BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
IRÁNYÍTÁSTECHNIKA ÉS INFORMATIKA TANSZÉK

Pilászy György

Új algoritmusok a magas szintű szintézis módszertanának kiterjesztésére elosztott rendszerek tervezéséhez és optimalizálásához

Doktori (PhD) értekezés tézisfüzete

Témavezető: Dr. Arató Péter
Konzulens: Dr. Móczár Géza

A disszertáció és a tézisfüzet elektronikus formában elérhető az alábbi tárhelyen:

http://directory.iit.bme.hu/~gpilaszy/phd/

Budapest, 2013

Bevezetés, célkitűzések

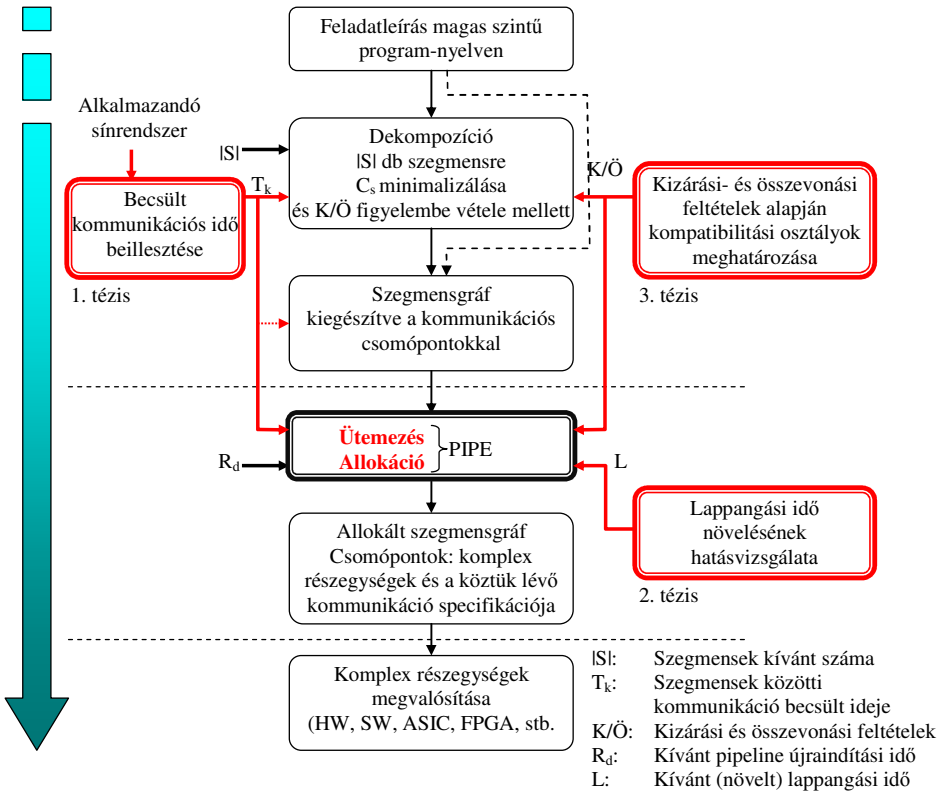
A folyamatos technológiai fejlődés következtében ma egyetlen áramköri tokozás akár több százmillió tranzisztort is tartalmazhat. Ennek következtében rendkívül nagy teljesítményű VLSI áramkörök (FPGA-k, processzorok) állíthatók elő alacsony áron. Ezen komponensek között egyaránt megtalálhatók a mikrokontrollerek és a különféle jelfeldolgozó egységek memóriával és megfelelő I/O képességekkel. Ezek a komplex jelfeldolgozó egységek önállóan képesek megoldani egyre összetettebb feladatokat egy elosztott rendszer részeként.

A fenti komplex egységeket felhasználó elosztott rendszerek tervezésére és optimalizálására azonban nem alkalmazhatók a jelenleg létező többé-kevésbé szisztematikus rendszer szintű szintézis módszerek megfelelő módosítások és kiterjesztések nélkül. Ezek a módszerek általában adatfolyam gráfból indulnak, amelyet egy magas szintű programozási nyelven (pl.: C)[7, 30] specifikált feladat dekompozíciója (partícionálása) révén állítanak elő. A jelenlegi tervező eszközök közvetlenül az adatfolyam gráfból kiindulva kísérlik meg az optimalizálást viszonylag egyszerű feldolgozó egységek feltételezésével, továbbá nem képesek kezelni a tervezés során kiadódó specifikációjú komplex feldolgozó egységeget.

Az értekezés célja, hogy a magas szintű szintézis elveinek és módszertanának vizsgálata alapján javaslatot tegyen annak olyan kiterjesztésére, amely alkalmassá teszi a meglévő magas szintű szintézis algoritmusok alkalmazását tetszőleges komplexitású részegységekből álló elosztott (többprocesszoros) rendszerek struktúrájának szisztematikus szintézisére és optimalizálására.

Ezen belül konkrétan:

- Módszert adni arra, hogy hogyan lehet a komplex elemi műveletek végrehajtására alkalmas komplex jelfeldolgozó egységek kommunikációs csatornáinak tulajdonságait figyelembe venni a tervezés folyamatában.
- Megvizsgálni a pipeline elvű rendszerek tervező algoritmusai által meghatározott lappangási idő adott mértékű növelésének hatását.
- Módszert adni annak vizsgálatára, hogy - azonos átbocsájtási tényező esetén - a kapott lappangási idő növelésével, lehet-e az eredetinél kedvezőbb költségű megoldást találni.
- Módszert kidolgozni arra, hogy hogyan lehet előre meghatározott peremfeltételeket (pl.: fizikai tulajdonságok, szabvány előírások, stb.) szisztematikus módon kezelni és azokat figyelembe venni a tervezés különböző fázisaiban.



1. ábra HLS módszerek kiterjesztése elosztott pipeline rendszerek tervezésére

Az elosztott rendszerek tervezésének folyamatát - az említett módszertant, és a későbbiekben ismertetett kiegészítéseket, algoritmusokat felhasználva – az 1. ábra mutatja. Az értekezésben bemutatott módszerek ebbe a folyamatba illesztik be a HLS algoritmusokat, kiterjesztve azok alkalmazásának határait (az ábrán ezek a kiterjesztések vastag keretben láthatók).

Tudományos eredmények

1. A kommunikáció idejének figyelembe vétele

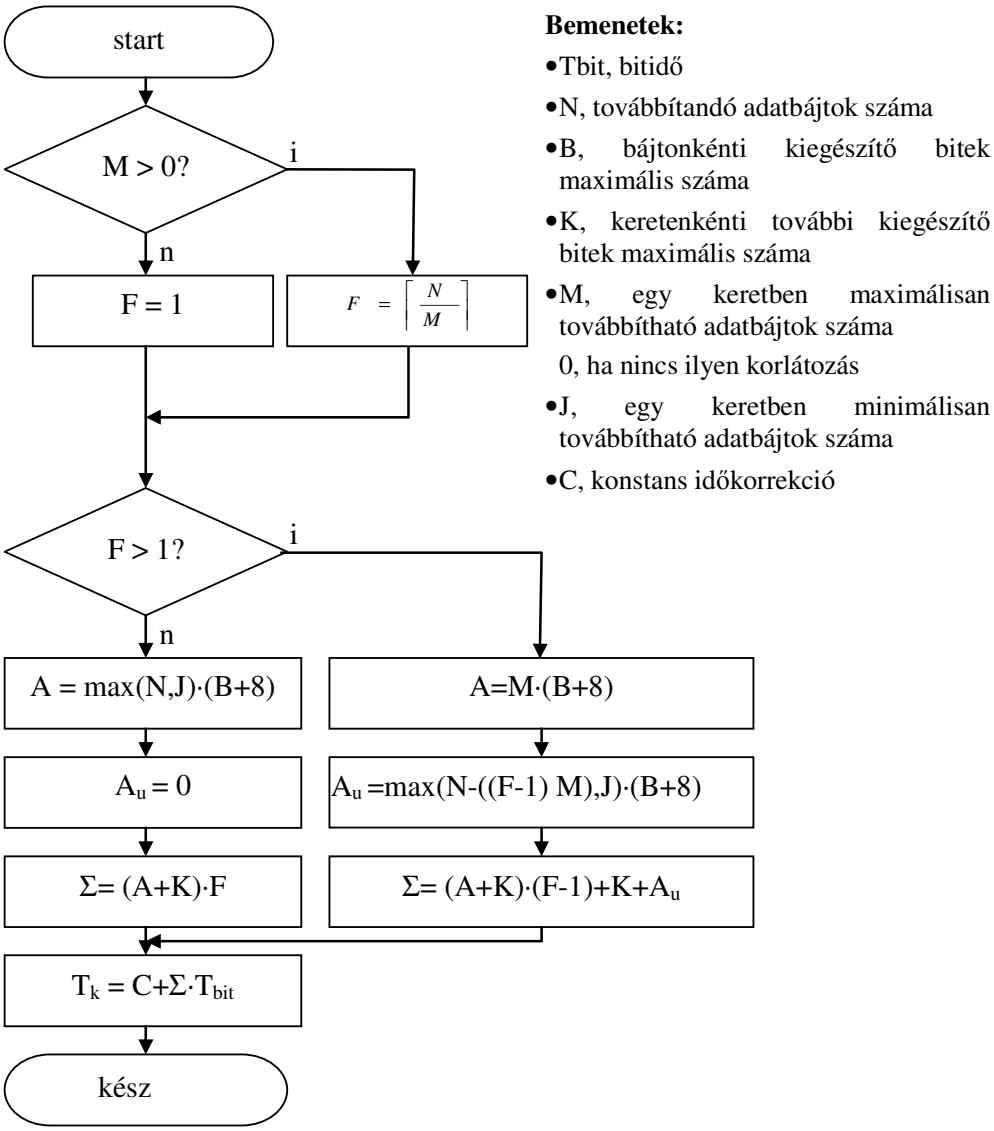
Különböző HLS tervező rendszerek és algoritmusok állnak rendelkezésre a műveleti gráf optimalizálására, ütemezésére, allokálására [3, 25]. Az ismert HLS tervező rendszerek közös tulajdonsága, hogy lényegében nem foglalkoznak az elemi műveleti gráf csomópontjai közötti és az allokáció révén kialakuló feldolgozó egységek közötti kommunikáció idejével, azt általában nulla végrehajtási idejű lépésnek tekintik. A [6, 31, 32] szakirodalom utal a kommunikáció figyelembevételének fontosságára, de a bitszámon és néhány alapvető jellemzőn túlmenően nem számol a kommunikációhoz szükséges idővel [6]. Ez a megoldás egy FPGA-n belüli kommunikáció esetén elfogadható közelítésnek tekinthető, azonban több, különálló komplex műveletvégző alkalmazása esetén a legtöbb esetben nem alkalmazható. Mikroprocesszorokat vagy mikrokontrollereket is tartalmazó komplex műveletvégzők alkalmazásakor a beintegrált kommunikációs csatornák (perifériák) használata esetén elkerülhetetlen azok időigényének figyelembe vétele [5].

Az általam kidolgozott módszer szerint, a kommunikációs kapcsolat egy pótlólagos elemi műveleti csomóponttal ábrázolható, azonban ehhez szükség van a csomópont műveletvégzési idejének ismeretére, vagy legalább megfelelő becslésére. Ha ugyanis alulbecsültük a kommunikáció tényleges idejét, akkor a végeredményként kapott hálózat működésképtelen lesz. Ha pedig jelentősen túlbecsültük a kommunikációt, akkor feleslegesen lassítjuk le a rendszert. Az optimális megoldás a pontos kommunikációs idő meghatározása lenne, de ez bizonyos esetekben nem lehetséges (pl.: bitbeszúrás adatfüggősége miatt), ezért ilyenkor egy felső becslés meghatározása tűnik célszerűnek.

A disszertációban példaként felhozott kommunikációs sínrendszerek működésének elemzése során az alábbi megállapításokat tehetjük:

- A legkisebb továbbítható adategység a legtöbb interfész esetén egy bájt.
- Az adatbájtokat gyakran kiegészítő bitekkel látják el (pl.: start, stop, ack).
- A kommunikációhoz általában tartozik keretenként (csomagonként) egy fix hosszúságú és egy változó hosszúságú kiegészítő bitmező.
- A kommunikáció időigényét jól becsülhetjük a bitidőzítés egész számú többszöröseként.

A vizsgált sínrendszerek alapján javasolt becslési algoritmust foglalja össze a 2. ábra



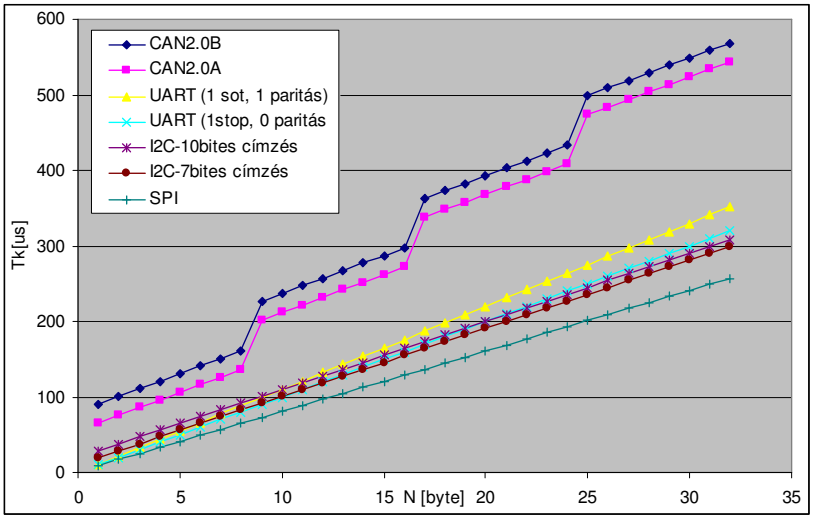
2. ábra A kommunikációs időbecslő algoritmus folyamatábrája

Az előzőekben kidolgozott algoritmus paraméterezése a példaként vizsgált négy sínnél és néhány további (itt részletesen nem bemutatott) interfész esetében az 1. táblázat szerint alakul. Ez utóbbi interfészek ismertetése a [S6, 27, 28, 29, 30] irodalmakban részletesen megtalálható.

Az algoritmus eredményeit felhasználva a 3. ábra mutatja, hogy pl.:1Mbit/s adatátviteli sebességet feltételezve, hogyan alakulnak az értekezésben részletesen is bemutatott különböző csatornák kommunikációs idejei 1-32 bájt továbbítása esetén. Az ábrán megfigyelhető, hogy a CAN interfész esetén a maximált 8 bájtos keretméret miatt az ezt meghaladó adatok továbbításához újabb keretekre van szükség, ami a kommunikációs overhead-et növeli.

Interfész	T_{bit}	C [µs]	B [Bit]	K [Bit]	M [Bájt]	J [Bájt]
SPI	50ns min.	0	0	1	0	0
I^2C, (100kHz-1MHz) 7bites cím	1-10µs	0	1	10	0	0
I^2C, (100kHz-1MHz) 10 bites cím	1-10µs	0	1	18	0	0
I^2C, High speed (3.4MHz), 7bites cím	295ns	25	1	10	0	0
I^2C, Ultra high speed (5MHz), 7bites cím	200ns	0	1	10	0	0
UART (1 stop, 0 paritásbit)	100ns min.	0	2	0	0	0
UART (1 stop, 1 paritásbit)	100ns min.	0	3	0	0	0
CAN 2.0A	1µs min.	0	2	56	8	0
CAN 2.0B	1µs min.	0	2	81	8	0
BME-PS CAN plus	1-10µs	0	2	113	4	4
Ethernet 802.3. (10Mbps)	100ns	0	0	26	1500	46
USB low-speed (1.5Mbps), control	666ns	0	1	504	8	0
USB low-speed (1.5Mbps), interrupt	666ns	0	1	152	8	0
USB full-speed (12Mbps), control	83,3ns	0	1	360	64	0
USB full-speed (12Mbps), Interrupt	83,3ns	0	1	104	64	0
USB full-speed (12Mbps), Isochronous	83,3ns	0	1	72	1023	0
USB full-speed (12Mbps), Bulk	83,3ns	0	1	104	64	0
USB high-speed (480Mbps), control	2,083ns	0	1	1384	64	0
USB high-speed (480Mbps), Interrupt	2,083ns	0	1	124	1024	0
USB high-speed (480Mbps), Isochronous	2,083ns	0	1	112	1024	0
USB high-speed (480Mbps), Bulk	2,083ns	0	1	440	512	0
SATA (1.5GB/s), parancskeret	666,6ps	0	2	96	64	64
Hyper Transport (2bit,200MHz)	2,5ns	0	0	64	64	4
Hyper Transport (32bit,200MHz)	156,25ps	0	0	64	64	4
PCI-Express x1 (2.5Gb/s)	400ps	0	2	144	4096	0
PCI-Express x2 (2.5Gb/s)	200ps	0	2	288	8192	0
PCI-Express x4 (2.5Gb/s)	100ps	0	2	1152	16384	0
PCI-Express x8 (2.5Gb/s)	50ps	0	2	2304	32768	0
PCI-Express x16 (2.5Gb/s)	25ps	0	2	4608	65536	0
PCI-Express x32 (2.5Gb/s)	12,5ps	0	2	9216	131072	0

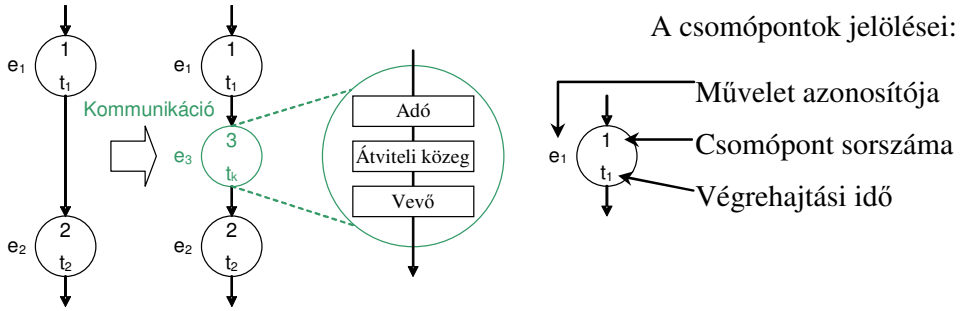
1. táblázat Különböző interfészek lehetséges paraméterei a becslő algoritmushoz



3. ábra Kommunikációs idők 1-32 byte továbbítása esetén 1µs bitidőt alkalmazva

Az eredmények alkalmazása HLS tervező rendszerben

A magas szintű tervezőrendszerek elemi műveleti gráffal (EOG), vagy más néven adatfolyam gráffal (DFG) reprezentálják a megoldandó feladatot. A 4. ábra egy egyszerű példát mutat az e_1 és e_2 jelű elemi műveletek közé beiktatott kommunikációs csatorna (e_3) ábrázolására a PIPE tervező rendszer elemi műveleti gráfjára alkalmazva.

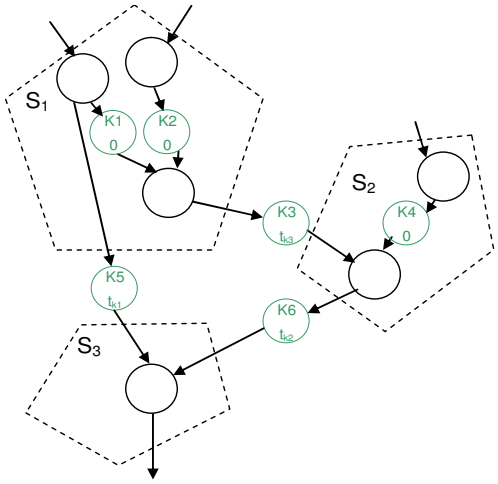


4. ábra Kommunikációs csatorna ábrázolása az EOG-ben

Az e_3 jelű kommunikációs művelet végrehajtási (lappangási) ideje t_k. Az előző pontban bemutatott kommunikációs idő (T_k) azonban a bitsebességtől függ, ezért azt át kell transzformálni a HLS tervező eszköz által alkalmazott időzítési rendszerbe. A kommunikációt jellemző T_k kommunikációs idő, míg a HLS rendszer alapütemezése a T órajel periódusidő. Ezek alapján a két időtartomány átskálázható a következő összefüggés szerint:

$$t_k = \frac{T_k}{T}$$

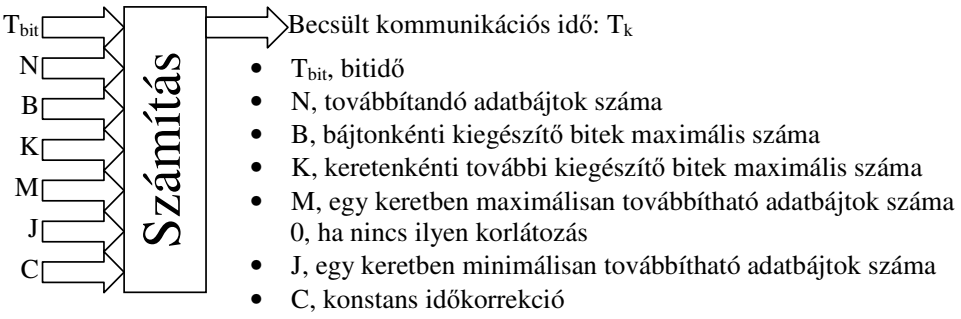
Az értekezésben javasolt egyik módszer szerint a tervezés elején a szegmentálás előtt minden elemi művelet közé be kell iktatni kommunikációs csomópontot. Ezt az esetet mutatja 5. ábra. A szegmensen belüli csomópontokat 0, míg a szegmensek közöttieket t_k kommunikációs idővel kell figyelembe venni



5. ábra Példa a szegmentálás során kiadódó szegmensgráfra a beszúrt kommunikációs csomópontokkal

1. Tézis

- **A magas szintű tervezőrendszereknél a megoldandó feladatot reprezentáló elemi műveleti gráfnál használt elemi művelet fogalmát kiterjeszthetjük komplex műveletre is. Ez megadható akár algoritmussal, a feladat leírásnál, vagy előállítható iteratív úton a HLS rendszerrel is. A komplex műveleteket komplex jelfeldolgozó egységekkel (processzor, mikrokontroller, jelfeldolgozó kontroller) valósíthatjuk meg. Ilyen komplex műveletvégző egységek esetében elkerülhetetlen a kommunikációs idő figyelembevétele.**
- **Kidolgoztam egy módszert, amely szerint komplex műveletvégzők alkalmazásakor az elemi műveleti gráfban a komplex műveletvégzők közötti kommunikációt műveletként kell figyelembe venni. Ennek műveleti ideje a műveletvégzőkben megvalósított kommunikációs csatornák tulajdonságai és az átviendő információ mennyisége alapján számított kommunikációs idő.**
- **Kidolgoztam egy algoritmust, amellyel ez az idő számítható és a rendszer órajelével skálázható. A leggyakrabban alkalmazott kommunikációs csatornákra meghatároztam a kommunikációs időket.**



- Példán bemutattam, hogy a hagyományos HLS algoritmus, amelynél a kommunikációs költséget nem veszik figyelembe olyan megoldást eredményez(het), amely fizikailag nem oldható meg a szokásosan beépített hardveres kommunikációs csatornával.
- Példán bemutattam, hogy az általam kidolgozott módszer szerint módosított HLS (PIPE) algoritmus realizálható megoldást eredményez.
- A módszer és az algoritmus kidolgozása teljes egészében saját eredmény.
- A tézishez kapcsolódó publikációk: [S2], [S6], [S8] és [S9].

2. A lappangási idő növelésének hatása az egyes elemi műveletek újrafelhasználhatóságára

A HLS tervező rendszerek - az algoritmusaik révén - az újraindítási (inicializálási) idő, illetve ennek reciprokaként számított átbocsájtási tényező optimalizálását elsődleges célnak tekintve, generálnak egy struktúrát. A lappangási időt ebből a struktúrából kiadódó paraméterként számítják [3]. Nem vizsgálják, hogy a lappangási idő kismértékű növelése milyen hatással lenne a teljes megoldás bonyolultságára, költségére pipeline működés esetén. A következőkben ismertetett vizsgálat célja, hogy megállapítsa, hogy adott R-hez tartozó L szándékos növelése milyen hatással lehet a teljes pipeline feldolgozó hálózat összköltség-függvényére. A vizsgálathoz a BME-IIT-n kifejlesztett PIPE HLS tervező rendszert használtam, de következtetések érvényesek más magas szintű tervező eszközök alkalmazása esetén is.

A lappangási idő növelésétől bizonyos esetekben a megvalósítás költségének csökkenése remélhető. Ez várható, ha egymást átlapoló elemi műveletek nagyobb lappangási idő megengedése és megfelelő ütemezés esetén, átlapolás nélkül időben egymás után indíthatóvá válnak, így az allokáció során ugyanazzal a műveletvégző egységgel lennének megvalósíthatók.

A költség csökkenéséhez szükséges (de nem elégséges) szabályok:

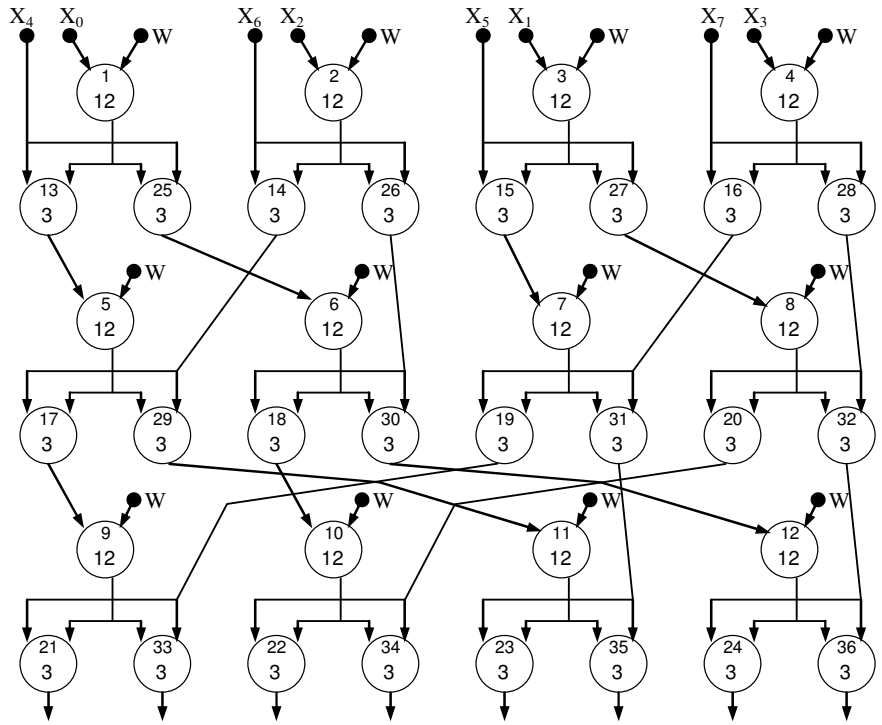
1. Nyilvánvalóan csak olyan műveletet végzőket érdemes vizsgálni, amelyből egynél többet tartalmaz allokált EOG.
2. Érdemes meghatározni a legkisebb költséget, mert ha ezt elértük, akkor további csökkenés már nem lehetséges. Például, ha egy EOG négy különböző egy egységbe nem összevonható elemi műveletet tartalmaz, akkor nyilvánvaló, hogy négynél kevesebb műveletvégzővel nem lehet az adott feladatot megoldani.
3. Adott újraindítási idő mellett meghatározható a különböző műveletvégző egységek elméletileg szükséges legkisebb darabszáma. Ezen minimális darabszám alapján kiszámítható az adott körülmények között elérhető legkisebb költség is. Ezen költség elérését leállási feltételként vehetjük figyelembe.
4. Egyszerre induló azonos végrehajtási idejű műveletek esetén legalább a művelet végrehajtási idejének megfelelő mértékben kell növelni L értékét, hiszen átlapoló műveletek nem kerülhetnek egy egységbe.
5. Egymást átlapoló műveletek legalább az átlapolás idejének megfelelő L növelés esetén ütemezhetők egymás után, ha más kizáró feltétel nincs.

6. Ahhoz, hogy egy M_i komplex műveletvégzőt egy újraindítási perióduson belül k-szor felhasználhassunk, szükséges, hogy a műveletvégzőbe allokált k darab elemi művelet végrehajtási idejének (t_i) összege ne legyen nagyobb az újraindítási időnél. $\sum_k t_k \le R$

   Amennyiben a komplex műveletvégzőbe allokált elemi műveletek azonos végrehajtási idejűek, akkor a fenti összefüggés az alábbi alakra egyszerűsödik:

   $k \cdot t_k \le R$

A disszertációban több példán keresztül vizsgáltam és értékeltem a lappangási idő növelésének hatását. A 6. ábra az F2 feladat elemi műveleti gráfját mutatja.



6. ábra Az F2 feladat elemi műveleti gráfja

Az elemi műveleti gráfon található háromféle műveletvégző jellemzőit a 2. táblázat foglalja össze.

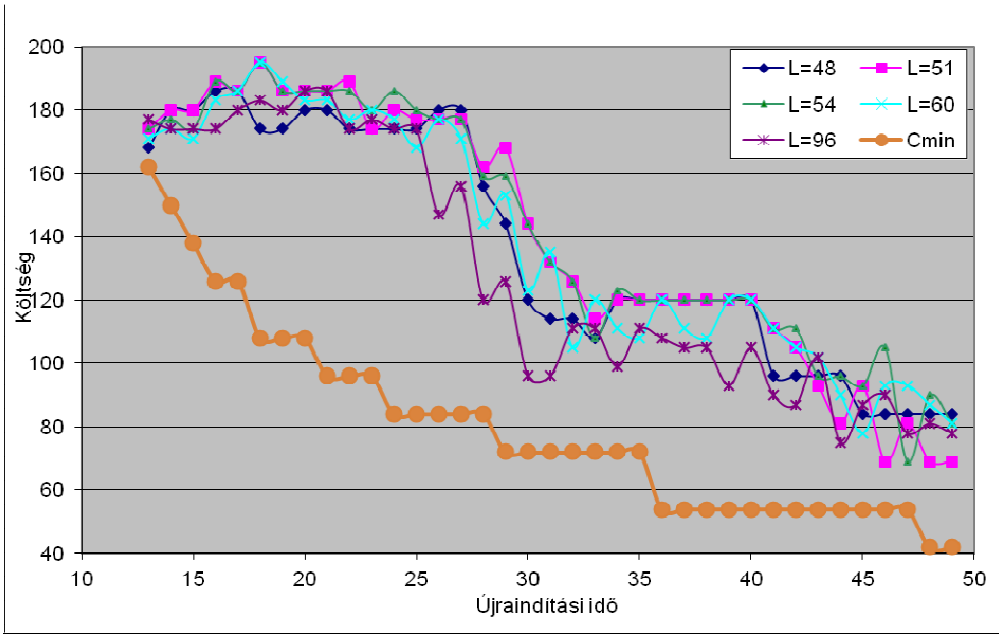
Műveletvégző	Elemi művelet	ti	Megvalósítandó elemi műveletek [db]
P_1	$e_1 \ldots e_{12}$	12	12
P_2	$e_{13} \ldots e_{24}$	3	12
P_3	$e_{25} \ldots e_{36}$	3	12

2. táblázat Az egyes műveletvégzők tulajdonságai és darabszáma

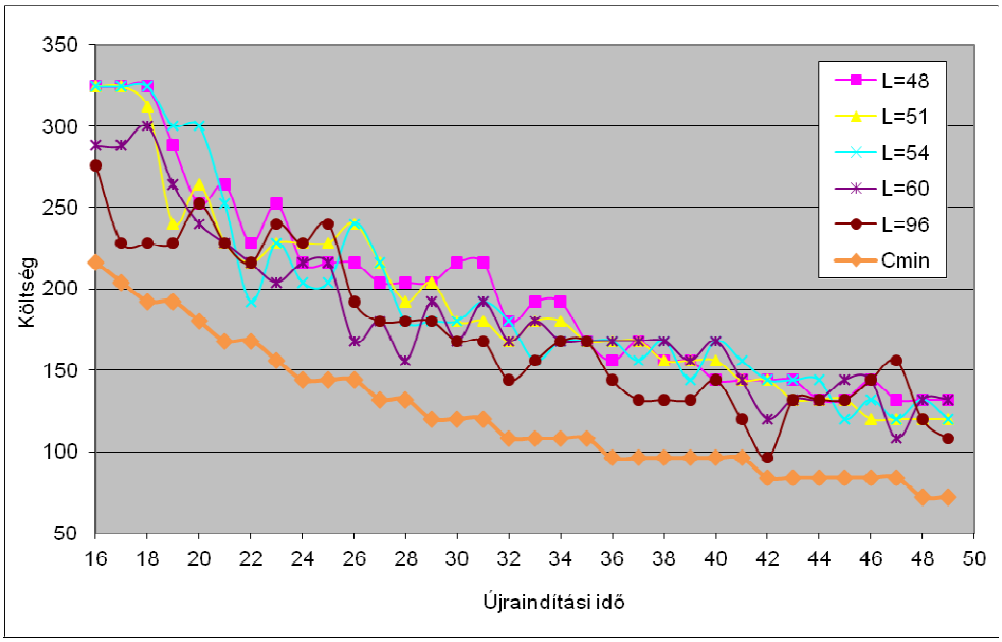
A különféle megoldások összehasonlítása egy egyszerű költségfüggvény alapján történt. A költséget a felhasznált műveletvégzők darabszámának és végrehajtási idejük szorzataként értelmeztük.

A szimulációs vizsgálatot többféle műveletvégző esetére is elvégeztem. A 7. ábra és a 8. ábra mutatja a költségfüggvényeket egyszerű és komplex műveletvégzők alkalmazása esetén. A

legalsó görbék az adott újraindítási időhöz tartozó elméleti minimumot mutatják. Megfigyelhető, hogy nagyobb lappangás esetén általában közelebb kerülünk ehhez a költséghez.



7. ábra Az F2 feladat költségfüggvényei háromféle műveletvégző alkalmazása esetén



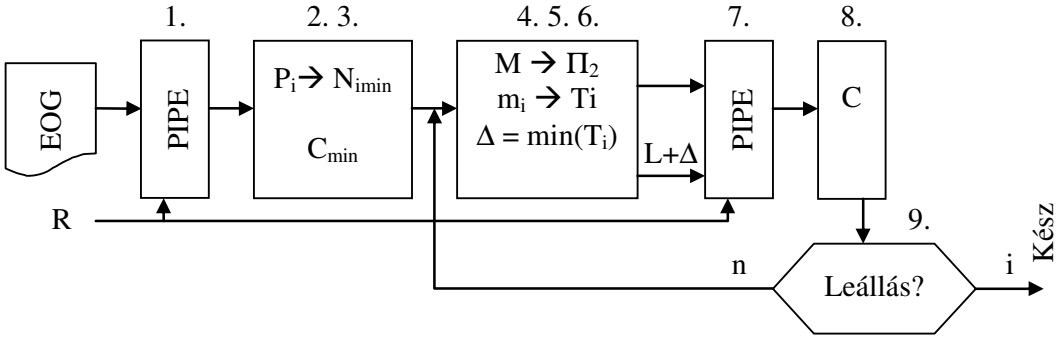
8. ábra Az F2 feladat költségfüggvényei komplex műveletvégzők alkalmazása esetén

A javasolt algoritmus a lappangási idő növelésének hatásvizsgálatára

A következő algoritmus paramétereinek pontos meghatározása az értekezésben található, itt csak a végeredményt ismertetem.

1. Az eredeti EOG-re a megadott R értékkel a PIPE lefuttatása. A futási eredmény alapján L_{min} adódik.
2. Minden P_i műveletvégzőhöz meghatározzuk a belőle felhasználható legkevesebb darabszámot (N_{iMIN}). $N_{iMIN} = \sum_k \left\lceil \frac{n_k \cdot t_k}{R} \right\rceil$ minden olyan k-ra amely műveletet az P_i végre tud hajtani
3. C_{min} kiszámítása
4. Létrehozzuk az M halmaz alapján a Π_M partíciót. A partíció blokkjai közül kiválasztjuk azokat, amelyekre teljesül, hogy $N_i > N_{iMIN}$ és $T_i \leq R/2$
5. A lappangási idő növeléséhez választunk egy Δ értéket az alábbiak minimumaként:
   - A legnagyobb N_i értékű m_i blokkhoz tartozó végrehajtási idő: T_i
   (Ha több azonos N_i értékű blokk van, akkor ezek közül a legkisebb T_i-t választjuk).
6. A kiválasztott Δ növekménnyel megnöveljük az aktuális lappangás értékét.
7. PIPE újrafuttatása.
8. C költség kiszámítása, leállási feltétel vizsgálat, majd ismétlés a 4. lépéstől.
9. Leállás az alábbi feltételek bármelyikének teljesülése esetén:
   - ha elértük az előre megadott (C_d) elvárt költséget vagy
   -ha elértük az előre megadott L_{max} maximális lappangási időt vagy
   -ha elértük a minimumok alapján számított legkisebb költséget (C_{min}).

Az algoritmus egyszerűsített folyamatábráját a 9. ábra mutatja.



9. ábra A lappangási idő növelő algoritmus folyamatábrája

A fenti algoritmus célja, hogy ne kelljen egyesével minden egyes L érték esetén újrafuttatni a PIPE tervező algoritmust az összes lehetséges R időre, hanem csak olyan L értékek esetén, ahol érdemben várható költségcsökkenés. Az L növelése egy határon túl természetesen értelmetlen, mert feleslegesen lelassítja a rendszer működését.

2. Tézis

A pipeline rendszerek magas szintű szintézisekor az optimalizálás célja a költség és az energiafelhasználás mellett az újraindítási idő lehetőség szerinti csökkentése. A lappangási idő az előzőek függvényében kiadódó paraméter, így a lappangási idő növelésének hatása nem tárgya az analízisnek.

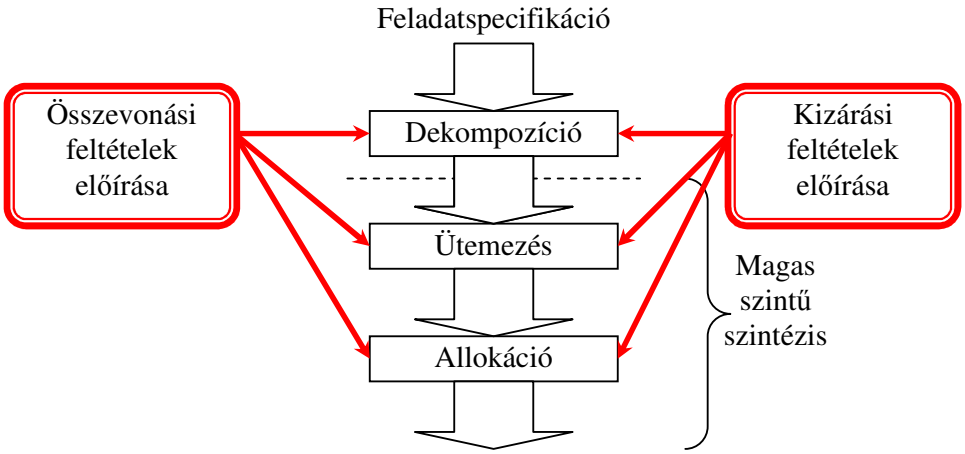
- **A pipeline rendszerek esetében érdemes a lappangási idő növelésének hatását is vizsgálni, mert a lappangási idő növelésének kedvező hatása lehet a teljes rendszer költségére, miközben ez nem lassítja a rendszer működését, ugyanis az átbocsájtási tényezőre gyakorolt hatása ilyenkor elhanyagolható.**
- **Olyan vizsgáló eljárást és algoritmust fejlesztettem ki, amelynek segítségével megállapítható a lappangási idő növelésének hatása az optimalizálás eredményére.**
- A kapott eredményeket szimulációs vizsgálatokkal ellenőriztem, értékeltem.
- Módszert adtam általános felhasználásra a lappangási idő változtatására.
- A módszer lehetővé teszi, hogy az R és L értékét együttesen változtatva határozzák meg az optimális költséget.
- Javaslatomra a felhasznált PIPE tervezőrendszert úgy módosították, hogy az lehetővé teszi a lappangási idő bemenő paraméterként való megadását (növelését).
- A tézisben megfogalmazott módszer és algoritmus, valamint a teszteredmények teljes egészében saját munkáim.
- Az eredményekhez kapcsolódó publikáció: [S3]

3. Előre megadott kizárási és összevonási előírások figyelembevétele az allokáció során

Ipari folyamatok automatizálása során az elosztott vezérlő rendszereket sokszor nem megfelelően partícionálják. Bár a kívánt feladatot a teljes rendszer hibátlan működés esetén ellátja, de bármilyen műszaki hiba miatti leállás esetén a rosszul megvalósított feladat-partícionálás miatt meglehetősen nehézkes és időigényes a hibakeresés, javítás. Egyes részegységek hibája miatt így esetleg olyan további részegységek is leállhatnak, amelyek önmagukban még működőképesek maradhatnának, ha nem lettek volna a meghibásodott részegységgel közös berendezésbe allokálva. Ugyancsak nehézséget okozhat, ha egymáshoz szorosan kapcsolódó funkciójú részegységek különböző, egymástól távoli berendezésekbe kerülnek. Ez a kialakítás megnehezíti az egyes funkciók kipróbálását és tesztelését is, hiszen egy próbához is minden érintett berendezésnek egyidejűleg működőképesnek kell lennie, illetve egy ilyen elosztott módon megvalósított funkció végrehajtásában bekövetkező hiba több érintett berendezés leállását is okozhatja.

A magas szintű szintézis módszereket ilyen peremfeltételek mellett is előnyösen lehetne alkalmazni, de a jelenlegi eszközök nem teszik lehetővé ezek szisztematikus figyelembevételét. Maga a probléma nagyon hasonlít a hardver-szoftver együttes tervezés bizonyos problémáira [22, 23, 24], hiszen a vezérlő rendszer egy része hardver, egy másik része szoftver (pl.: szelepek + jelfogók + érzékelők + PLC + működtető program). Természetesen vannak kötöttségek, bizonyos funkciók kizárólag hardverelemekkel oldhatók meg (pl.: jelátalakítók, érzékelők, beavatkozók), azonban a vezérlési és feldolgozási feladatok már egyaránt megoldhatók szoftverrel és hardverrel. Speciális alkalmazásoknál az előírt feladaton túl bizonyos szabványok előírásainak (pl.: biztonság kritikus előírások [11]) is meg kell felelni. Egy ipari gyártási folyamat vagy annak vezérlő algoritmusa - hasonlóan a magas szintű logikai szintézis feladatokhoz – reprezentálható EOG-vel. Az EOG előállítása után a HLS eszközök alkalmazhatóak a gráf alapján történő optimalizálásra.

A magas szintű tervezés folyamatát a 10. ábra mutatja. A feladatspecifikációból kiindulva első lépésként többnyire dekompozíciót is végezve képezzük EOG-t. Az EOG-t ütemezzük, majd allokáljuk az egyes elemi műveleteket. Ez utóbbi két lépés elvégzéséhez különféle magas szintű tervező rendszerek állnak rendelkezésre [3, 24].



10. ábra Összevonások és kizárások előírása a tervezés különböző fázisaiban

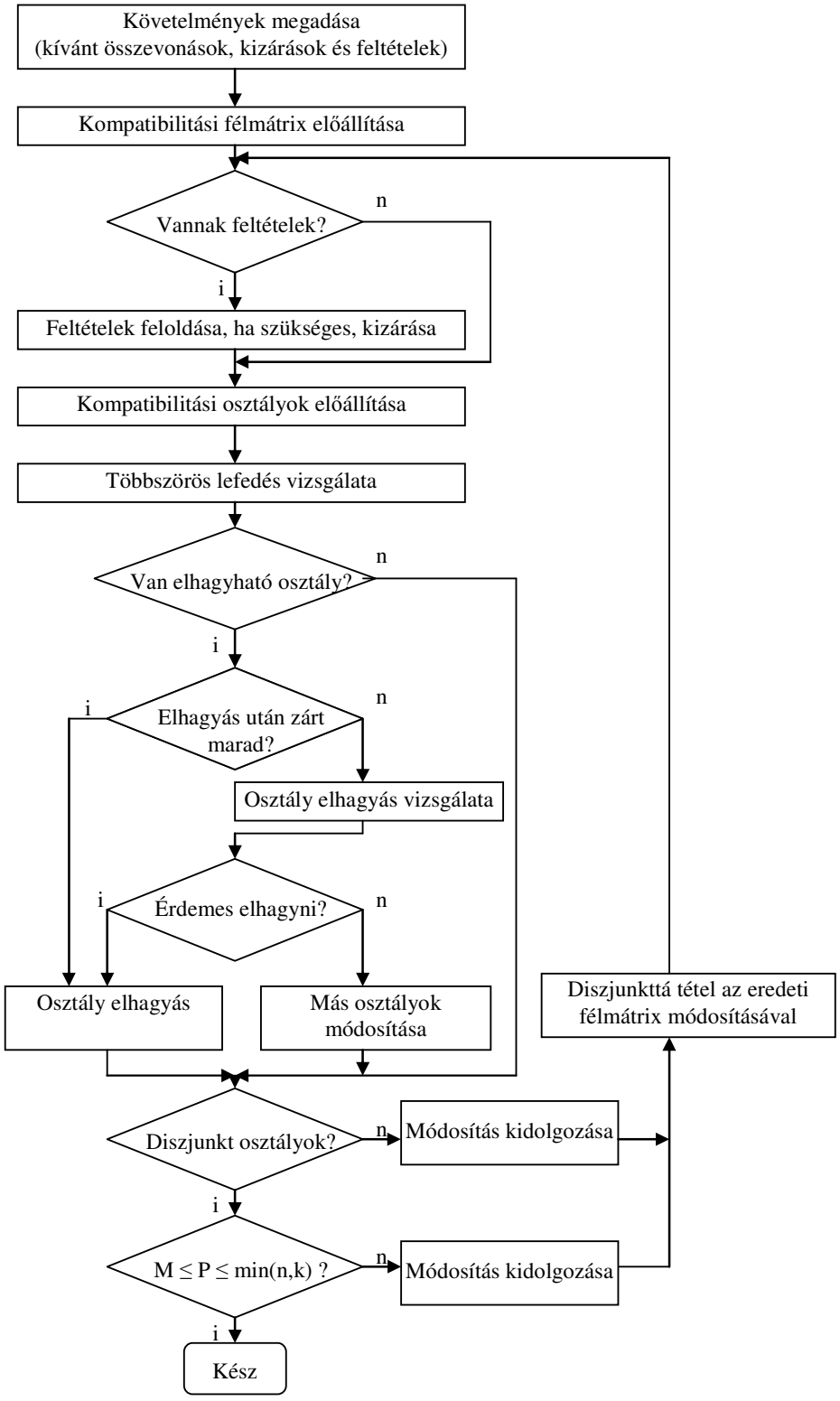
A 10. ábra szemlélteti azokat a tervezési fázisokat, amelyekben az általam megfogalmazott módszerrel a kizárási és összevonási peremfeltételek figyelembe vehetők. Az értekezésben részletesen megvizsgálom, hogy a tervezés különböző fázisaiban milyen hatásai vannak ezeknek a pótlólagos peremfeltételeknek. A páronkénti összevonhatóságról bizonyítható, hogy kompatibilitási reláció [12, 13], mivel a reflexivitás és szimmetria fennáll, a tranzitivitás viszont nem. A maximális kompatibilitási osztályokból kiindulva egy kedvező, mindenképpen diszjunkt és zárt lefedést kell meghatározni, mert egy elemi művelet értelemszerűen csak egyetlen műveletvégzőbe allokálható.

A javasolt új allokációs algoritmus

Az értekezésben bemutatottak alapján az alábbi új algoritmus fogalmazható meg:

1. A követelmény specifikáció alapján meghatározzuk az összevonhatóságot, feltételes összevonhatóságot és kizárásokat tartalmazó kompatibilitási félmátrixot.
2. Amennyiben a kompatibilitási félmátrix feltételeket is tartalmaz, úgy a feltétel rendszer zártságát is ellenőrizzük, ha nem zárt a feltétel lánc, akkor zárttá tesszük a megfelelő feltétel(ek) módosításával (a megfelelő új kizárások megadásával).
3. A kompatibilitási félmátrix alapján meghatározzuk a maximális kompatibilitási osztályokat.
4. Ellenőrizzük a kapott osztályok diszjunktságát.
5. Ha az osztályok nem diszjunktak, megvizsgáljuk, hogy van-e elhagyható osztály, amelynek elhagyása esetén a feltétellánc továbbra is zárt marad. Ha van ilyen osztály, akkor azt elhagyjuk. Ha az elhagyás révén sérül a zártság, akkor vagy nem hagyjuk el az egész osztályt, vagy mérlegelhető, hogy mégis elhagyjuk és inkább más osztályok felbontásával biztosítjuk a zártságot. Ez a lépés feladatfüggő, ezért próbálgatást igényel.
6. Amennyiben az osztályok nem diszjunktak, úgy diszjunkttá tesszük őket. A diszjunkttá tett osztályok zártságát ellenőrizzük, ha nem zártak, akkor a zárttá tesszük őket. Ez általában további osztályok létrejöttét eredményezheti.
7. Ha túl sok nem diszjunkt osztályt kapunk, és nincs mód az összes lehetőség végigpróbálgatására, akkor az $M \leq P \leq \min(n,k)$ egyenlőtlenség [13] adhat támpontot a kapott eredmény „jóságáról".
   - P: az éppen összevont, legkevesebb műveletvégzőt tartalmazó gráf műveletvégzőinek száma
   - n: a kiindulásként adott adatfolyam gráf csomópontjainak (elemi műveleteinek) száma
   - k: az eredeti kompatibilitási félmátrix alapján meghatározott maximális kompatibilitási osztályok száma
   - M: a legegyszerűbb zárt lefedést biztosító (az eredeti osztályokból képzett) maximális kompatibilitási osztályok száma.
8. A kapott osztályok alapján az allokációt elvégezzük

Az algoritmus folyamatábráját mutatja a 11. ábra.



11. ábra Allokációs algoritmus egy lehetséges folyamatábrája

3. Tézis

- **Összetett feladatok tervezése során gyakran előfordul, hogy a megvalósítandó feladaton kívül, bizonyos üzemeltetési, karbantartási vagy szabvány előírások figyelembevétele is szükséges.**
- **Új allokációs módszert dolgoztam ki, amely az ismert eljárásokhoz képest azt is lehetővé teszi, hogy az ipari alkalmazások és a mikroprocesszor illetve mikrokontroller alapú beágyazott rendszerek esetében kizáró és kívánt feltételes összevonásokat is figyelembe lehessen venni.**

- A kapott módszer előnyösen alkalmazható például összetett ipari folyamatok vagy biztonság-kritikus rendszerek szisztematikus tervezésekor.
- A kapott módszer alkalmazható az első tézisben megfogalmazott kommunikációs csatornák allokációjának elemkészlet szerinti befolyásolására is.
- Elindult egy új ütemező és allokáló algoritmus fejlesztése a PIPE tervező rendszerhez, amely már alkalmazza az itt kidolgozott módszert.
- A tézisben megfogalmazott módszer és algoritmus, valamint a teszteredmények teljes egészében saját munkáim.
- Az eredményekhez kapcsolódó publikáció: [S9] és [S10]

Az eredmények alkalmazásának lehetőségei

Az egyes tézisek konkrét alkalmazási területeit a tézisek ismertetésénél már megjelöltem. Ugyancsak leírtam a már megvalósított felhasználást is.

A Tanszéken a Digitális Célendszerek csoport foglalkozik meglévő rendszerek optimalizálásával és újratervezésével is. A disszertációban kidolgozott eredmények ezeknél a munkáknál előnyösen felhasználhatók. Várhatóan ipari megkeresések is lesznek az újratervezéshez kapcsolódó témákban.

Az első tézishez kidolgozott becslés elsősorban a HLS eszközökhöz készült, de megfelelő paraméterezéssel alkalmazható rendszerfelügyeleti célokra is például úgy, hogy egy üzenet továbbítása előtt megbecsüljük a továbbítás idejét, majd ha a becsült időn belül nem sikerült a továbbítás, akkor hibajelzést generálunk.

A második tézisben kidolgozott eljárás jól alkalmazható pipeline feldolgozó egységek tervezésekor. A Tanszéken fejlesztés alatt lévő PIPE tervezőprogramba beépítésre került a lappangási idő növelésének lehetősége.

A módszer lehetővé teszi azt is, hogy az R és L értékét együttesen változtatva határozzák meg az optimális költséget.

A harmadik tézisben kidolgozott allokációs módszer alapján elindult egy új ütemező és allokáló algoritmus fejlesztése a PIPE tervező rendszerhez.

A kapott módszer előnyösen alkalmazható például összetett ipari folyamatok vagy biztonság-kritikus rendszerek szisztematikus tervezésekor.

A kapott módszer alkalmazható az első tézisben megfogalmazott kommunikációs csatornák allokációjának elemkészlet szerinti befolyásolására is.

Az értekezésben bemutatott példák ugyan a PIPE tervezőprogrammal készültek, de a megoldás más tervező eszközök esetében is alkalmazható, nem kötődik speciálisan ehhez a tervező rendszerhez. Az egyes tézisek eredményei részben alkalmazásra kerültek a BME Irányítástechnika és Informatika Tanszékén folyó OTKA K72611, valamint a TÁMOP 4.2.1/B-09/1/KMR-2010-0002 kutatási projektjeiben.

Köszönetnyilvánítás

Hálás köszönettel tartozom témavezetőmnek, Dr. Arató Péternek és konzulensemnek, Dr. Móczár Gézának a sok segítségért. Tanácsaik, javaslataik és hasznos észrevételeik nagymértékben hozzájárultak a sikeres kutatói munka végzéséhez. Rengeteg gondolatom született a közös munka és a megbeszéléseink során.

Kutató munkámat az OTKA K72611 számú programja, a TÁMOP 4.2.1/B-09/1/KMR-2010-0002 programja támogatta Budapesti Műszaki és Gazdaságtudományi Egyetem Irányítástechnika és Informatika Tanszékén.

Publikációk

Lektorált folyóiratcikk

S1. Komáromi Zoltán, Pilászy György, Móczár Géza, “Hardware-software co-design in control engineering based on MATLAB environment”, ISAST TRANSACTIONS ON COMPUTERS AND INTELLIGENT SYSTEMS 2:(2) pp. 95-100. (2010)

S2. György Pilászy, György Rácz, Péter Arató,”Communication Time Estimation in High Level Synthesis”, Periodica Polytechnica, 2012, lektorálás és megjelenés folyamatban

S3. György Pilászy**,** György Rácz, Péter Arató, **„**The effect of increasing the latency time in High Level Synthesis”, 2013, Periodica Polytechnica, lektorálás és megjelenés folyamatban

Könyvfejezet

S4. Pilászy György, Móczár Géza, “Angol-magyar informatikai szótár”, a számítógép architektúrák és digitális rendszerek témakör könyv fejezeteinek elkészítése, Budapest: Tinta Könyvkiadó, 2006. pp. 1-384., ISBN: 963 9372 79 X

Könyv, Felsőoktatási tankönyv

S5. Pilászy György, Horváth Tamás, ”Digitális technika feladatok és megoldásuk”, Elektronikus formában közzétett egyetemi jegyzet az MSC villmosmérnök szakra jelentkező hallgatók számára

S6. Pilászy György, “Gyakorlati anyagok a Beágyazott irányító rendszerek tantárgy gyakorlataihoz.: A CAN és LIN protokollok”, Elektronikusan közzétett egyetemi jegyzet

S7. Pilászy György, Horváth Tamás, “Digitális technikai alapmérések”, Jegyzetszám: VI205010, 2011, BME Viking Zrt

Konferenciaközlemény

S8. Pilászy György, Móczár Géza, ”Moduláris mikrokontrolleres rendszerek távoli felügyelete”, In: Measurement and Automation. Miskolc, Magyarország, 2001.03.01-2001.03.02. pp. 45-50. Paper, Section F:Measurement and Automation, microCAD 2001, Miskolc, 2001

S9. Pilászy György, Móczár Géza, “Modular Microcontroller System”, In: Lehoczky László, Kalmár László (szerk.), MicroCAD 2004 International Scientific Conference. Miskolc, Magyarország, 2004.03.18-2004.03.19. Miskolc: Miskolci Egyetem Innovációs és Technológia Transzfer Centruma, pp. 81-86. Vol. G, Automation and telecommunication (Automatizálás és telekommunikáció) (ISBN: 963 661 615 9)

S10. Pilászy György, Komáromi Zoltán, Móczár Géza, “Központi irányítóegység moduláris, elosztott intelligenciájú irányítástechnikai rendszerekhez”, In: Bíró Károly Ágoston Sebestyén-Pál György (szerk.), IX. ENELKO - XVIII. SámOkt Nemzetközi energetikai-elektrotechnikai és számítástechnikai konferencia. Csíksomlyó, Románia, 2008.10.09-2008.10.12. (EMT ; Babes-Bolyai University) Kolozsvár: Erdélyi Magyar Műszaki Tudományos Társaság, pp. 172-178. Paper &. Vol. 1

S11. Pilászy György, Komáromi Zoltán, Móczár Géza, “Elosztott irányítórendszer napkollektoros enegetikai rendszerekhez”, In: Bíró Károly Ágoston, Sebestyén-Pál György (szerk.), IX. ENELKO - XVIII.SzámOkt Nemzetközi energetikai-elektrotechnikai és számítástechnikai konferencia. Csíksomlyó, Románia, 2008.10.09-2008.10.12. (EMT ; Babes-Bolyai University) Kolozsvár: Erdélyi Magyar Műszaki Tudományos Társaság, pp. 220-225. Paper &. Vol. 1

S12. Pilászy György, Komáromi Zoltán, Móczár Géza, “A method for designing and installing distributed intelligent, embedded systems”, In: ISW1 International Scientific Workshop on DCS 1th Meeting: &. Lillafüred, Magyarország, 2010.10.25-2010.10.27. Miskolc-Lillafüred: University of Miskolc, pp. 58-65. Paper, (Proceedings of 1st International Scientific Workshop on DCS; 1.) Vol. 1, Proceedings of 1st International Scientific Workshop on DCS (ISBN: 978-963-661-950-3) This publication is supported by the Hungarian Section of IEEE

S13. Pilászy György, Móczár Géza, Rácz György, “Microcontroller based variable-speed low-power induction motor drive”, In: Dr Bíró Károly Ágoston, Dr Sebestyén-Pál György (szerk.)ENELKO2011 XII. Nemzetközi Energetika-Elektrotechnika konferencia. Kolozsvár, Románia, 2011.10.06-2011.10.09. Kolozsvár: pp. 79-90. Paper 13. (ISBN: 1842-4546)

S14. Rácz György, Pilászy György, Móczár Géza, ”Efficient microcontroller based methods for single phase induction motor control”, In: Ildikó BÖLKÉNY (szerk.), Proceedings of 2nd International Scientific Workshop on DCS. Miskolc, Magyarország, 2011.10.24-2011.10.26. Miskolc: pp. 26-33. Paper 3. (ISBN: &)

Egyéb, Diplomamunka, TDK dolgozat

S15. Pilászy György, ”Programozható hő- és villamos energia kapcsoló berendezés”, (1998) TDK

S16. Pilászy György, Móczár Gergő, ”Beágyazott rendszerek távoli felügyelete”, (2000) TDK

S17. Pilászy György, Móczár Gergő, ”Távoli eléréssel rendelkező intelligens mikrokontrolleres vezérlő”, (2000) TDK

S18. Pilászy György, Király Pál, ”Villamos kisülések gázokban”, (2000) TDK

S19. Pilászy György, ”Távoli elérésű intelligens mikrokontrolleres vezérlő”, (2002), Diplomamunka

S20. Móczár Géza, Pilászy György, Nagy Ákos, ”Intelligens, elosztott, öntanuló irányítási rendszer kialakítása napenergiás és vegyes rendszerekhez.: GVOP-3.1.1.-2004.05.0515/3.”, pp. 1-170., (2005)

S21. Arató Péter, Móczár Géza, Pilászy György és társai, ”Feladatfüggő felépítésű többprocesszoros célrendszerek szintézis algoritmusainak kutatása.: OTKA T72611.”, (2011)

Irodalomjegyzék

[1] CAN specification 2.0A (http://www.can-cia.org/index.php?id=441)

[2] CAN specification 2.0B (http://www.can-cia.org/index.php?id=441)

[3] Péter Arató, Tamás Visegrády, István Jankovits: High Level Synthesis of Pipelined Datapaths, John Wiley & Sons, New York, ISBN: 0 471495582 4, 2001

[4] I2C-bus specification and user manual Rev. 4 — 13 February 2012 (http://www.nxp.com/documents/user_manual/UM10204.pdf)

[5] Pilászy György, Móczár Géza, Remote control of modular microcontroller systems, Microcad 2001, In: Measurement and Automation. Miskolc, Hungary, 2001.03.01-2001.03.02. pp. 45-50.

[6] Michel Goraczko, Jie Liu, Dimitrios Lymberopoulos: Energy-Optimal Software Partitioning in Heterogeneous Multiprocessor Embedded Systems, DAC 2008, June 8–13, 2008, Anaheim, California, USA

[7] P. Arató, D. Drexler, G. Kocza, G. Suba: Synthesis of a Task-dependent Pipelined Multiprocessing Structure, submitted to the ACM Transactions on Design Automation of Electronic Systems

[8] The I2C Bus specification version 2.1, January 2000, http://www.nxp.com/documents/other/39340011.pdf

[9] Philippe Coussy, Daniel D. Gajski, Michael Meredith, Andres Takach, An Introduction to High-Level Synthesis, IEEE Design & Test of Computers, 2009

[10] 8251A Programmable communication interface, 1993, Intel Corporation, Document number: 205222-003

[11] IEC/EN60730 Safety Standard

[12] Arató Péter, Logikai rendszerek tervezése, BME Printer kft, 2011

[13] H. Peterson, Introduction to switching theory and logical design, Wiley, 1968

[14] D. Lewin, Logical design of switching circuits, Nelson, 1968

[15] Raul Camposano, Wayne Wolf, High-Level VLSI Synthesis, Kluwer, 1991

[16] Claude Baron, Jean-Claude Geffroy, Gilles Motet, Embedded System Applications, Kulwer, 1997

[17] Jiang Xu, Wayne Wolf, A Design Methodology for Application-Specific Networks-on-Chip, ACM Transactions on Embedded Computing Systems, Vol. 5, No. 2, May 2006, Pp 263–280

[18] Kandár Tibor Hierarchikus rendszer szintű szintézis, doktori értekezés, 2007, BME

[19] CORPORATE Inc. Institute of Electrical and Electronics Engineers. IEEE Standard Description, Language Based on the VERILOG Hardware Description Language, 1364-1995. IEEE Standards, Office, New York, NY, USA, 1996.

[20] IEEE. IEEE Standard VHDL Reference Manual. IEEE, 1988.

[21] T. Kandár. Systematic VHDL code generation based on high level synthesis results. In INES2003, The 7th IEEE International Conference On Intelligent Engineering Systems, pages 716–720, Assiut, Luxor, Egypt, March 3–7 2003. ISBN:977.246.048.3, ISSN:1562-5850.

[21] Ahmed A. Jerraya, Jean Mermet, System-Level Synthesis, NATO science Series E, Applied Sciences – Vol. 357, Kluwer, 1999

[22] Zoltán Ádám Mann, András Orbán, Péter Arató (dec 2007) Finding optimal hardware/software partitions. Form. Methods Syst. Des. 31 (3) pp. 241–263. Kluwer Academic Publishers. Hingham, MA, USA.

[23] P. Arato, S. Juhasz, Z.A. Mann, A. Orban, D. Papp (sept. 2003) Hardware-software partitioning in embedded system design. In Intelligent Signal Processing, 2003 IEEE International Symposium on. pp. 197 - 202.

[24] Péter Arató, Zoltán Ádám Mann, András Orbán (jan 2005) Algorithmic aspects of hardware/software partitioning. ACM Trans. Des. Autom. Electron. Syst. 10 (1) pp. 136–156. ACM. New York, NY, USA.

[25] High-Level Synthesis Blue Book, 2010, Mentor Graphics Corporation

[26] Gergely Suba, Hierarchical pipelining of nested loops in high-level synthesis, 2013, Periodica Polytechnica, lektorálás és megjelenés folyamatban

[27] Gál Tibor: Interfésztechnikák, Szak kiadó, 2010, ISBN 978-963-9863-13-2

[28] USB 2.0 Specification, (http://www.usb.org/developers/docs)

[29] Andrew S. Tannembaum: Számítógép hálózatok, Panem, 1999

[30] BME-PS CAN plus protokoll specifikáció, BME-IIT, 2006

[31] Junwei Hou, Wayne Wolf: Process Partitioning for Distributed Embedded Systems, proceedings of the 4th International workshop on hardware/software co-design (Codes/CASHE '96), 0-8186-7243-9/96, 1996, IEEE

[32] Jiang Xu, Wayne Wolf, A Design Methodology for Application-Specific Networks-on-Chip, ACM Transactions on Embedded Computing Systems, Vol. 5, No. 2, May 2006, Pages 263-280