BOHUS MIKLÓS
BME Vezetéknélküli Híradástechnikai Tanszék

Szimbolikus nyelvek felhasználása digitális rendszerek funkcionális és parametrikus szimuláeiójára*

ETO 518.5:681.32:800.92

Valamely digitális rendszer formális nyelven történő leírásához meg kell adni (deklarálni kell) a rendszer felépítését. Ezért deklarálni kell:

a) a rendszert felépítő automatákat, ezek állapotait, az állapotok összetartozó csoportjait (szegmenseket),

b) az automatákat felépítő funkcionális egységeket (regiszterek, tárolók stb.), műveletvégző egységeket (kombinációs hálózatok, kombinációs hálózatcsoportok),

c) időzítést, késleltetést biztosító hálózatokat,

d) a rendszer bemenő és kimenő kapcsait (kapocspárok csoportját, sínrendszert).

Az előzetesen értelmezett (deklarált) egységek (automaták, funkcionális egységek stb.) be- és kimenő kapcsai (mint operandusok) között végrehajtható műveleteket kell definiálni. A műveletek határozzák meg az egységek egymásra kifejtett hatását, azok összekapcsolódását, az állapotváltozásokat stb. A műveletek értelmezése során meg kell adni, hogy

a) mik lehetnek egy művelet operandusai (kombinációs hálózat, vagy funkcionális egység, vagy automata stb. ki- és bemenetei),

b) a művelet feltételes-e, vagyis végrehajtását feltétel teljesülése előzi-e meg,

c) a művelet időzítését mi és hogyan határozza meg,

d) egyidejűleg más művelettel együtt is végrehajtható-e (párhuzamos műveletvégzés).

A deklarálható egységek típusa és száma, a végrehajtható műveletek jellege és száma az egyes nyelvekben lényegesen eltér egymástól.

A műveletek értelmezésén kívül megadandó a rendszer vezérlésének struktúrája.

A fentiek alapján szimulálható a rendszer logikai, funkcionális működése. Pontosabb leíráshoz az időviszonyokat is meg kell adni, ami az időzítések kezelését teszi szükségessé.

A nyelvekkel szemben általában eltérő követelményeket támasztunk:

1. Legyen független a rendszer fizikai felépítésétől (pl. rendszerben levő tárolók típusa, kapacitása, működési sebessége; az áramköri készlet típusa — MSI, LSI elemek stb.)

2. Legyen független a rendszer szervezésétől (időosztás, párhuzamos műveletvégzés stb.).

3. Legyen alkalmas különböző mélységű szimulációra, vagyis a rendszer bizonyos részét akár építőelem szinten szimulálja, a többi résznek csak a működés szempontjából érdekes funkcionális működését írja le.

4. Tegye lehetővé a hierarchikus leírást.

5. Legyen alkalmas parametrikus modellezésre, vagyis a funkcionális leíráson túl a rendszer működése szempontjából érdekes paraméterek (pl. időzítések) meghatározására. Ezt a tulajdonságot a modellezési nyelv teljesítőképességével jellemzik. Annál nagyobb teljesítőképességű a nyelv, minél több tulajdonságát képes leírni a modellezendő objektumnak.

6. A nyelven leírt objektum automatikus tervezésére is legyen felhasználható. Tegye lehetővé az egységes tervezést oly módon, hogy a formális nyelven leírt programból számítógépes eljárással

6.1 meghatározható legyen a rendszer logikai struktúrája (logikai szintézis) ennek segítségével,

6.2 konstrukciós
6.3 technológiai
6.4 üzemeltetési
} tervezésre legyen felhasználható

A cikkben rövid áttekintést adunk a deklarálható egységek kiválasztásáról, a megengedett műveletekről, a vezérlések struktúrájáról és a parametrikus modellezés legfontosabb kérdéséről az időzítések kezeléséről.

A legismertebb formális nyelvek felépítésének, jellemzőinek még vázlatos ismertetése is meghaladja egy cikk kereteit, ezeket több tanulmány tartalmazza. [41-ben] összefoglalóan ismertettük a DDL, LOTIS, IVERSON nyelvet és az IVERSON nyelv alkalmazási lehetőségeit. Az OSZSZ-2 formális nyelv részletes leírása 1969-ben készült el [20]. Az OSZSZ-2 nyelv alkalmazási lehetőségeit és továbbfejlesztésére vonatkozó javaslatokat [22] tartalmazza. A további nyelvekről [1—41] hivatkozásokban további irodalmi utalások találhatók. Az OSZSZ-2 nyelv egy egyszerűsített változatának (SUBSET) felépítéséről, utasításairól, konkrét megvalósításáról [21] számol be.

I. Egységdeklarációk

1. Alapdeklarációk

Valamennyi formális nyelv változói (operandusai) között megtalálhatók a be-, ill. kimeneti kapcsok csoportjának (terminal), regisztereknek, tárolóknak

* A munka a Számítástechnikai Koordinációs Intézet megbízásából készült.

Beérkezett: 1972. III. 4.

a deklarációja. Ezeket alapdeklarációknak tekinthetjük. Ha ezeket az egységeket egymástól meg kívánjuk különböztetni, akkor a deklaráció típusának megjelölésére külön szimbólum vezetendő be (pl. TERMINAL, REGISTER, MEMORY). A deklaráció típusa az áttekinthetőség rovására el is hagyható, mivel minden egységhez egyedi azonosító is rendelhető.

A deklaráció típusa után helyezkedik el a deklaráció törzse. Ez adott számú egység (be-, ill. kimeneti csatlakozócsoport, regiszter, tároló) megadását teszi lehetővé. Egyszerűbb nyelvekben egy deklarációban egyetlen egység adható meg (pl. egyetlen regiszter).

Minden egység önmagában „n" dimenziós lehet. (A nyelvek egy részében a dimenziószám egy vagy kettő). Az egység egyes elemeit pontosan meg kell határozni. A meghatározás a dimenziószámhoz tartozó index és a pozíciószám megadásával történik. A pozíciószám meghatározása az egyes pozíciók

a) alsó és felső határát vagy egyszerűbb esetben

b) a pozíciók számát

adja meg.

Mind a pozíciók alsó, illetve felső határát, mind a pozíciók számát egy másik egység (pl. regiszter) információtartamának értékével is meghatározhatjuk, vagy előre megadjuk.

Az így deklarált egységekre a hivatkozás történhet:

a) az egység egyedi azonosítójának megadásával, ha a teljes egységre (regiszter, tároló stb.) hivatkozunk,

b) az egység azonosítójának és pozíciószámának megadásával, ha az egység adott helyértékére hivatkozunk.

A fentiek alapján pl., ha a dimenziószám 2, a dimenziószámhoz tartozó index 8, a pozíciószám 32; a deklaráció törzsében levő azonosítók száma 4, a deklaráció típusa regiszter, akkor egyetlen deklarációval 4 darab regisztercsoport deklarálható. Minden regisztercsoportban 8 regiszter van, a regiszterek szóhossza 32 bit.

A hivatkozás bármely regiszterre, annak bármely elemére vagy a teljes egységre történhet. A műveletek egy része a teljes egységgel vagy annak előírt részeivel végezhetők.

Ha a nyelv egydimenziós egységek deklarálását teszi lehetővé, akkor az egységekhez egy-egy vektor rendelhető hozzá, és a vektor elemeinek kijelölése a pozíciók megadásával történik (alsó és felső határ vagy egyszerűbb esetben a pozíciók száma előre elfogadott pozícióhelyzet értelmezés alapján).

Az egyes deklarációkat határolók választják el egymástól. A deklarációkat lezáró határolók után megjegyzések helyezhetők el.

Az alapdeklarációkban minden nyelvben megtalálható az egyesítési operátor, mely a külön-külön értelmezett egységek közös egységként történő értelmezését teszi lehetővé. (Legegyszerűbb esetben pl. egydimenziós egységvektor-elemek összefogása egyetlen vektorba.)

Az alapdeklarációkban szereplő mennyiségek állandók vagy változók lehetnek.

2. Kombinációs hálózatok deklarálása

Be-, ill. kimeneti csatlakozó kapcsok, vagy a hálózat tetszőleges pontjai között levő kombinációs logikai hálózat leírása általában logikai egyenlet segítségével történik (Boole-deklaráció).

A deklaráció típusára külön szimbólum vezethető be (pl. BOOLE), vagy ez esetben is lehetőség van a típus megadásának elhagyására.

A deklaráció törzse tetszőleges vagy adott számú logikai egyenletet tartalmaz (egyszerűbb esetben csak egy-egy logikai egyenlet adható meg deklarációként).

A logikai egyenlet változói általában az alapdeklarációban szereplő változók (regiszterek, csatlakozó kapcsok, tárolók). A hivatkozás ennek megfelelően az alapdeklaráció azonosítóival történik. Ezzel elkerülhető, hogy a kombinációs hálózat deklarálásakor a dimenziókat külön meg kelljen adni (ez az alapdeklarációkban szerepel).

A logikai egyenletben előírt logikai funkció értelmezése összetartozó helyértékenként történik.

Így, ha az alapdeklarációban szereplő változók dimenziója 1, a deklaráció törzsében levő azonosítók száma 4, a pozíciószám 32, 4 egyenlettel 4×32 logikai egyenlet (hálózat) írható le.

A logikai hálózat kimenete és bemenete oly módon különböztethető meg, hogy a logikai egyenlet egyik oldalához (pl. bal oldalához) rendelhető a kimenet.

A Boole-deklaráció segítségével értékadás végezhető, ha a logikai egyenlet kimenetéhez tartozó oldal megegyezik az adott változóval. Hasonlóan lehet az átvitelt, a különböző típusú feltételes (kapuzott) átvitelt is felírni szimbolikusan.

A kombinációs hálózat kimenetére történő hivatkozás a logikai egyenlet kimenetéhez tartozó változóval történik.

Gyakran előfordul, hogy ugyanazt a kombinációs logikai áramkört különböző időpontban más és más változók (regiszterek, csatlakozó pontok stb.) veszik igénybe. Ily módon ez időosztásos üzemben is igénybe vehető. Ekkor a logikai hálózat deklarálásában is utalni kell erre a lehetőségre. A deklaráció tipusa tehát eltér az egyszerű Boole-deklarációétól (pl. OPERATOR) és a deklaráció törzsében elhelyezendők azok az azonosítók, melyek értékét egy következő kifejezés határoz meg. Egy deklarációban, ebben az esetben is tetszőleges vagy adott számú időosztással igénybe vehető hálózat írható le. A deklaráció törzse logikai egyenlet. Az utasítások sorrendje határozza meg azt, hogy a kombinációs hálózat mikor milyen kapocspárokhoz csatlakozik.

3. Jelkésleltetés, órajelek deklarálása

A deklaráció típusa (pl. DELAY) után a deklaráció törzse adott számú egyedi azonosítóval megkülönböztetett késleltető elemet tartalmaz, mindegyikhez tetszőleges, előre megadott értékű vagy változóval beállítható késleltetési idő rendelhető.

A késleltető elemre történő hivatkozás az egyedi azonosítók megadásával történik.

A késleltető elemhez hasonló módon deklarálhatók a rendszerben lévő periodikus órajelet adó óragene-

rátorok. Az óragenerátorokban szereplő idő a periódusidő lehet. Az órajelek kezdő fázisára a nyelvek általában nem tartalmaznak előírást.

Az óragenerátorokra történő hivatkozás a generátorok egyedi azonosítóinak megadásával történhet.

4. Funkcionális egység deklarálása

Önálló funkció realizálására szolgáló egységekre, a funkció belső szerkezetének részletes megadása nélkül, mint egyetlen önálló elemre gyakran kell hivatkozni. Ilyen funkcionális egységként bármely elem deklarálható, ha azt a fordítóprogram értelmezni tudja.

A legtöbb nyelv lehetővé teszi az operatív tárra történő hivatkozást az üzemmód, vezérlőjelek, címregiszter, tartalomregiszter, beírandó, ill. kiolvasandó szavak (byte-ok) száma stb. megadásával.

A deklaráció típusa fejezi ki azt, hogy önálló funkcionális egységről van szó, törzse tartalmazza az egység(ek) egyedi azonosítóit a bemenő-kimenő változók, vezérlőjelek megadásával.

Önálló funkcionális egység lehet a dekódoló, kódoló, vektorból mátrix, mátrixból vektor generálására szolgáló egység stb.

5. Véges automata deklaráció

Az önálló vezérlő- és funkcionális egységekkel rendelkező logikai hálózatok véges automataként deklarálhatók. A deklaráció típusa (pl. AUTOMATION) fejezi ki, hogy önálló automata deklarációjáról van szó. Törzse tartalmazza a benne szereplő funkcionális egységeket, az automata állapotait stb.

6. Rendszerdeklaráció

A rendszerdeklaráció véges automatákból, esetleg önálló funkcionális egységből (egységekből) álló logikai hálózat deklarálására szolgál. A deklaráció típusa (pl. SYSTEM) fejezi ki, hogy a deklaráció rendszerre vonatkozik, törzse tartalmazza a benne levő véges automaták, funkcionális egységek azonosítóit.

Az általában deklarálható egységeket és megadásukhoz szükséges legfontosabb jellemzőket összefoglalásképpen az 1. táblázat tartalmazza.

II. Állapotdeklaráció

A rendszer állapotát az állapotregiszter(ek) tartalma határozza általában meg (ez lehet pl. az utasításregiszter). Ezért célszerű az állapotazonosító bevezetése.

A rendszer előírt állapotba hozható az állapotregiszter megfelelő beállításával. Az állapotok egy meghatározott csoportja állapotszegmensbe foglalható össze.

Az állapotdeklaráció során a típus jelzi, hogy egy rendszer állapotainak megadásáról van szó, a deklaráció törzse tartalmazza az értelmezett állapotok azonosítóit (állapotlista).

Az állapotok közötti átmenetet pl. az állapotregiszter vezérelheti. Az állapotok, állapotcsoportok közötti átmenetek, visszatérések, várakozásos helyzetek stb. előírására önálló szimbólumok vezetendők be. Néhány jellegzetes állapotot a 2. táblázat tartalmaz.

1. táblázat

Deklarálható egységek

Egység	Megadás módja
Regiszter Csatlakozó kapcsok Tároló	Alapdeklaráció Megadandó: dimenzió, index pozíció hossz
Kombinációs hálózat Értékadás Átvitel (kapuzott adatátvitel)	Adott kapocspárok között állandó összeköttetés lehet, vagy időosztással tetszőleges kapocspárokhoz csatlakozhat Megadandó: Boole-egyenlet
Jelkésleltetés Periodikus órajelek	Idődeklaráció Megadandó adat: idő
Funkcionális egység	Funkcionális egység típusának, kimenő-bemenő kapcsoknak, vezérlőjeleknek stb. felsorolásával
Véges automata	A benne szereplő funkcionális egységeknek, az automata állapotainak stb. megadásával
Rendszer	A rendszert megvalósító automaták és funkcionális egységek megadásával

2. táblázat

Rendszer állapot

Állapot	Megadás módja
Előírt	Az állapot regiszter tartalmával
Várakozásos	Adott feltétel teljesüléséig az adott állapot, ezután előírt állapot megadásával
Adott állapot csoporthoz tartozó	Állapot szegmens

3. táblázat

Speciális jelsorozatok

Név	Értelmezés
Egységvektor	Az adott szóhosszú vektor valamennyi eleme 1
Null vektor	A vektor valamennyi eleme 0
Prefix vektor (j, n)	A balról számított első j elem 1, a többi 0 (Szóhossz: n)
Suffix vektor (j, n)	A jobbról számított első j elem 1, a többi 0 (Szóhossz: n)
Intervallum vektor (j, n)	A vektor a következő számokból álló jelsorozatot állítja elő: $j, j+1, \ldots (j+n-1)$
Előírt jelsorozat	Az előírt pozíciószámon (számokon) 1 van, a többi elem 0

Az állapotregiszter lehet az operatív tároló egy része is.

III. Jelsorozatok deklarálása

Rendszerszimuláció fontos eszköze a különböző bináris, decimális jelsorozatok előzetes deklarálása. Több nyelvben csak egydimenziós jelsorozatokat deklarálnak speciális vektorokként (3. táblázat).

A jelsorozat elemei lehetnek:

a) előírt determinisztikus értékek,

b) statisztikusan változó előírt hosszúságú véletlen számok.

IV. Megengedett műveletek

A deklarált egységek bemenő és kimenő kapcsai — mint operandusok — között általában előírt prioritással értelmezett műveletek végezhetők el. A műveleteket az egyszerűbb szóhasználat érdekében operátorokkal írjuk elő. A nyelvek általában az alábbi operátorokat vezetik be:

1. Aritmetikai operátorok (aritmetikai jellegű műveletek elvégzésére).

2. Relációkat kiértékelő operátorok.

3 Logikai operátorok (logikai műveletek kifejezésére).

4. Redukciós operátorok (egy egység valamennyi elemével végzett aritmetikai reláció, logikai művelet kifejezésére).

5. Bázisérték képző operátorok (pl. regiszterben levő jelsorozathoz egy adott módon értelmezett szám vagy fordítottjának előállítására).

6*a)* Regiszteroperációk (léptetés, számlálás, átvitel, maszkolás stb.)

b) Állapotregiszterre vonatkozó operátorok, más szóval vezérlésátadó operátorok.

7. Egyéb speciális operátorok (pl. aktiválás, összekötés stb.).

A műveletek határozzák meg, hogy valamely egység mikor, milyen feltétel mellett, milyen hatást gyakorol egy vagy több másik egységre.

Legtöbb nyelv különböző módon ugyan, de bevezeti a feltételes műveleteket, így:

1. ha a művelet végrehajtását egy feltétel előzi meg, akkor a

a) feltétel teljesülésekor egyik, nem teljesülésekor másik műveletet (vagy műveletcsoportot) végzi el,

b) feltétel teljesülésekor előírt műveletet (művelet csoportot) végez, nem teljesülésekor nem történik műveletvégzés stb.

2. ha a művelet végrehajtását „k" bites feltétel előzi meg, és 2^k számú művelet végezhető, akkor a feltétel minden kombinációjához egy-egy művelet rendelhető stb.

Az egyidejűleg végzett műveletek szimulálása jelenti az egyik legnagyobb nehézséget. Legegyszerűbb esetben a nyelv egyszerűen lehetőséget ad egy műveletcsoport deklarálására. A műveletcsoport deklarációjának egyik oldalán a csoportra utaló művelettípusjel szerepel, másik oldalán az egyidejűleg végrehajtandó műveletek azonosítói. Ebben az esetben a fordító (esetleg értelmező) program maga gondoskodik az egyidejű műveletvégzésről. Ha erre a nyelv más lehetőséget nem ad, a műveletek (utasítások) megfelelő vezérléséről a programírónak magának kell gondoskodnia.

1. Aritmetikai operátorok

Az operátorok vonatkozhatnak:

a) skalárra és egyetlen operandusra (abszolút érték előállítás, előjel előállítás, egész-rész előállítás stb.),

b) skalárra és két operandusra (összeadás, kivonás, szorzás, osztás, két mennyiség közül a kisebb vagy nagyobb kiválasztása, az egyik skalár rezíduuma másik skalár által megadott moduló szerint stb.),

c) egyetlen vektorra, ha az aritmetikai műveleteket a vektor komponensei között kell elvégezni (összeadás, szorzás, a vektor elemekből a legkisebb, legnagyobb kiválasztása stb.),

d) értelmezhetők két vektor között (két vektor összeadása, kivonása, szorzása, osztása stb.) mégpedig helyértékenként, vagy a vektorok jelsorozatához hozzárendelt számokkal,

e) mátrixok között,

f) vegyes műveletek definiálhatók skalár-vektor, skalár-mátrix, vektor-mátrix stb. között.

Az aritmetikai operátorok értelmezhetők skalár, vektor, mátrix: konstansokra, illetve változókra,

A skalár, vektor, mátrix elemek bináris, oktális, decimális, hexadecimális stb. számrendszerben értelmezhetők.

2. Reláció operátorai

A relációk operátorai ($<$, $\leqq$, $=\geqq$, $>$, $\neq$) értelmezhetők:

a) aritmetikai értelemben. Ebben az esetben két skalár konstans vagy skalár változó között értelmezzük. Általában vezérlésátadásra használható. Ha a relációban előírt feltétel teljesül, akkor nem a következő utasítás hajtódik végre, hanem vezérlésátadás történik,

b) logikai értelemben, ekkor a feltétel teljesülése logkai igaz értéket állít elő. Alkalmazható két skalár konstans vagy változó között, vagy két vektor között, ekkor a művelet eredménye szintén vektor és a művelet vektor elemenként végzendő el.

3. Logikai operátorok

A nyelvek a megengedett logikai operátorokban különböznek egymástól. Szokásos logikai műveletek: TAGADÁS, ÉS, VAGY, NEMÉS, NEMVAGY, ANTIVALENCIA, EKVIVALENCIA stb. A műveleteket vektorok esetében helyértékenként kell értelmezni.

4. Redukciós operátorok

A 4. táblázatban felsorolt leggyakoribb redukciós operátorokon kívül a redukciós operátor alkalmazható:

4. táblázat

Redukciós operátorok

Művelet	Értelmezés
Aritmetikai művelet vektorra vonatkoztatva	Az aritmetikai művelet ismételt végrehajtása a vektor elemek között
Logikai művelet vektorra vonatkoztatva	A logikai művelet ismételt végrehajtása a vektor elemei között
Speciális művelet (pl. nagyobb vagy egyenlő egész) vektorra vonatkoztatva	A művelet végrehajtása a vektor elemei között (pl. legnagyobb elem egész részének előállítása)
Mátrix sorának redukciója	Az előírt aritmetikai, logikai stb. művelet alkalmazása
Mátrix oszlopának redukciója	

5. táblázat

Regiszter operációk

Művelet	Értelmezés
Léptetés (jobbra vagy balra) előírt helyértékkel	Logikai vagy aritmetikai értelemben
Ciklikus léptetés (jobbra vagy balra) előírt helyértékkel	A regiszter tartalmának ciklikus forgatása jobbra vagy balra
Számlálás (előre vagy hátra) előírt értékkel	A regiszter tartalmának növelése vagy csökkentése
Átvitel	A regiszterbe egy vektor által meghatározott jelsorozat irandó
Átvitel az állapot regiszterbe	Állapot regiszter tartalmának megváltoztatása
Maszkolás	Két vektorból egy bináris jelsorozat segítségével előírt módon egy új vektor előállítása
Egyesítés	Két regiszter egymás mellé helyezése
Leválasztás	Regiszter egy részének leválasztása speciális jelsorozat és redukciós operátor segítségével
Expanzió	Egy vektor, valamely bináris jelsorozat segítségével egy másik vektor adott elemének helyére írható
Feltételes operációk (pl. átvitel)	Egy vagy több — egybites — feltételtől függően a művelet különböző képpen hajtódik végre (pl. különböző pozíciókba íródik az információ, vagy különböző információ íródik adott helyértékekre stb.)
Időosztásos adattranszfer (pl. sínrendszer)	A rendszer adatútjainak (sínrendszernek) időosztásos használata

a) vektor és mátrix között. Ebben az esetben a vektor binárs jelsorozatot tartalmaz általában, és alkalmazható a mátrix sorára, illetve oszlopára,

b) két mátrix között.

Pontosan értelmezendő, hogy milyen operátorok állhatnak a kompressziós operátorok előtt.

5. Bázisérték-képző operátorok

A bázisérték-képző operátorok egy vektor (vagy mátrix sorának, ill. oszlopának) elemeit adott számrendszerben értelmezett számjegyeknek tekintik és ezekből általában fixvesszős számot állítanak elő, vagy e művelet fordítottját végzik el. Minden nyelv közös tulajdonsága, hogy bármely vektorhoz egy operátorral érték (szám), minden számhoz pedig vektor rendelhető. Ily módon lehet vektorból skalár értékeket előállítani.

6. Regiszter operációk

A vektorokra értelmezett műveletek közül igen nagy jelentőségük van a regiszter operációknak. Az 5. táblázat tartalmazza a leggyakrabban használt regiszter operációkat, feltüntetve azok értelmezését is.

Az átviteli operátorokkal létrehozott állapotváltozás speciális esete az állapot regiszter tartalmának megváltoztatása. Ennek segítségével történhet interrupt működés szimulálása.

7. Speciális operátorok

A nyelvek a rendszerek leírását egyéb speciális operátorokkal is segíthetik. Ezek egy részéhez nem tartozik külön hardware (pl. aktiválás). Néhány gyakorlati speciális operátor értelmezése a 6. táblázatban látható.

6. táblázat

Speciális operátorok

Művelet	Értelmezés
Aktiválás	Közös funkcionális egységekkel rendelkező automaták közül az egyik automata aktiválása hatást gyakorol egy vagy több automatára
Globális feltétel alól történő feloldás	Valamely művelet vagy műveletcsoport felszabadítása a globális feltételek alól
Másolás	Ismétlődő áramkör kezelésének módja, ha az áramkör leírása csak egyszer történik. Az áramkörök megkülönböztetésére az operátor indexe szolgál
Makro utasítás	A nyelv operátorainak segítségével előállított — külön definiált — makro operátor

V. Vezérlések struktúrája és időzítése

A szimuláció a logikai, funkcionális működésen túl a rendszer működésének időviszonyaira is kiterjedhet. Ebben az esetben az operátorok specifikálása

azok válaszidejének megadásával bővül. Így egy több operátorból álló művelet válaszidejének meghatározása:

a) az operátorok válaszidejéből (mint annak legkedvezőtlenebb értéke) számítható,

b) arra egy konstans vagy változó intervallum írható elő.

Ebben az esetben az egyidejű műveletvégrehajtás kezelése a már leírtaknak megfelelően — egymás befolyásolása nélkül — történik, de a válaszidők külön-külön (esetleg eltérő módon) számítandók.

Az operátorok specifikálásához szükséges adatokat a 7. táblázatban foglaltuk össze.

Feltételezve, hogy a művelet sorozat végrehajtása szekvenciális (monoton növekvő címkék szerint történik), az egyes műveletek megkezdésének vezérlése lehet szinkron, késleltetett szinkron, aszinkron vagy vegyes.)

Szinkron szekvenciális vezérlést a deklarált periodikus órajelgenerátorok vezérelhetik. A késleltetett szinkron vezérléshez ugyancsak a deklarált jelkésleltető egységek használhatók. Aszinkron vezérlések esetén az egyik művelet befejezése indítja a következő műveletet. Vegyes vezérlés könnyen elképzelhető, pl. valamely művelet befejezése után, adott késleltetés elteltével történő órajel megjelenése indítja a következő műveletet.

A műveletek végrehajtásának vezérlése így leírható abban az esetben, ha egyidőben csak egy művelet hajtódik végre. Sok esetben (pl. több autonom vezérlő egység, decentralizált vezérlés stb.) a műveletek sorozata egyidejűleg hajtódik végre. Ilyenkor a sorozatok csoportokra bonthatók, melyekben a végrehajtás egyidejű. A csoporton belül történő vezérlés lehet szinkron, aszinkron, vegyes.

A vezérlés strukturájának leírásához ismerni kell azokat a lehetőségeket, amiket a vezérlés lehetővé tesz (8. táblázat). Egy csoporton belül a vezérlés sorrendjének megváltoztatása általában elágazással történik. Csoportos műveletvégzés esetén aktiválással, tiltással vagy behívással lehet a vezérléseket változtatni. Ezek a lehetőségek időosztással, többszörös műveletvégzéssel rendelkező rendszerek leírásához szükségesek.

A vezérlés struktúrájának meghatározása után történhet a vezérlések időzítésének (pl. tiltási időpont) megadása.

7. táblázat

Operátor megadás

Művelet specifikálása	Művelet válaszideje	Művelet végrehajtásának időpontja
a) Feltétel nélküli	*a)* állandó	*a)* szinkron vezérelt
b) Feltételes	*b)* változó	*b)* szinkron vezérelt állandó vagy változó késleltetéssel
c) Több művelettel egyidejűleg végrehajtható		*c)* asszinkron vezérelt
		d) változó vezérlésű

8. táblázat

Vezérlési módok

Művelet végzés	Művelet módosítás módja	
Egyidőben egyszeres	Elágazás *a)* Direkt elágazás (előre adott lépésre) *b)* Indirekt elágazás (az ugrás helye előre nem ismert)	
Csoportos	Aktiválás (valamely művelet más csoportban egy vagy több sorozatot aktivál)	
	Tiltás (valamely művelet egy másik csoportban egy vagy több műveletet letilt)	
	Behívás (adott csoportba egy másik csoportban leíró művelet vagy eljárás behívása) Típusai:	
	csoport neve	eljárás neve
	ismert	ismert
	ismert	nem ismert
	nem ismert	nem ismert

IRODALOM

[1] *J. J. Kelly Jr.—C. Lochbaum—V. A. Vyssotsky:* A block-diagram compiler. Bell Sys. Techn. *J.* vol. 40. No. 3 May 1961.

[2] *G. Gordon:* A general purpose systems simulation program. IBM System *J.* vol. 1. Sept. 1962.

[3] *K. E. Iverson:* A Programming Language. John Wiley and Sons, New York. 1962.

[4] *H. M. Markowitz,—B. Hausner,—H. W. Karr:* Simscript: A Simulation Programming Language The RAND Corporation. RM—3310. November 1962.

[5] *K. Young:* A User's Experience wiht Three Simulation Languages (GPSS, SIMSCRIPT, SIMPAX). Santa Monica, California System Development Corp. TM—1755 /000/00. 1963.

[6] *D. E. Knuth—J. L. McNeley:* SOL- A Symbolic Language for General-Purpose Systems Simulation. IEEE Trans. Electr. Comp. vol. EC—13 Aug. 1964.

[7] *R. M. Proctor:* A logic design transistor experiment demonstrating relationships of language to systems and logic design. IEEE Trans. Electronic Computers vol. EC—13 pp. 422—430. Aug. 1964.

[8] *H. P. Schlaeppi:* A formai language for deseriting machine logic, timing and sequencing (LOTIS). IEEE Trans. Electronic Computers vol. EC—13 1964.

[9] *A. D. Falkoff,—K. E. Iverson,—E. Sussenguth:* A Formai Description of System/360 IBM System J. vol. 3 No 3. 1964.

[10] *H. Schorr:* Computer-aided digital systems design and analysis using a register transfer language. IEEE Trans. Electronic Computers vol. EC—13 pp. 730—737 *Dec.* 1964.

[11] *G M. Weinberg:* "PL/1 Programming Primer. McGraw Hill New York, 1966.

[12] *J. R. Dumey—D. L. Dietmeyer:* A digital system design language (*DDL*). IEEE Trans. Computers vol C—17. pp. 850—861. September 1968.

[13] *N. A. Matjuhin:* Primenenije vücsiszlityelnüh masin dija proektirovanija cifrovüh usztvojsztv. Szovjetszkoje Radio, 1968,

[14] *N. Chapin:* 360 Programming in Assembly Language. McGraw-Hill, New York, 1968.
[15] General Purpose Systems Simulator III. — Introduction IBM Corporation Manual No B20—0001—0.
— User's Manual IBM Corporation Manual No H20—0163—1.
[16] *J. R. Duley*—D. L. *Dietmeyer:* Translation of a DDL Digital System Specification to Boolean Equations. IEEE Trans. Computers vol. C—18, 1969.
[17] *M.* L. *Dertovzos*—*M. E. Kaliski*—*K.* P. *Polzen:* On-line Simulation of Block-Diagram Systems. IEEE Trans. Computers, vol. C—18, 1969.
[18] *T.* D. *Friedman*—*S. C. Yang:* Methods Used in an Automatic Logic Design Generator (ALERT). IEEE Trans. Computers, vol C—18 jul. 1969.
[19] *G. G. Hays:* Computer Aided Design: Simulation of Digital Design Logic. IEEE Trans. Computers. vol. C—18. jan. 1969.
[20] *Jazük* dija opiszanija sztrukturnüh algoritmov i szbem (OCC—2). Otraszlevoj sztandart: OCT 4. 10.000.025 Moszkva, 1969.
[21] *Bohus M.*—Dr. *Németh G.*—*Trón T.*—*Varró* L. SUBSET szimulációs nyelv digitális rendszerek funkcionális vizsgálatára. Híradástechnika XIII. évf. 6. sz.
[22] *Bohus M.*—*Németh G.*—*Trón T.*—*Varró* L.: Strukturális algoritmusok szimulációja (OSzSz—2 SUBSET). Számítástechnikai Koordinációs Intézet Budapest, 1970.
[23] *V. M. Glutkoo:* Perszpektivü avtomatizacii proektirovanija vücsiszlityelnüh masin. Vesztnik CCCP No 4 1967.
[24] *H. Herscovitch*—*I. H. Schneider:* GPSS III — an Expanded General Purpose Simulator. IBM Systems Journak vol. 4 No 3. 1965.
[25] *H. M. Markowitz, B. Hansner:* Simscript — a Simulation Programming Language. Prentice-Hall. New York, 1963.
[26] D. *E. Knuth, J. L. McNeley:* A Formai Definition of SOL. IEEE Trans. on E. C. vol. V. EC—13 Aug. 1964.
[27] *Ole- Jahon-Dahl, Kristen Nygaard:* SIMULA — a Language for Programming and Description of Discrete Event Systems. Norwegian Computing Center, maj. 1966.
[28] L. *A. Kalinicsenko:* Formalnoje opiszanije jazüka SZLENG, v. szb „Teorija arbomatov" vüp. 1. izd. I. K. A. N. USzSzR Kiev, 1967.
[29] L. *A. Kalicsenko:* SZLENG — ekszperimentalnüj jazük programirovanija, orientirovannüj na opiszanije i modelirovanije vücsiszlityelnüh masin i szisztem. V. cb. „Teorija avtomatov" vüp. 1 izd. I. K. A. N. USzSzR 1967.
[30] *V. M. Gluskov*,—L. *A. Kalinicsenko*,—*G.* P. *Marjanovics*,—*V. M. Moszkalenko*,—*M. A. Szahnjuk:* SZLENG — Szisztema programmorovanija dija modelirovanija diszkretnüh szisztem, Kiev, 1969.
[31] -APL/ 360 Primer Student Text. IBM Technical Publications Department. New York, 1969.
[32] *O. J. Dahl*—*K. Nygaard:* The Simula Language Norwegian Computing. Centre. Oslo, 1964.
[33] Time-Sharing UNIVAC 1108.
— I—IE FORTRAN conversationnel.
— II—IE langage BASIC.
[34] Analysis of Some Queuing. Models in Real-Time Systems. IBM Manuel. Number F20—0007—0. New York, 1965.
[35] *James Martin:* Programming Real-Time Computer Systems. Prentice-Hall. London, 1965.
[36] *James Martin:* Design of Real-Time Computer Systems. Prentice-Hall. London, 1967.
[37] *A. I. Pressman:* Design of Transistorized. Curcuits for Digital Computers. John F. Rider. New York, 1959.
[38] *M. V. Wilkes:* Time-Sharing Computer Systems. MacDonald, New York, 1968.
[39] *I. Flores:* Computer Organization. Prentice-Hall, New York, 1969.
[40] Simpac user's Manual, Santa Monica, California: Systems Development Corporation, TM602/000/00 April 15, 1962.
[41] *Bohus M.*—*Flesch I.*—*Géher K.*—*Németh* G.—*Pápay* Zs.—*Szittya O.*—*Theisz* P.: Logikai rendszerek számítógépes szimulációja. Tanulmány a Számítástechnikai Koordinációs Intézet számára, Budapest, 1969.