi, azt követőleg az új értékkel szerepel a jel. A felfutási vagy lefutási idő kezelésére, amennyiben az alkalmazó igényt tart rá, a leíró nyelv az X jelértéket biztosítja.

A működések azon kimeneti jelek kezelésére különösen előnyösek, amelyekre nem lehet egyértelműen meghatározni, hogy mikor történik a jelváltás, hanem csak egy intervallum definiálható, amelyen belül a jelváltás megtörténik. Ilyenkor az adott intervallumon belül a jelre nem szabad hivatkozni.

A struktúra specifikációja

A tervezendő egység funkcionális specifikációját követőleg, fel kell építeni az egységet mint alacsonyabb szintű funkcionális egységekből álló struktúrát. A struktúrát felépítő részegységeket hasonlóképpen le kell írni, mint funkcionális specifikációit és a külső kapcsolataik összeköttetése által épül fel a struktúra.

A strukturális leírás elemei:

- bemenetek;
- kimenetek;
- a struktúrát alkotó építőelemek;
- a struktúra elemeinek kapcsolatát definiáló kötési lista.

A struktúrát megvalósító elemek felépítése:

```
ELEMENTS: típusnév 1: realizáció 1,
                      realizáció 2,
                      .
                      .
                      .
                      realizáció i;
          típusnév 2: realizáció 1,
                      realizáció 2,
                      .
                      .
                      .
                      realizáció j;
          típusnév n: realizáció 1,
                      .
                      .
                      .
                      realizáció k.
```

A fenti szerkezetben a típusnév a funkcionális egységet azonosítja, a realizációnevek pedig a megvalósulásokat, másnéven példányokat azonosítják.

A realizáció megadására azonosító szolgál, melyet paraméterlista követ. A paraméterlistán a típus definíciójakor megadott paraméterlistabeli felsorolási rendnek megfelelően meg lehet határozni összeköttetéseket. Ha az adott funkcionális egység valamely bemenete vagy kimenete a struktúra külső csatlakozási pontjával össze van kötve, akkor a külső csatlakozási pontot a paraméterlistán szerepeltetve, az összeköttetés definiálva lesz. A paraméterlistán szerepelhet:

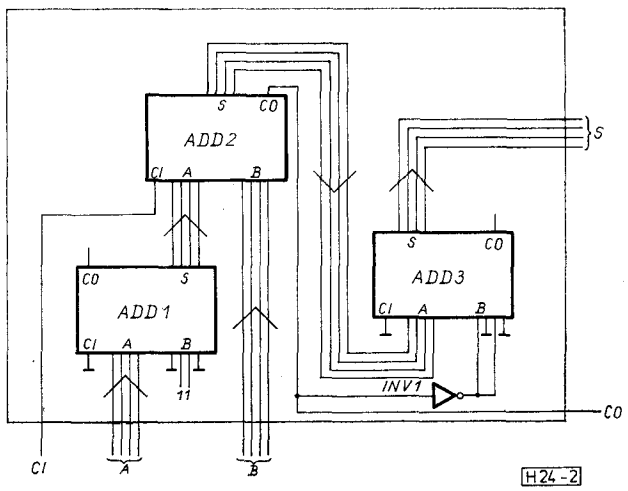
— jelnév;
— konstans;
— *karakter.

A paraméterlistán szereplő jelnév az adott típus paraméterlistájának megfelelő pozíciójában szereplő jelnek az adott struktúrabeli jelnévvel való összeköttetését határozza meg. Ha a paraméterlistán konstans érték szerepel, akkor az adott típus paraméterlistájának megfelelő pozíciójában szereplő jelnek a konstans értékre való csatlakoztatását jelenti.

Ha a paraméterlistán csillagkarakter szerepel, akkor az adott típus megfelelő pozíciójában szereplő jelnév csatlakozási pontját a kötési listán definiáljuk.

Adott realizáció be- vagy kimenetére úgy hivatkozhatunk, hogy a realizáció nevét @ karakterrel kötjük össze a szóban forgó jel azonosítójával.

A funkcionális specifikáció és a strukturális leírás együttes alkalmazására bemutatjuk egy egydekádos BCD kódú összeadó leírását. Az összeadót 4 bites bináris összeadókból építjük fel (2. ábra).



2. ábra. BCD kódú összeadó strukturális felépítése

A megoldásnál használjuk fel a négy bites bináris összeadó (SUM4) leírását.

Külső funkcionális specifikáció:

```
TYPE: BCDF (A,B,CI,S,CO; BCDFADD).
INPUTS: A,B 4 BITS, 1 DEC DIGIT; CL
OUTPUTS: S 4 BITS, 1 DEC DIGIT; CO.
INPUT EVENTS: INCHG: ANY—INPUTS
CHANGE.
OPERATIONS:
```

```
  SUM: STARTS_AT CHG
       TERMINATES_AT CHG+7Ø;
       RESULT: S=A+B+CI.
  SUM: END.
  CARRY: STARTS_AT CHG
  TERMINATES AT CHG+7Ø;
  RESULT:
  IF(A+B+CI)>9 THEN CO=1 BIN;
  IF(A+B+CI)<=9 THEN CO=1 BIN.
  CARRY END.
OPERATIONS END.
BCDF END.
```

A struktúra leírása:

```
MODEL: BCDF.
INPUTS: A,B 4 BITS, 1 DEC DIGIT; CL
OUTPUTS: S 4 BITS, 1 DEC DIGIT; CO.
ELEMENTS:
  SUM: ADD1 (A,6 DEC, Ø BIN, *,*),
       ADD2 (*,B,CI,*,*),
       ADD3 (*,*, Ø BIN,*,S);
  INV: INV1 (*,*).
CONNECTIONS:
  ADD2@S—ADD3@A,
  ADD2@CO—INV1@A—CO,
  INV1@Y—ADD3@B(1)—ADD3@B(3),
  ADD3@B(0)—ADD3@B(2)—ØBIN.
BCDF END.
```

A specifikáció leírásának és ellenőrzésének számítógépes támogatása

A specifikáció megadása a leíró nyelv segítségével történik. A specifikáció leíró nyelve a CARS (Computer Aid for Recoursive Synthesis). A nyelvhez rendelkezésre áll fordítóprogram SIEMENS és IBM számítógépekre.

A leírások szimulációjára szimulátorprogram, a szimulációs eredmények kiíratására output program, a funkcionális és strukturális leírás eredményeinek összevetésére ekvivalenciaprogram áll rendelkezésre [6].

Bemenő jelsorozatok definiálása

A típus vagy struktúra bemenő jeleinek megváltozása idéz elő bemeneti eseményeket, azaz bemeneti változásokat.

A bemeneti jelek értékváltásait a következő formában lehet definiálni:

⟨időpont megadása⟩:⟨értékdefiníciók⟩;

⟨értékdefiníciók⟩::=⟨értékdefiníció⟩

|⟨értékdefiníciók⟩, ⟨értékdefiníció⟩

⟨értékdefiníció⟩::=

=⟨azonosító⟩=**X**|⟨azonosító⟩=⟨érték⟩

azaz egy adott időpontban tetszőleges számú jel értéke **X** értékűvé vagy valamely számértékkel definiált értékűvé definiálható. A jel-érték összerende-

léseket egymástól vesszővel kell elválasztani. A megadás végét ; jel jelzi.

A működtető jelek sorozatának végét a leállítási időponttal adhatjuk meg a következőképpen:

STOP_MODEL_TIME=⟨időpont megadása⟩;

Az időpont megadása:

⟨időpont megadása⟩::=

=⟨decimális szám⟩ ⟨időegység⟩

⟨időegység⟩::=SEC|MSEC|USEC|NSEC|PSEC

A jel értékének megadása:

⟨érték⟩::=⟨szám⟩|⟨szám⟩ ⟨radix⟩

⟨radix⟩::=DEC|OCT|HEX|BIN

A leírások szimulációja, az eredmények kiíratása, a szimuláció eredményeinek összevetése

A leírások szimulációja időhelyes dinamikus szimulátorral történik. Az eredmények táblázatosan vagy idődiagram formájában jeleníthetők meg.

A táblázatos formánál kiírásra kerülnek azon időpontok, amikor valamilyen jelváltozás történik. Kiírásra kerül a jelváltozást okozó esemény neve, előfordulási száma (hányadszor következett be), valamint a megváltozott jel értéke.

Az idődiagramos forma ettől annyiban tér el, hogy a jelváltozást okozó esemény neve nem kerül kiíratásra.

A típus és a struktúra szimulációjára kapott eredményeket gépi úton összevethetjük. Az összehasonlító program azokat az idő intervallumokat jelzi ki, amelyekben a szóban forgó jelek nem ekvivalensek.

Működtető jelsorozatok, tesztelés

A konstruktőr a tervezési folyamat minden szintjén, amikor elkészít egy összetartozó külső specifikációt és strukturális leírást, elkészíti a működtető jelsorozatot. A bemeneti jelszekvenciát úgy kell összeállítani, hogy a kapott eredményekből eldönthető legyen biztosan a helyes működés. Az így specifikált bemeneti jelszekvenciák egyben a funkció tesztelésére is szolgálnak a tervezés során.

A tervezési folyamat során az összetartozó külső specifikációt és strukturális leírást a konstruktőr ugyanazzal a bemeneti jelszekvenciával működteti. A bemeneti jelszekvenciákra, mint funkcionális tesztsorozatra, kapott kimeneti jelszekvenciákat verifikálja. Helyes eredmények esetén elkészíti a következő szint leírását és bemenő jelszekvenciáit. Ezek a lépések addig folytatódnak míg megvalósított funkcionális részegységekig vagy katalóguselemekig jut a tervező.

Összefoglalás

A cikk áttekintette a digitális berendezések többszintű hierarchikus tervezésének támogatására szolgáló hardware leíró nyelvvel szemben támasztott követelményeket. Ismerteti a cikk a CARS tervező rendszer hardware leíró nyelvének elemeit és áttekinti a rendszer szolgáltatásait.

IRODALOM

[1] *Csopaki Gy.:* Hardware description language for designing of digital equipment. Preprints. Proceeding of IMACS European Simulation Meeting on Simulation in Research and Development. Eger, Hungary, 27—30 August, 1984. pp. 151—161.

[2] *Bohus M.—Csopaki Gy.—Filp A.—Hinsenkamp A.—Máté L.:* Computer Aid for Recoursive Synthesis, Working Paper, Computer and Automation Institute, Hungarian Academy of Sciences, Apr. 1982.

[3] *Bohus M.—Csopaki Gy.—Filp A.:* Számítógépek és részegységeik szimulációja. Kutatási jelentés az SZKI számára. Budapest, 1979.

[4] *Bohus M.—Csopaki Gy.—Filp A.—Hinsenkamp A.—Máté L.:* Digitális berendezések szintézisének számítógépes támogatása. Híradástechnika. XXXII. évf. 1981. 1. szám. pp. 8—12.

[5] *Máté L.—Bohus M.—Csopaki Gy.—Filp A.—Hinsenkamp A.:* System CARS and its Description Language. Lecture Notes in Computer Sciences. 152. Specification and Design of Software Systems. Springer-Verlag. Berlin, Heidelberg, 1983.

[6] Digitális rendszerek többszintes, időhelyes szimulációjára szolgáló programrendszer felhasználói kézikönyve. Budapest, 1983.