



---

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

---

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

PhD értekezés

Pilászy György

Témavezető: Dr. Arató Péter  
egyetemi tanár, a Magyar Tudományos Akadémia rendes tagja  
Konzulens: Dr. Móczár Géza

Budapest  
2013

## Tartalmi kivonat

Az értekezés a magas szintű szintézis elveinek és módszertanának vizsgálata alapján javaslatot tesz annak kiterjesztésére. A meglévő magas szintű szintézis algoritmusok a javasolt módosításokkal és módszerekkel tetszőleges komplexitású részegységekből álló elosztott (többprocesszoros) rendszerek struktúrájának szisztematikus szintézisére és optimalizálására is alkalmazhatókká válnak.

Az értekezés három problémakört vizsgál. A vizsgálatból leszűrhető tanulságokat, módszereket, módosításokat és eredményeket három tézisben foglalja össze.

Az értekezés első része az adatfolyam gráf egyes csomópontjai közötti kommunikációval foglalkozik. A kidolgozott módszer alkalmas a komplex jelfeldolgozó egységekbe integrált kommunikációs csatornák tulajdonságai és a továbbítandó információ mennyisége alapján, a kommunikáció elemi műveletként való figyelembe vételére a magas szintű tervezés során.

A második rész a lappangási idő növelésének hatását vizsgálja pipeline elven működő rendszerekben. A magas szintű tervező algoritmusok generálnak egy struktúrát és ehhez számítanak egy lappangási időt. A bemutatott vizsgálat és a javasolt módszer a kapott lappangási idő növelésével, - azonos átbocsájtási tényező esetén - csökkentheti a megvalósítás költségét a kedvezőbb allokáció révén.

A harmadik rész egy új allokációs módszert mutat be, amely képes figyelembe venni előre megadott kizárási és összevonási feltételeket is. A módszer egyik újdonsága, hogy lehetővé teszi feltételek megadását is az összevonás vagy kizárás előírására. A kidolgozott módszer a feltétel nélküli és feltételes előírások alapján maximális kompatibilitási osztályokat képez. Az allokációs módszer az így meghatározott, diszjunktá tett speciális zárt kompatibilitási osztályok meghatározásán alapul. A kidolgozott módszer jól használható biztonság-kritikus vagy speciális tervezési igényeket is figyelembe vevő alkalmazások esetén.

## **Abstract**

This dissertation is focusing to the extension of the high level synthesis methodology for making it applicable to systematic design of distributed (multiprocessing) systems consisted of processing units (components) with arbitrary complexity.

The new algorithms and the extended methodology are summarized in three theses. The first one provides a procedure for handling the communication buses integrated in the complex processing units. Based on an estimation method, the communication bus transfers are handled as additional nodes between the components with the execution time determined by the bus arbitration and transfer time. Since the final communication demand is determined by the decomposing (allocation) step of the synthesis. It is crucial to estimate the communication time realistically as early phase of the synthesis as possible. The method presented in the thesis is suitable for such a time estimation generalized from some frequently used interface systems.

The second thesis examines the effects of increasing the latency in pipeline distributed systems. The available high level synthesis tools handle the latency as an output parameter. The method proposed in this thesis may reduce the implementation cost by increasing the latency only and provides an impact assessment of the latency increment ranges calculated by a proposed algorithm.

The third thesis presents a new allocation method, which can take into consideration predefined separation and fusion conditions as well. The new feature of the presented method is that it allows the user to specify conditional fusions, too. The method is based on the search for a special closed disjoint cover by starting from the maximal compatibility classes. For example, the method can be applied in the design of safety-critical or easily testable systems.

## Nyilatkozat

Alulírott Pilászy György kijelentem, hogy ezt a doktori értekezést magam készítettem és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

A dolgozat bírálatai és a védésről készült jegyzőkönyv a későbbiekben, a dékáni hivatalban érhető el.

Budapest, 2013. április 14.

.....

## **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.

# Tartalomjegyzék

Tartalmi kivonat.....	2
Abstract.....	3
Nyilatkozat .....	4
Köszönetnyilvánítás .....	5
Tartalomjegyzék.....	6
Ábrák jegyzéke.....	8
Táblázatok jegyzéke.....	10
1. Bevezetés.....	11
1.1. Az értekezés célkitűzése.....	14
2. A PIPE tervező rendszer rövid bemutatása.....	15
2.1. A PIPE tervező rendszer moduljai.....	16
2.1.1. Az INPUT modul .....	16
2.1.2. A SCHEDULE modul .....	17
2.1.3. Az ALLOCATE modul .....	17
2.1.4. Az OUTPUT modul (további kiegészítő modulok) .....	17
2.2. A tervezési eredmények értékelése, összehasonlítása költségfüggvény alkalmazásával .....	18
3. A kommunikáció idejének figyelembe vétele.....	19
3.1. Kommunikációs idő meghatározása .....	19
3.1.1. Protokoll nélküli egyszerű interfészek .....	20
3.1.2. Egyszerű protokollt is alkalmazó interfészek.....	21
3.1.3. Többrétegű protokollt megvalósító interfész.....	23
3.2. Eljárás a kommunikáció idejének becslésére .....	27
3.2.1. A javasolt algoritmus lépései.....	28
3.2.2. A becslő algoritmus paraméterezése .....	31
3.3. Eredmények alkalmazása a HLS tervező rendszerek kiterjesztésében.....	33
3.3.1. A kommunikáció figyelembe vétele a szegmentálás során.....	34
3.3.2. A kommunikáció figyelembe vétele a szegmensgráfban .....	34
3.3.3. A kommunikáció figyelembe vétele a PIPE ütemező és allokáló algoritmusában .....	35
3.3.4. Mintaalkalmazás a kommunikáció figyelembevételére a PIPE tervező rendszerben.....	36

3.4. Első tézis .....	40
4. A lappangási idő növelésének hatása az egyes elemi műveletek újrakészíthetőségére .....	41
4.1. A költség csökkentéséhez szükséges szabályok .....	43
4.1.1. A lappangási idő növelés gyakorlati kivitelezése PIPE tervező rendszerben .....	44
4.1.2. Hatásvizsgálat (F1 mintafeladat) .....	44
4.1.3. Hatásvizsgálat (F2 mintafeladat) .....	52
4.2. A javasolt algoritmus a lappangási idő növelésének hatásvizsgálatára .....	62
4.3. Második tézis .....	66
5. Előre megadott kizárási és összevonási előírások figyelembevétele az allokáció során .....	67
5.1. Összevonási feltételek figyelembevétele .....	69
5.2. Kizárási feltételek lehetséges figyelembe vétele .....	70
5.2.1. A kizárási feltételek megadása allokációhoz .....	70
5.3. A javasolt új allokációs algoritmus .....	77
5.4. Kompatibilitási félmátrix automatikus generálása az ütemezés alapján .....	80
5.5. Harmadik tézis .....	85
6. Az eredmények alkalmazásának lehetőségei .....	86
7. Az értekezésben használt rövidítések jegyzéke .....	87
8. Irodalomjegyzék .....	88

## Ábrák jegyzéke

1.1. ábra A magas szintű szintézis fontosabb lépései [3].....	12
1.2. ábra HLS módszerek kiterjesztése elosztott pipeline rendszerek tervezésére .....	13
2.1. ábra Az EOG jelölései [3] .....	15
2.2. ábra A PIPE fontosabb moduljai [3] .....	16
3.1. ábra SPI adattovábbítás .....	20
3.2. ábra UART adattovábbítás .....	20
3.3. ábra I <sup>2</sup> C adattovábbítás (7 bites címzéssel).....	21
3.4. ábra CAN adat keretek felépítése [1], [2].....	24
3.5. ábra Bitbeszúrás az eredeti adatfolyamba.....	24
3.6. ábra A becslő algoritmus folyamatábrája.....	29
3.7. ábra Kommunikációs idők 1-32 byte továbbítása esetén 1 $\mu$ s bitidőt alkalmazva ..	30
3.8. ábra Kommunikációs csatorna megjelenítése az EOG-n.....	33
3.9. ábra Példa a szegmentálás során kiadódó szegmensgráfra a beszúrt kommunikációs csomópontokkal.....	34
3.10. ábra EOG hangforrás lokalizációhoz .....	37
3.11. ábra Allokált gráf kommunikáció nélkül.....	37
3.12. ábra EOG a beszúrt kommunikációs műveletekkel.....	38
3.13. ábra Allokált műveleti gráf hangforrás lokalizációhoz (R=130, L=260).....	39
3.14. ábra Allokált műveleti gráf hangforrás lokalizációhoz (R=130, L=360).....	39
4.1. ábra Példa egy pipeline rendszer ütemezésére .....	42
4.2. ábra Pipeline ütemezés megnövelt (L=32) lappangási idő esetén .....	42
4.3. ábra Ütemezés megnövelt újraindítás és lappangás (L=R=40) esetén .....	42
4.4. ábra F1 elemi műveleti gráfja .....	44
4.5. ábra F1 feladat költségfüggvényei .....	48
4.6. ábra Az F1 feladat elemi műveleti gráfja allokáció után (R=25, L=39) .....	49
4.7. ábra Az F1 feladat költségfüggvényei többfunkciós műveletvégzők alkalmazása esetén.....	51
4.8. ábra Az F2 feladat elemi műveleti gráfja.....	52
4.9. ábra Az F2 feladat költségfüggvényei.....	55
4.10. ábra Az F2 feladat költségfüggvényei többfunkciós műveletvégzők alkalmazása esetén.....	58
4.11. ábra Az F2 feladat költség függvényei komplex műveletvégzők használata esetén .....	60
4.12. ábra EOG (a) és allokált műveleti gráf (b).....	63

4.13 A lappangási idő növelő algoritmus folyamatábrája .....	64
5.1. ábra Összevonások és kizárások előírása a tervezés különböző fázisaiban .....	68
5.2. ábra A megvalósítandó feladat elemi műveleti gráfja.....	71
5.3. ábra Kompatibilitási félmátrix.....	72
5.4. ábra Az osztályok diszjunktta tételének lehetséges esetei (1) .....	73
5.5. ábra Az osztályok diszjunktta tételének lehetséges esetei (2) .....	74
5.6. ábra Az osztályok diszjunktta tételének lehetséges esetei (3) .....	74
5.7. ábra Az osztályok diszjunktta tételének lehetséges esetei (4) .....	74
5.8. ábra Allokált műveleti gráf (1. eset) .....	76
5.9. ábra Allokált műveleti gráf (4. eset) .....	76
5.10. ábra Allokációs algoritmus egy lehetséges folyamatábrája.....	78
5.11. ábra Az ütemezés végeredményének részlete .....	80
5.12. ábra A műveletek foglaltsága ( $R=25$ , $L=39$ ).....	80
5.13. ábra A cosinus egyenlethez tartozó kompatibilitási félmátrix .....	81
5.14. ábra A maximális kompatibilitási osztályokhoz tartozó lefedési tábla.....	82
5.15. ábra Módosított kompatibilitási félmátrix .....	82
5.16. ábra Allokáció a módosított kompatibilitási félmátrix szerint.....	83
5.17. ábra Kompatibilitási félmátrix A és D osztályok kizárásával .....	83
5.18. ábra Allokált műveleti gráf a B és C osztályokkal.....	84

## Táblázatok jegyzéke

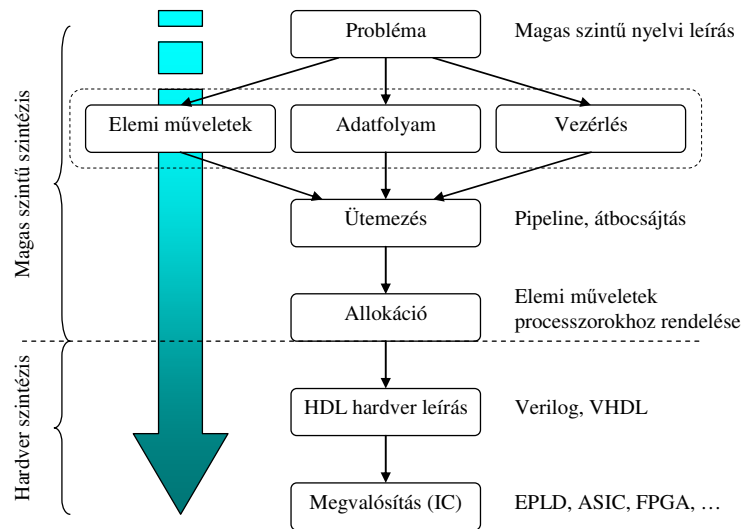
3.1. táblázat I <sup>2</sup> C kommunikációs idő becslése .....	21
3.2. táblázat I <sup>2</sup> C szabványos sebességek .....	21
3.3. táblázat I <sup>2</sup> C kommunikációs idők 7 bites címzés és különböző órajel frekvenciák esetén .....	22
3.4. táblázat A CAN üzenet maximális bitszáma .....	25
3.5. táblázat Egyetlen CAN üzenet bitszámának és továbbítási idejének becslése .....	26
3.6. táblázat A CAN üzenet továbbítás maximális ideje 1Mbit/s átviteli sebesség esetén .....	26
3.7. táblázat A becslő algoritmus paraméterei a bemutatott sínekre .....	30
3.8. táblázat Különböző interfészek lehetséges paraméterei a becslő algoritmushoz .....	31
3.9. táblázat Az elemi műveletek futási ideje a [6] alapján .....	36
3.10. táblázat A kommunikációs csomópont jellemzőinek meghatározása .....	37
3.11. táblázat A hangforrás lokalizáló feladat elemi műveleteinek paraméterei .....	38
3.12. táblázat Erőforrás használat az allokáció után .....	38
3.13. táblázat Erőforrás használat az allokáció után .....	39
4.1. táblázat Az F1 elemi műveleti gráf műveleteinek jellemzői .....	45
4.2. táblázat F1 feladat műveletvégző igényei különböző L=31, 37, 39 esetén .....	46
4.3. táblázat F1 feladat műveletvégző igényei különböző L= 43, 62 esetén .....	47
4.4. táblázat Az F1 feladat műveletvégző igénye többfunkciós műveletvégzők esetén .....	50
4.5. táblázat Az egyes műveletvégzők tulajdonságai .....	52
4.6. táblázat Az F2 feladat műveletvégző igényei L=48, 51, 54 esetén .....	53
4.7. táblázat Az F2 feladat műveletvégző igényei L= 60, 96 esetén .....	54
4.8. táblázat Az F2 feladat műveletvégző igénye többfunkciós műveletvégzők esetén L=48, 51, 54 .....	56
4.9. táblázat Az F2 feladat műveletvégző igénye többfunkciós műveletvégzők esetén L=60, 96 .....	57
4.10. táblázat Az F2 feladat műveletvégző igénye komplex műveletvégző esetén .....	59
4.11. táblázat Műveletvégzők és elemi műveletek tulajdonságai .....	63
5.1. táblázat A példában alkalmazott műveletvégzők tulajdonságai (e <sub>1</sub> ..e <sub>5</sub> ) .....	71
5.2. táblázat A kompatibilitási félmátrix javasolt jelölései .....	71
5.3. táblázat Lefedési tábla .....	73
5.4. táblázat Az egyes műveletek kizárása .....	81

## 1. Bevezetés

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 (particionálása) révén állítanak elő. A jelenlegi tervező eszközök közvetlenül az adatfolyam gráfból kiindulva kísérik meg az optimalizálást viszonylag egyszerű feldolgozó egységek feltételezésével, továbbá nem képesek kezelni a tervezés során kiadódó specifikációjú komplex feldolgozó egységeket.

A magas szintű szintézis főbb lépéseit az 1.1. ábra foglalja össze. A specifikált probléma leírását meg lehet adni – a feladattól függően - elemi műveletekkel, adatfolyam vagy vezérlési gráffal [22]. A tervezés következő fázisában történik a műveletek ütemezése, majd allokálása. Az allokáció során az egyes elemi műveleteket konkrét feldolgozóegységekhez rendelik.



**1.1. ábra A magas szintű szintézis fontosabb lépései [3]**

A magas szintű szintézis után kapott allokált gráf-reprezentációt a hardver szintézis során először valamilyen szabványos hardver leírássá (pl.: VHDL<sup>1</sup>) alakítják. A hardver leírásból, megfelelő tervező programokkal, előállítható a konkrét áramköri megvalósítás (pl.: EPLD<sup>2</sup>, ASIC<sup>3</sup>, FPGA).

Fontos kiemelni, hogy a későbbiekben ismertetett módszertan szerint a magas szintű szintézis végeredményét nem csak a kizárólag hardver generálására szolgáló hardver szintézishez lehet felhasználni. Az egyes komplex jelfeldolgozó egységek a hozzájuk rendelt feladatokat felépítésüktől függően szoftverrel is megvalósíthatják. A magas szintű tervező eszközökkel szisztematikus módon alakíthatók ki a pipeline elvű rendszerek, amelyekben az egyes részfeladatot megvalósító feldolgozó egységek átlapolt működése révén biztosítják a nagy feladatvégzési sebességet akkor is, ha a feladat-specifikációban nincs hatékonyan kihasználható párhuzamosság. A korszerű, programozható komplex jelfeldolgozó eszközök alkalmazása lehetővé teszi, hogy a pipeline rendszerek feldolgozó egységeit ezek feladatfüggő programozása révén valósítsuk meg.

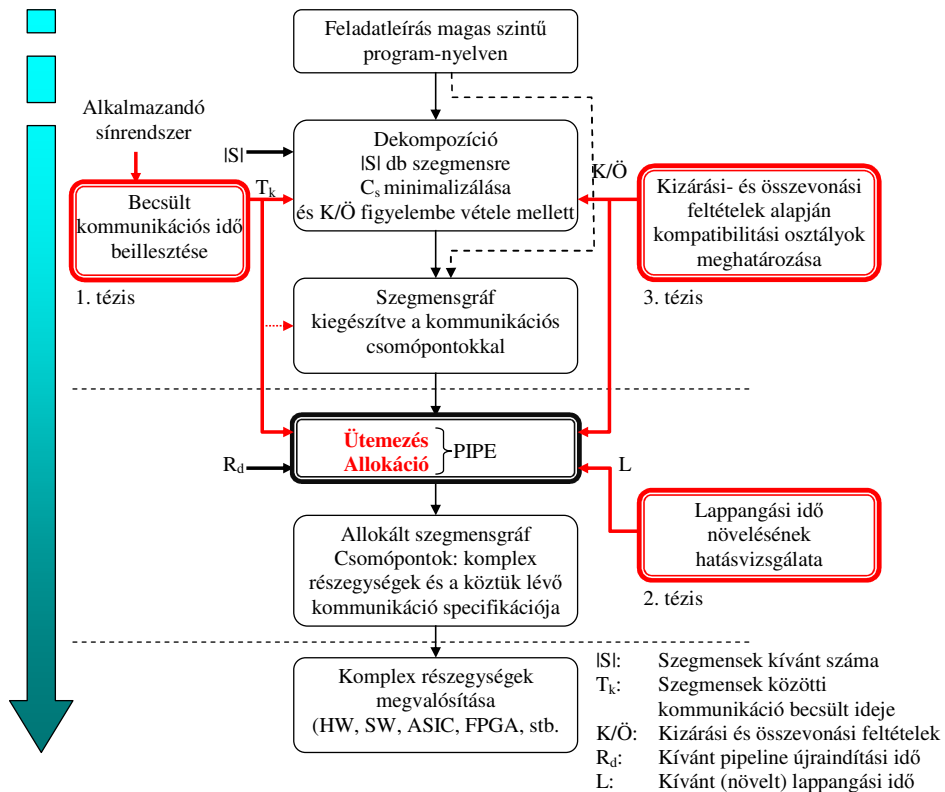
Az elosztott rendszerek tervezésének folyamatát - az említett módszertant, és a későbbiekben ismertetett kiegészítéseket, algoritmusokat felhasználva – az 1.2. ábra mutatja. A magas szintű nyelven leírt program partícionálására számos eljárás létezik [7, 30]. A korábbi tervezési módszerek a szegmensgráf meghatározása után általában allokációval majd konkrét realizációval folytatódtak. Az értekezésben bemutatott

<sup>1</sup> VHDL: VHSIC (Very High Speed Integrated Circuit) Hardware Description Language

<sup>2</sup> EPLD: Electrically Programmable Logic Device

<sup>3</sup> ASIC: Application Specific Integrated Circuit

módszerek ebbe a folyamatba beillesztik a HLS algoritmusokat, kiterjesztve azok alkalmazásának határait (az ábrán ezek a kiterjesztések vastag keretben láthatók). A tervezési folyamat során lehetőség nyílik - a dekompozíció kihagyásával - csak a HLS algoritmusok alkalmazására is, de ennek komoly hátránya, hogy hosszú program esetén meglehetősen nagy csomópontszámmal kell dolgozni, ami a HLS algoritmusok futásidejét jelentősen megnöveli. Ilyenkor természetesen nem a szegmensgráf, hanem a feladat leíró programból képzett elemi műveleti gráf a HLS rendszer (PIPE) bemenete. Ezt az esetet szaggatott nyíllal mutatja az 1.2. ábra.



1.2. ábra HLS módszerek kiterjesztése elosztott pipeline rendszerek tervezésére

### **1.1. Az értekezés célkitűzése**

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.

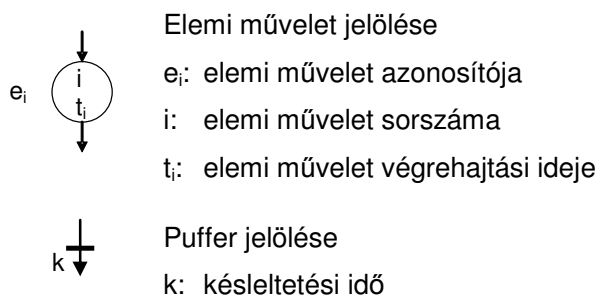
Ezen belül konkrétan:

- 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.: előírt kizárási és összevonási feltételek) szisztematikus módon kezelni és azokat figyelembe venni a tervezés különböző fázisaiban.

## 2.A PIPE tervező rendszer rövid bemutatása

Az alábbi fejezetben röviden ismertetésre kerül a PIPE tervező rendszer, mivel a további fejezetekben szereplő demonstrációs feladatok kidolgozását is ezzel a tervező eszközzel végeztem. A program első változata parancssorból volt használható [3], az elmúlt évek fejlesztései során készült hozzá grafikus kezelő felület is, amely lényegesen leegyszerűsíti az egyszerűbb feladatok bevitelét és az eredmények megjelenítését. Az eredeti rendszert és a továbbfejlesztéseket is a BME Irányítástechnika és Informatika Tanszékén készítették.

A feladatot egy úgynevezett elemi műveleti gráf (EOG<sup>4</sup>) formájában (ami tulajdonképpen egy speciális adatfolyam gráf (DFG<sup>5</sup>) kell specifikálni szöveges állományban [3]. A gráf leírása grafikus felület segítségével is előállítható. A különböző program modulok ezt a gráfot alakítják tovább tervezés folyamán. Az EOG egyes csomópontjai egy-egy - tovább nem osztható - **elemi műveletet** jelölnek. Ezen elemi műveletekből lehet több azonos típusú (pl.: összeadás, adott matematikai kifejezés, algoritmus részlet, stb.). A tervezés során a PIPE az elemi műveletek tényleges **műveletvégző** egységekhez rendeli. Az EOG-ban alkalmazott jelöléseket mutatja a 2.1. ábra.



2.1. ábra Az EOG jelölései [3]

A PIPE tervezőrendszer pipeline rendszerek szisztematikus tervezését támogatja, amelyeknek a legfontosabb jellemzői az R újraindítási (inicializálási) idő és az L lappangási idő. Az újraindítási idő megadja, hogy hány órajel ciklusonként kaphat új bemenő adatot a feldolgozó hálózat. A lappangási idő vagy látencia a teljes műveleti gráf végrehajtási ideje, vagyis a környezetből érkező bemeneti jel(ek) és az ennek hatására szolgáltatott kimeneti jel(ek) megjelenése közötti időt fejezi ki órajel ciklusok

<sup>4</sup> EOG: Elementary Operation Graph

<sup>5</sup> DFG: Data Flow Graph

darabszámában. A szakirodalomban használatos még az átbocsátási tényező fogalma, amely az újraindítási idő reciproka ( $1/R$ ). Az egyes műveletek végrehajtási ideje rokon fogalom a lappangással, tulajdonképpen nem más, mint az elemi műveletek lappangási ideje. A program indításakor paraméterként megadható a kívánt  $R$  újraindítási idő. Az optimalizálás történhet sebességre (pl.: a lehető legrövidebb, vagy kívánt értékű újraindítási időre), energiafogyasztásra vagy költségre. A program módosított változatában (erről a 2. tézis tárgyalásakor még lesz szó) lehetőség van a kiszámított minimális lappangásnál nagyobb lappangási idő megadására is.

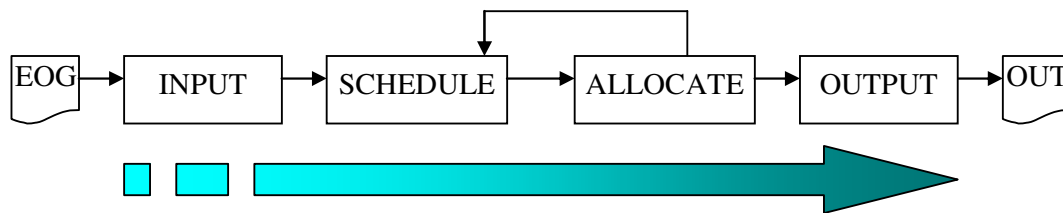
Ha egy elemi művelet végrehajtási ideje nagyobb, mint az előírt újraindítási idő, akkor az elemi művelet a szükséges darabszámban többszörözésre kerül [3]. A példákban alkalmazott vizsgálatok szempontjából kulcsfontosságú az ütemezés és az allokáció lépése, mert a végeredmény szempontjából fontos műveletvégző darabszámok alakulását ez a két lépés jelentősen befolyásolja.

Egy műveletvégző képes lehet több elemi művelet elvégzésére is (pl: ALU, amely képes összeadni, kivonni), de egyszerre mindig csak egy műveletet tud elvégezni.

A vezérlés feladata, hogy biztosítsa a megfelelő adatokat a megfelelő műveletvégzők bemenetén, majd kijelölve a végrehajtandó művelet típusát, elindítsa a feldolgozást. A művelet eredményét a vezérlő egység felügyeletével továbbítjuk a következő fokozathoz.

## 2.1. A PIPE tervező rendszer moduljai

A tervezőrendszer legfontosabb moduljait mutatja a 2.2. ábra.



2.2. ábra A PIPE fontosabb moduljai [3]

### 2.1.1. Az INPUT modul

Bemenetként a PIPE-nak szüksége van egy olyan gráf-leírásra, amely bemeneteket, kimeneteket, csomópontokat és éleket tartalmaz. Ez a leírás függvényhívás szerű, a csomópontok a függvényeket valósítják meg. A függvény nevében szerepel a csomópont által megvalósított elemi művelet neve. A függvény bemenő paraméterei a csomópont bemenetei, kimenő paramétere a csomópont kimenete. A gráf leírása tartalmazza még az elemi műveleteket is. Minden elemi

műveletnek két fő tulajdonsága van, a művelet neve és a végrehajtási ideje [3]. Az értekezés eredményei alapján javasolt fejlesztés alatt álló új változatban lehetőség lesz az egyes adatutak bitszámának megadására is. A bemeneti fájl előállítható szövegszerkesztővel, de a továbbfejlesztett változatban grafikus szerkesztő program is segíti a gráf megadást. Szintén fejlesztés alatt van egy olyan modul, amely magas szintű C nyelvű programkódból képes gráf leírást generálni a PIPE számára.

### 2.1.2. A SCHEDULE modul

Az ütemező modul három almodulból áll, amelyek a kívánt újraindítási idő beállítását (RESTART-modul), az adatutak szinkronizálását (SYNC-modul) és a teljes gráf ütemezését (SCHEDULE-modul) valósítják meg. A RESTART algoritmus bufferek közbeiktatásával és bizonyos műveletek többszörözésével állítja be a kívánt újraindítási időt (R). A SYNC modul az egyes adatutak szinkronizálását valósítja meg szinkronizációs bufferek beiktatása révén. A SCHEDULE modul az egyes csomópontokra meghatározott legkorábbi (ASAP<sup>6</sup>) és legkésőbbi (ALAP<sup>7</sup>) indítási időpontokon alapulva meghatározza a műveletek mobilitását, majd a mobilitási tartományon belül annak indítási idejét. Többféle ütemező algoritmus is implementálható, a jelenlegi változat egy erő-vezérelt (force-directed) ütemező algoritmust valósít meg.

### 2.1.3. Az ALLOCATE modul

Ez a modul a már ütemezett gráf elemi műveleteit rendeli konkrét műveletvégző egységekhez. A hozzárendelés konkurencia ellenőrzésen alapul. Egy algoritmus (CONCHECK) meghatározza az időben átlapolódó (konkurens) műveleteket. Az allokációs algoritmus az előre megadott processzor készlet és a konkurens / nem konkurens műveletek ismerete alapján konkrét feldolgozó egységekhez rendeli a műveleti gráf csomópontjait. Amennyiben nincsenek előre megadott műveletvégző egységek, úgy azok az allokáció révén kerülnek specifikálásra.

### 2.1.4. Az OUTPUT modul (további kiegészítő modulok)

A már allokált műveleti gráf bizonyos esetekben nagyszámú multiplexert és belső összeköttetést tartalmazhat. Ezeknek az adatkapcsolatoknak és multiplexereknek a darabszámát lehet optimalizálni a BUSRED és MUXRED algoritmusoknak a futtatásával. A két algoritmus részletes bemutatása a [18] PhD értekezésben található.

---

<sup>6</sup> ASAP: As Soon As Possible

<sup>7</sup> ALAP: As Least As Possible

Fejlesztés alatt állnak még olyan modulok, amelyek a végeredményként kapott gráf reprezentációt valamely szabványos hardver leíró nyelvre (VHDL [20], VERILOG [19]) alakítják tovább. A VHDL kód generálásának algoritmikus megvalósítása a [18] és [21] irodalomban részletesen megtalálható.

## **2.2. A tervezési eredmények értékelése, összehasonlítása költségfüggvény alkalmazásával**

A különféle beállításokkal kapott szimulációs eredmények összehasonlítását költségfüggvények segítségével végezzük. A PIPE-ban a megvalósítás költsége az allokációt követően ténylegesen felhasznált műveletvégzők összköltsége. Egy adott típusú műveletvégző költsége a darabszáma és a hozzá tartozó végrehajtási idő szorzataként áll elő. Ha többféle végrehajtási idejű művelet kerül egy műveletvégzőbe, akkor a legnagyobb időt tekintjük a műveletvégző végrehajtási idejének.

A bufferek, multiplexerek, buszok és a vezérlő hálózat ezekben a költség számításokban nulla költségként kerülnek figyelembevételre. Ezek a feltételek az elosztott rendszerek tervezésekor is elfogadhatóak. A tipikusan alkalmazott komplex jelfeldolgozók már tartalmazzák ezeket az elemeket. Ugyanakkor nem tartható az az egyszerűsítés, hogy a kommunikációt, illetve az ehhez szükséges időt sem veszi figyelembe. Az értekezés első tézise fog megoldás javasolni a buszok kommunikációs idejének (költségének) a figyelembe vételére.

A fentiek alapján az allokáció eredményét felhasználva egy adott megvalósítás költsége:

$$C = \sum_i C_i$$

Ahol  $C_i$  az egyes felhasznált műveletvégzők költsége:

$$C_i = n_i \cdot t_i$$

$t_i$ : Az  $M_i$  elemi műveletvégző által végrehajtható elemi műveletek végrehajtási idejeinek maximuma

$n_i$ : a  $t_i$  végrehajtási idejű műveletvégzők ténylegesen felhasznált darabszáma

$e_k$ : a  $M_i$  műveletvégzőbe allokált elemi művelet

$t_k$  az  $e_k$  elemi művelet végrehajtási ideje

A fenti egyszerű költségfüggvény alkalmazásával a különböző végrehajtási idejű műveletvégzők darabszámának alakulása jól szemléltethető, de adott esetben a rendszer követelményektől függően természetesen más költség függvény is használható.

### **3.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 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].

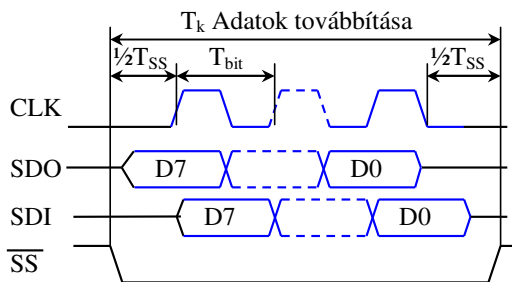
Az általam kidolgozott módszer szerint, a kommunikációs kapcsolat egy pótlólagos elemi műveleti csomóponttal ábrázolható, azonban ehhez szükség van a csomópont végrehajtási idejének ismeretére, vagy legalább megfelelő becslésére. Ha ugyanis alulbecsültük a kommunikáció tényleges idejét, akkor a végeredményként 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.

#### **3.1. Kommunikációs idő meghatározása**

Mikrokontrolleres, jelfeldolgozó kontrolleres környezetben a kis lábszám és más erőforrás korlátok miatt elterjedt a különböző soros kommunikációs vonalak alkalmazása. Az alábbiakban négy, gyakran beintegrált soros vonal kommunikációs idejének meghatározását mutatom meg.

### 3.1.1. Protokoll nélküli egyszerű interfészek

Az következőkben a jól ismert SPI<sup>8</sup> és UART<sup>9</sup> interfész kommunikációs idejét és bitszámát határozom meg. Ezen interfészek közös jellemzője, hogy az átvinni kívánt információ alapegysége tipikusan 8 bit. Ennek továbbításához szükséges időzítésen túl nem tartalmaz további előírást (protokollt). A 3.1. ábra és a 3.2. ábra mutatja az adattovábbítás folyamatát és ez alapján egyértelműen megállapítható a tényleges továbbítási idő ( $T_k$ ).



3.1. ábra SPI adattovábbítás

A kommunikáció ideje:

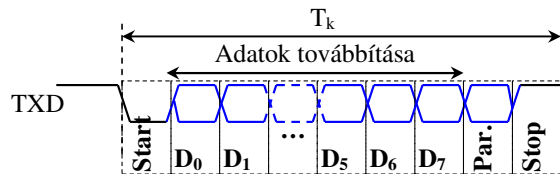
$$T_k = b \cdot T_{bit} + T_{SS} \quad (1)$$

$$T_{SS} \cong T_{bit} \quad (2)$$

$$T_k = (b + 1) \cdot T_{bit} \quad (3)$$

$$T_k = (N \cdot 8 + 1) \cdot T_{bit} \quad (5)$$

- $T_k$ : kommunikáció ideje  
 $T_{bit}$ : Egy bit továbbításának ideje (bit idő)  
 $T_{SS}$ : Indítás és leállítás ideje  
 $b$ : továbbított bitek száma  
 $N$ : Továbbítandó (hasznos) adatbájtok száma



3.2. ábra UART adattovábbítás

A kommunikáció ideje:

$$T_k = (b + 1 + p + s) \cdot T_{bit} \quad (6)$$

$$T_k = N \cdot (1 + 8 + p + s) \cdot T_{bit} \quad (7)$$

- $p$ : paritás vagy egyéb vezérlő bit száma (0 vagy 1)  
 $s$ : stop bitek száma (0, 1, 1.5, 2)

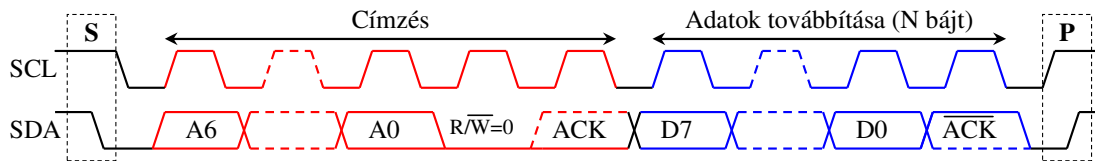
Ezen interfészek bitidejét meghatározó frekvencia tág határok között beállítható tipikus felső értéke SPI esetén < 20MHz, UART esetén < 10MHz. A konkrét értékek természetesen a fizikai közegtől is függenek, ennek figyelembe vétele már alkalmazási kérdés. Így az is, hogy pont-pont közötti, egy adó- több vevő, illetve több adó-több vevő sín alakítható ki.

<sup>8</sup> SPI: Serial Peripheral Interface

<sup>9</sup> UART: Universal Asynchronous Receiver Transmitter

### 3.1.2. Egyszerű protokollt is alkalmazó interfészek

Az átviendő információ kiegészítésre kerül az interfész típusától függő protokoll előírásai szerint. Ennek egyik elterjedt típusa az első sorban integrál áramkörök közötti (rövid távolságú) kommunikációra szolgáló I<sup>2</sup>C<sup>10</sup> interfész [8]. Az I<sup>2</sup>C adattovábbítás folyamatát hét bites címzés esetén mutatja a 3.3. ábra. A 3.3 ábra és az interfész specifikációja alapján meghatározható a kommunikáció ideje. Ez a kommunikációs idő a valóságban kis mértékben változhat, amennyiben a kommunikációban résztvevő megcímzett egység (slave) várakozásra kényszerítheti a küldő egységet (master). Az alkalmazható topológia tehát sín rendszerű.



3.3. ábra I<sup>2</sup>C adattovábbítás (7 bites címzéssel)

	7 bites címzés	10 bites címzés
Keretezés [bit]	2	2
Címzés [bit]	9	9 + 9
Adatok [bit]	N·9	N·9
Továbbítandó bitek száma [bit]	11+ N·9	20+ N·9
Becsült kommunikációs idő (T <sub>k</sub> )	C+(11+ N·9)·T <sub>bit</sub>	C+(11+ (N+1)·9)·T <sub>bit</sub>

3.1. táblázat I<sup>2</sup>C kommunikációs idő becslése

A [4] dokumentáció jelenleg elérhető változata öt különböző sebesség tartományt definiál, melyeket a 3.2. táblázat foglal össze.

Üzem mód	Max. sebesség [kbit/s]
Normál	100
Gyors	400
Gyors +	1000
Nagy sebességű	3400
Ultra nagy sebességű	5000

3.2. táblázat I<sup>2</sup>C szabványos sebességek

<sup>10</sup> I<sup>2</sup>C: Inter-IC Bus

**Megjegyzések:**

1. A „Nagysebességű” üzemmódok más hardver megoldást igényelnek, illetve a nagy sebességű átvitel inicializálása során 400Khz-en kapcsolat felépítés is történik. A bemutatott becslés ezt a C jelű konstans idővel ( $25\mu\text{s}$ ) veszi figyelembe.
2. Az Ultra nagy sebességű kivitel csak egyirányú (írás) adatátvitelre alkalmas és csak egy master kapcsolódhat a sínre [4].

1-8 bájt méretű üzenetek továbbítására meghatározott kommunikációs időket a 3.3. táblázat mutatja.

Adat (N byte)	100kHz	400kHz	1MHz	3,4MHz	5MHz
1	200 $\mu\text{s}$	50 $\mu\text{s}$	20 $\mu\text{s}$	30,9 $\mu\text{s}$	4 $\mu\text{s}$
2	290 $\mu\text{s}$	72,5 $\mu\text{s}$	29 $\mu\text{s}$	33,55 $\mu\text{s}$	5,8 $\mu\text{s}$
3	380 $\mu\text{s}$	95 $\mu\text{s}$	38 $\mu\text{s}$	36,2 $\mu\text{s}$	7,6 $\mu\text{s}$
4	470 $\mu\text{s}$	117,5 $\mu\text{s}$	47 $\mu\text{s}$	38,85 $\mu\text{s}$	9,4 $\mu\text{s}$
5	560 $\mu\text{s}$	140 $\mu\text{s}$	56 $\mu\text{s}$	41,5 $\mu\text{s}$	11,2 $\mu\text{s}$
6	650 $\mu\text{s}$	162,5 $\mu\text{s}$	65 $\mu\text{s}$	44,15 $\mu\text{s}$	14 $\mu\text{s}$
7	740 $\mu\text{s}$	185 $\mu\text{s}$	74 $\mu\text{s}$	46,8 $\mu\text{s}$	15,8 $\mu\text{s}$
8	830 $\mu\text{s}$	205,5 $\mu\text{s}$	83 $\mu\text{s}$	49,45 $\mu\text{s}$	17,6 $\mu\text{s}$

3.3. táblázat I<sup>2</sup>C kommunikációs idők 7 bites címzés és különböző órajel frekvenciák esetén

### 3.1.3. Többrétegű protokollt megvalósító interfész

Az összetettebb kommunikációs protokollok funkcióit az OSI<sup>11</sup> rétegmodell alapján csoportosíthatjuk. A szakirodalomban [29] részletesen megtalálható az egyes kommunikációs rétegek meghatározása, szerepük ismertetése. Az alábbi pontban egy olyan interfészt vizsgálok meg részletesebben, ahol az egyszerűbb adat keretezésen felül további (a kommunikációs idő meghatározása szempontjából fontos) kiegészítő információk továbbításával is számolni kell. Az ilyen és ehhez hasonló tulajdonságú kommunikációs perifériák az OSI rétegmodell fizikai rétegén túlmenően általában az adatkapcsolati réteget is hardveresen kezelik. Egy ilyen összetett protokollt megvalósító interfész a CAN<sup>12</sup>.

A CAN egy soros kommunikációs protokoll, amely támogatja a valós idejű elosztott irányítást, és akár 1 Mbit/s-os sebességet is lehetővé tesz. A protokollnak két fajtája létezik: a CAN V2.0A [1] **standard** formátum, amely 11 és a CAN V2.0B [2] **extended** formátum, amely 11+18 = 29 bites azonosítót használ az üzenetek továbbításához. Azok az eszközök, amelyek az extended formátumot is képesek használni, tudnak standard módban is kommunikálni, így azokkal az eszközökkel is együtt tudnak működni, amelyek csak a standard formátumot ismerik, de csak Standard módban.

A [1] és [2] irodalom alaposan részletezi a CAN üzenetek felépítését, így itt most csak az alkalmazás szempontjából fontos mélységig részletezem az üzenetek felépítését.

A CAN adatkeret felépítését a 3.4. ábra mutatja. Az előzőekben bemutatott három interfész esetében az üzenet hossza nem függött a továbbított adatbitek értékétől. A CAN rendszer által alkalmazott bitbeszúrás adatfüggősége miatt a kommunikációs időre csak becslés adható. Ehhez először meg kell becsülni a leghosszabb üzenet bitszámát.

#### A maximális CAN üzenethossz meghatározása

A bitbeszúrások miatt a CAN üzenet hossza a továbbított adatoktól függően változhat. A bitbeszúrás szabálya az, hogy 5 azonos értékű bit továbbítása után kötelező egy ellenkező értékű bitet beszúrni [1, 2]. A bitbeszúrás beszúrás működése a SOF<sup>13</sup> bittől kezdődően egészen a CRC<sup>14</sup> utolsó bitjéig engedélyezett.

---

<sup>11</sup> OSI: Open System Interconnection

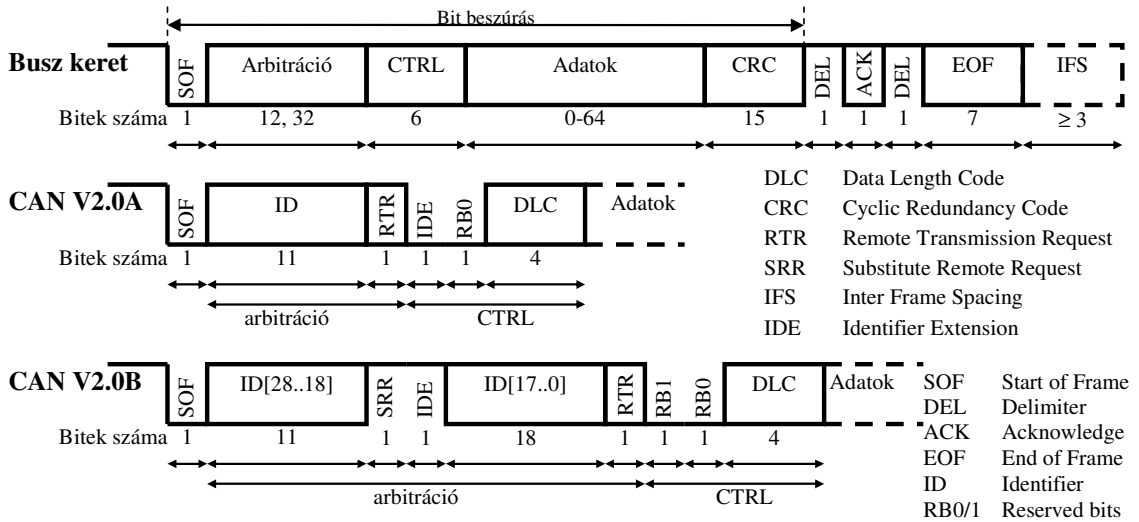
<sup>12</sup> CAN: Controller Area Network

<sup>13</sup> SOF: Start Of Frame

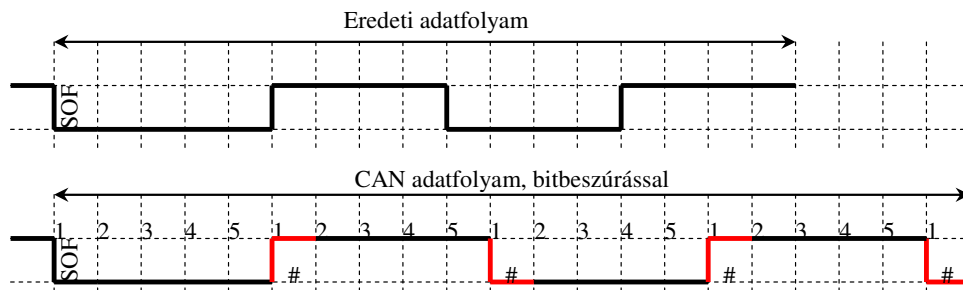
<sup>14</sup> CRC: Cyclic Redundancy Check

A legtöbb beszúrt bit darabszámának felső becsléséhez tekintsünk el attól, hogy néhány fix vezérlő bit értéke rögzített (pl.:IDE<sup>15</sup>, RB0, RB1<sup>16</sup>, stb.). A SOF miatt a keret biztosan 0 bittel kezdődik, ezért a teszt minta sorozat azonosítója is 0 bitekkel kezdődik. Öt darab 0 bit után (a SOF-al együtt) beszúrára kerül egy 1 értékű bit. Ezután folytassuk a teszt sorozatot 1 értékű bitekkel további 4db 1 értékű bit után ismét elértük a beszúráshoz szükséges 5 azonos értékű bitet, így most egy 0 kerül beszúrára. További 4db 0 után ismét 1 következik és így tovább. A gondolatmenet első 16 bitjét a 3.5. ábra mutatja. Az ábrán az idődiagram feletti számok a váltást követő azonos bit értékeket számolják.

A fenti jelsorozatról látható, hogy 16 bit továbbításához 4 bit beszúrára volt szükség. A „#” karakter a beszúrt biteket jelöli.



3.4. ábra CAN adat keretek felépítése [1], [2]



3.5. ábra Bitbeszúrással az eredeti adatfolyamba

<sup>15</sup> IDE: Identifier Extension

<sup>16</sup> RB0, RB1: Reserved Bits

Figyelembe véve a fejléc fix mezőinek bitszámát, a változó adathosszakot és a keret végén lévő üres részt (13bit) a 3.4. táblázat szerinti maximális bitszámok határozhatók meg a bitbeszúrásban résztvevő mezőkre.

	CAN2.0A	CAN2.0B
Fix mező	34 bit	54 bit
<b>N</b> [byte]	<b>Teljes üzenet bitszáma</b>	
1	53+13	78+13
2	63+13	88+13
3	73+13	98+13
4	83+13	108+13
5	93+13	118+13
6	103+13	128+13
7	113+13	138+13
8	123+13	148+13

3.4. táblázat A CAN üzenet maximális bitszáma

A beszúrára kerülő (Stuff) bitek számának (s) egy lehetséges felső becslését megkapjuk, ha az üzenet bitek számát elosztjuk 4-el, majd a kapott eredményt felfelé kerekítjük.

$$s = \left\lceil \frac{\text{adatbitek száma}}{4} \right\rceil = \left\lceil \frac{H + A}{4} \right\rceil \quad (8)$$

H: fix mezők bitszáma

A: adatmező biteinek száma

A = 8·N, ahol N az adatbájtok darabszáma

A teljes keret bitszáma (D) ezek alapján a következőképpen alakul:

$$D = H + A + s + E \quad (9)$$

E: a CRC utáni (nem bitbeszúrt) mezők bitszáma Z = 13.

#### Az üzenet továbbítási idő becslése

A CAN interfész sebességének ismeretében a maximális üzenet továbbítási idők becsülhetők. A becsléshez a korábban megbecsült üzenet hosszát meg kell szorozni az egy bit továbbításának idejével. A (9) összefüggést átrendezve:

$$D = H + \left\lceil \frac{H}{4} \right\rceil + E + A + \left\lceil \frac{A}{4} \right\rceil \quad (10)$$

Mivel az A mező mérete 0..8 bájt lehet, az A részbe bájtonként beszúrt bitek maximális száma 2. Az adatbájtok számát N-el jelölve az adatmező bitek számát a következőképpen írható:

$$A + \left\lceil \frac{A}{4} \right\rceil = N \cdot (8 + 2) = N \cdot 10 \quad (11)$$

Az előzőekben bemutatott eredményeket és a 3.4. táblázatot felhasználva az eredmények összefoglalása zárt alakban a 3.5. táblázatban látható. Amennyiben számítani kell az üzenet ismétlésére, úgy a meghatározott kommunikációs időt meg kell szorozni az ismétlések számával. Hibátlan kommunikáció esetén nincs ismétlés.

CAN	Üzenet bitek maximális darabszáma ( $N \leq 8$ )	Üzenet továbbítás maximális ideje $T_k$
2.0A	$56 + N \cdot 10$	$(56 + N \cdot 10) \cdot T_{bit}$
2.0B	$81 + N \cdot 10$	$(81 + N \cdot 10) \cdot T_{bit}$

3.5. táblázat Egyetlen CAN üzenet bitszámának és továbbítási idejének becslése

Például: 1Mbit/s adattovábbítási sebesség mellett a 3.4. táblázat a következőképpen írható át.

N [byte]	CAN2.0A	CAN2.0B
1	66 $\mu$ s	91 $\mu$ s
2	76 $\mu$ s	101 $\mu$ s
3	86 $\mu$ s	111 $\mu$ s
4	96 $\mu$ s	121 $\mu$ s
5	106 $\mu$ s	131 $\mu$ s
6	116 $\mu$ s	141 $\mu$ s
7	126 $\mu$ s	151 $\mu$ s
8	136 $\mu$ s	161 $\mu$ s

3.6. táblázat A CAN üzenettovábbítás maximális ideje 1Mbit/s átviteli sebesség esetén

### 3.2. *Eljárás a kommunikáció idejének becslésére*

Az előző pontban ismertetett kommunikációs interfészek 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 bitidőzés egész számú többszöröseként.

Mivel a jelen vizsgálat célja, hogy általános módszert javasoljon a kommunikáció idejének becsléséhez, ezért a vizsgált sínrendszerek alapján a következő általánosítható módszer javasolható:

#### **Bemenetek:**

- $N$ , továbbítandó adatbájtok száma
- $T_{\text{bit}}$ , bitidő (Ez bizonyos interfészek esetén rögzített érték, míg más interfészeknél adott tartományon belül változtatható paraméter)
- $B$ , bájtonkénti kiegészítő bitek maximális száma
- $K$ , keretenkénti további kiegészítő bitek maximális száma (pl.: keret fejléc, keret vége, ha van keretezés)
- $M$ , egy keretben maximálisan továbbítható adatbájtok száma (ennél nagyobb adatblokk esetén több keretre van szükség), 0, ha nincs ilyen korlátozás
- $J$ , egy keretben minimálisan továbbítható adatbájtok száma,  $J = M$ , ha rögzített a keretméret (tehát kevesebb bájt esetén kitöltő adatokkal van tele a kihasználatlan rész)
- $C$ , konstans időkorrekció

### 3.2.1. A javasolt algoritmus lépései

1. Keretek számának ( $F$ ) meghatározása

$$F = \begin{cases} \left\lceil \frac{N}{M} \right\rceil & \text{Ha } M > 0 \\ 1 & \text{Ha } M = 0 \end{cases}$$

2. Egy kereten belüli adatfüggő mezők ( $A$ ) és az utolsó keret adatfüggő mezői ( $A_u$ ) bitszámának meghatározása

$$A = \begin{cases} \max(N, J) \cdot (B + 8) & \text{ha } F = 1 \\ M \cdot (B + 8) & \text{ha } F > 1 \end{cases}$$

$$A_u = \begin{cases} 0 & \text{ha } F = 1 \\ \max(N - ((F - 1) \cdot M), J) \cdot (B + 8) & \text{Az utolsó keret esetén, ha } F > 1 \end{cases}$$

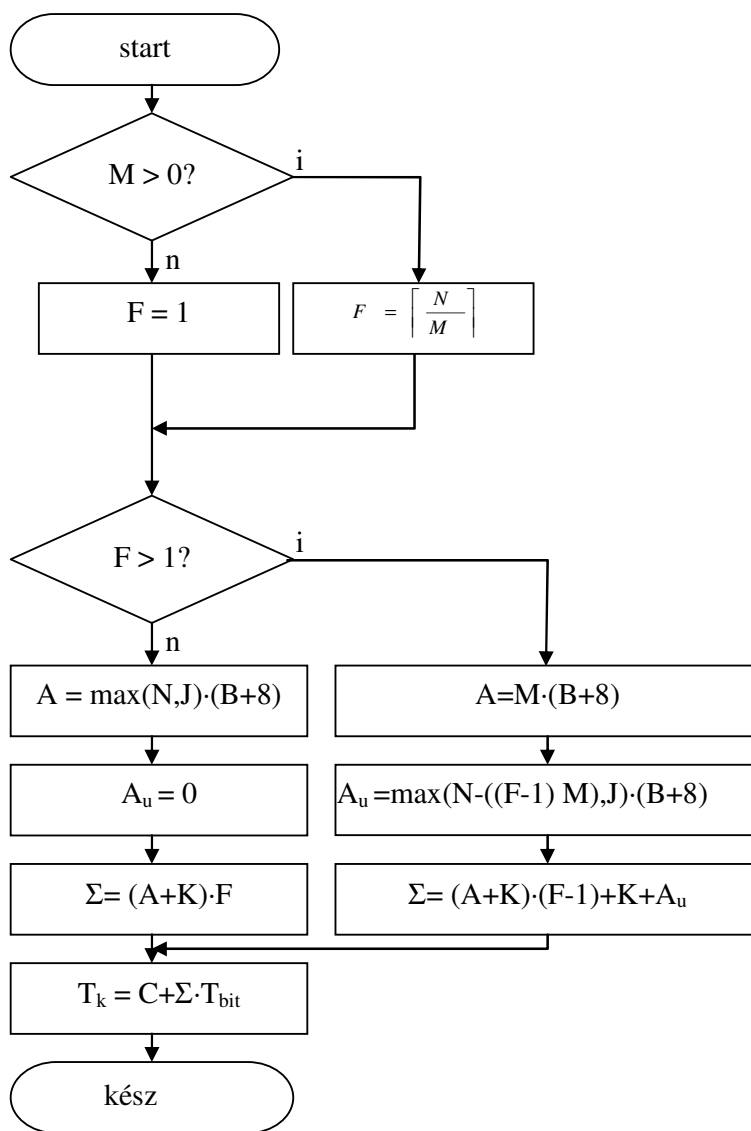
3. A teljes adatmennyiséghez tartozó továbbítandó bitszám meghatározása

$$\Sigma = \begin{cases} (A + K) \cdot F, & \text{ha } F = 1 \\ (A + K) \cdot (F - 1) + (K + A_u), & \text{ha } F > 1 \end{cases}$$

4. A becsült kommunikációs idő meghatározása

$$T_k = C + \Sigma \cdot T_{bit}$$

A becslési algoritmust a 3.6. ábra foglalja össze.



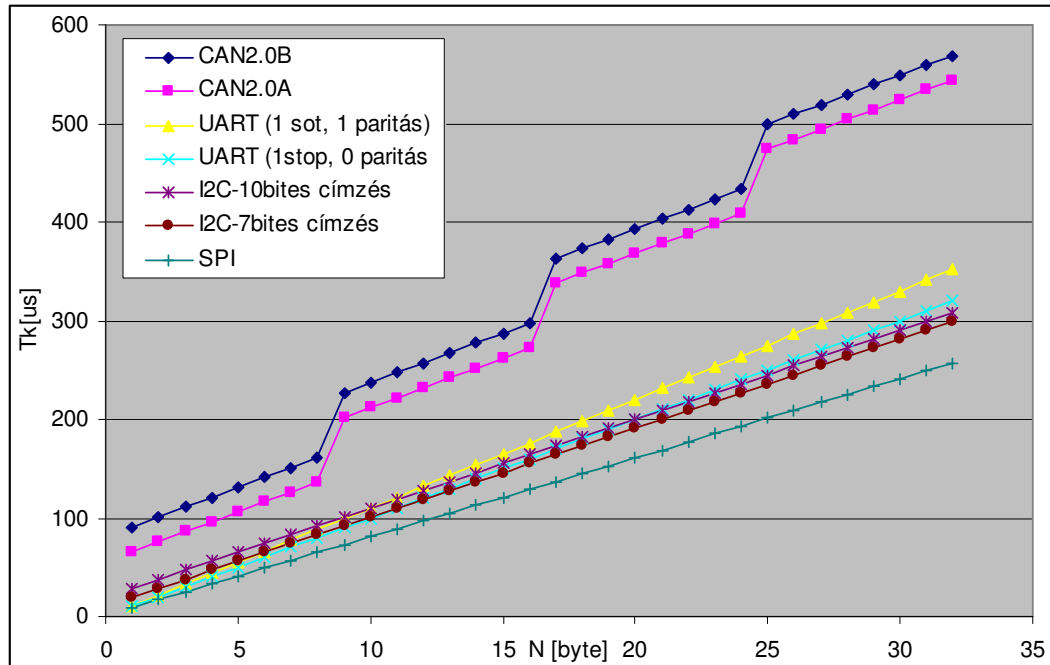
3.6. ábra A becsló algoritmus folyamatábrája

A bemutatott sínekre az algoritmus paramétereit a 3.7. táblázat mutatja.

Interfész	$T_k$ N≤8 esetén	C [μs]	B [bit]	K [bit]	M [byte]	J [byte]
SPI	$(N \cdot 8 + 1) \cdot T_{bit}$	0	0	1	0	0
I <sup>2</sup> C-7 bites címzés	$C + (N \cdot 9 + 11) \cdot T_{bit}$	0 vagy 25	1	10	0	0
I <sup>2</sup> C-10 bites címzés	$C + ((N + 1) \cdot 9 + 11) \cdot T_{bit}$	0 vagy 25	1	18	0	0
UART (1 stop, 0 paritásbit)	$N \cdot 10 \cdot T_{bit}$	0	2	0	0	0
UART (1 stop, 1 paritásbit)	$N \cdot 11 \cdot T_{bit}$	0	3	0	0	0
CAN2.0A	$(N \cdot 10 + 56) \cdot T_{bit}^*$	0	2	56	8	0
CAN2.0B	$(N \cdot 10 + 81) \cdot T_{bit}^*$	0	2	81	8	0

3.7. táblázat A becsülő algoritmus paramétereit a bemutatott sínekre

Az algoritmus eredményeit felhasználva a 3.7. ábra mutatja, hogy 1Mbit/s adatátviteli sebességet feltételezve hogyan alakulnak a 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.



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

### 3.2.2. A becslő algoritmus paraméterezése

Az előzőekben kidolgozott algoritmus paraméterezése a példaként vizsgált négy sínnel és néhány további (itt részletesen nem bemutatott) interfész esetében a 3.8. táblázat szerint alakul. Ez utóbbi interfészek ismertetése a [S4, 27, 28, 29, 30] irodalmakban részletesen megtalálható.

Interfész	T <sub>bit</sub>	C [μs]	B [Bit]	K [Bit]	M [Byte]	J [Byte]
SPI	50ns min.	0	0	1	0	0
I <sup>2</sup> C, (100kHz-1MHz) 7 bites cím	1-10μs	0	1	10	0	0
I <sup>2</sup> C, (100kHz-1MHz) 10 bites cím	1-10μs	0	1	18	0	0
I <sup>2</sup> C, High speed (3.4MHz), 7 bites cím	295ns	25	1	10	0	0
I <sup>2</sup> C, Ultra high speed (5MHz), 7 bites 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 <sup>17</sup> 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 <sup>18</sup> (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 <sup>19</sup> -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

3.8. táblázat Különböző interfészek lehetséges paraméterei a becslő algoritmushoz

<sup>17</sup> USB: Universal Serial Bus

<sup>18</sup> SATA: Serial Advanced Technology Attachment

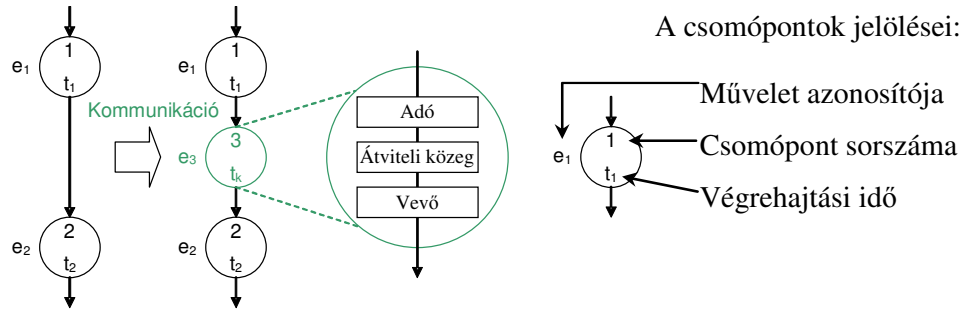
<sup>19</sup> PCI: Peripheral Component Interconnect

**Megjegyzések:**

- A kidolgozott módszer csak korlátozottan alkalmas többszintű (többrétegű) keretezést alkalmazó protokollok kommunikációs idejének becslésére.
- A J paraméter 0-tól különböző értékét akkor is célszerű kihasználni, ha fix keretméretekkel dolgozik a tervezendő rendszer. (például, ha előírás, hogy minden CAN üzenetnek fix 8 bájtos adatmezővel kell rendelkeznie, függetlenül az adatbájtok tényleges darabszámától, akkor  $J = 8$ )
- Ha igény van a kommunikációs keretek közötti szünet figyelembe vételére, akkor a K paramétert meg kell növelni a szünetnek megfelelő mértékben.
- Mivel a protokollok egy része többféle átviteli módot is képes támogatni, ezért megadtam az üzenet típus illetve átviteli mód nevét is az interfész neve mellett.
- Közös jellemzője ezeknek az interfészeknek, hogy mindegyik használ bájt szintű keretezést is az adatok továbbítása során.

### 3.3. **Eredmények alkalmazása a HLS tervező rendszerek kiterjesztésében**

A magas szintű tervezőrendszerek elemi műveleti gráffal reprezentálják a megoldandó feladatot. A 3.8. 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 grájára alkalmazva.



3.8. ábra Kommunikációs csatorna megjelenítése az EOG-n

Az  $e_3$  jelű kommunikációs elemi művelet végrehajtási (lappangási) ideje  $t_k$ . Ez tulajdonképpen az az idő, ami alatt az információ a feltételezett csatornán áthalad. Az előző pontokban meghatározott kommunikációs idő ( $T_k$ ) azonban a sebességtől függ, ezért azt át kell transzformálni a HLS tervező eszköz által alkalmazott időzítési rendszerbe. A kommunikációt jellemző időegység a  $T_{\text{bit}}$  bitidő illetve a  $T_k$  kommunikációs idő, míg a HLS rendszer alapütemezése a  $T$  órajel periódusidő. Ezek alapján a két időtartomány átskálázható a következő összefüggés szerint:

$$t_k = \frac{T_k}{T} \quad (12)$$

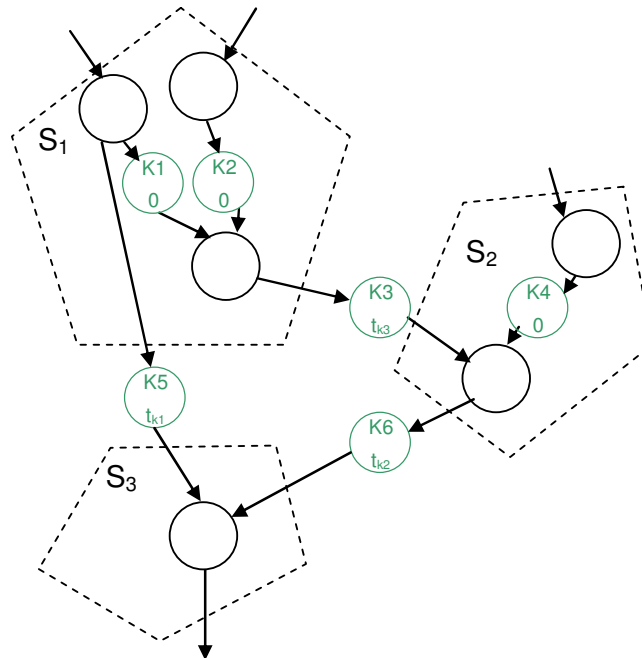
A (12) azt fejezi ki, hogy a HLS rendszerben a kommunikáció hány ciklusig tart.

A fizikai megvalósítás során a kommunikációs csatorna adója és vevője az adó és vevő oldalon lévő csomópontnak fogja részét képezni, csak a modell ábrázolja külön elemi műveletként. A kommunikáció lebonyolítása során természetesen mindkét csomópontnak foglalkoznia kell az adatátvitel kezelésével, de ezt a modellben a  $t_k$  paraméterben koncentrálni vesszük figyelembe. A vevő oldalon a kommunikációt követő elemi művelet ütemezése már a kommunikációs időt figyelembe véve, annak befejezését követően történik.

Az 1.2. ábra alapján a tervezés során három különböző lépés során is figyelembe vehetjük a kommunikációt. A három eset a következő:

### 3.3.1. A kommunikáció figyelembe vétele a szegmentálás során

Első lépésként valamennyi elemi művelet közé be kell iktatni a kívánt kommunikációs csomópontot. Amennyiben nincs szükség a kommunikációs csatornára akkor annak végrehajtási idejét nullára kell állítani és a továbbiakban a csomópont akár el is hagyható. Ennek eldöntése a feladat dekompozíció során a szegmensgráfokat létrehozó algoritmus feladata. Ennek egy lehetséges támogatása az 5. fejezetben kifejtett módszer. Egy ilyen szegmensgráfra mutat példát a 3.9. ábra. Az értekezésben ennek megvalósításával nem foglalkozom. Feltételezem, hogy ezek a szegmensgráfok rendelkezésre állnak.



3.9. ábra Példa a szegmentálás során kiadódó szegmensgráfra a beszűrt kommunikációs csomópontokkal

### 3.3.2. A kommunikáció figyelembe vétele a szegmensgráfban

Amennyiben a 3.3.1. megvalósításra került, akkor a szegmensgráf tartalmazza a kommunikációs csomópontokat minden olyan elemi művelet között, amely más szegmensben lévő elemi művelethez kapcsolódik. Amennyiben nem áll rendelkezésre, akkor ezt ebben a lépésben kell beszűrni a fentiek szerint. Ennek megoldására a 3.3.4-ben bemutatásra kerülő hangforrás lokalizáló struktúra optimalizálása mutat példát.

### **3.3.3. A kommunikáció figyelembe vétele a PIPE ütemező és allokáló algoritmusában**

A PIPE a bemenetén kapott EOG-ban már azt a szegmensgráfot kapja, amelybe a kommunikációs csomópontok már szerepelnek.

Elvileg lehetőség van az allokációt követően is beszúrni kommunikációs csomópontokat (amennyiben a szegmensgráf ezt nem tartalmazta), azonban ez felborítja az ütemezést, ezért a beszúrás követően újra kell szinkronizálni, majd ellenőrizni, hogy a kapott megoldás megfelel-e az elvárt kritériumoknak. A továbbiakban azt feltételezem, hogy a PIPE már a szükséges kommunikációs csomópontokat is tartalmazó EOG-t kap.

### 3.3.4. Mintaalkalmazás a kommunikáció figyelembevételére a PIPE tervező rendszerben

A kidolgozott módszer bemutatásához mintaalkalmazásként a [6]-ban bemutatott hangforrás lokalizáló struktúrába illesztettem CAN kommunikációs hálózatot az ott alkalmazott mikrokontrollerek közé. Ez a feladat egy létező elosztott rendszer, amelyben a mikrokontrollerek közötti kommunikációt speciálisan kialakított párhuzamos adatvonalakkal (24 bites párhuzamos csatornák) biztosították. Ennek kommunikációs idejét az ütemezés során elhanyagolták. A következőkben bemutatom, hogy egy szabványos kommunikációs csatornát felhasználva, az általam kidolgozott módszerrel hogyan lehet figyelembe venni a kommunikációs időt. A feladat optimalizálásához a PIPE HLS tervező rendszert használtam.

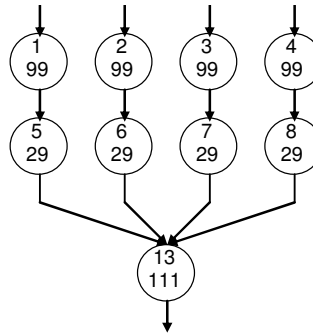
A feldolgozó hálózat minden bemenete egy mikrofon jelét vizsgálja. Az adatokat egy mikrokontroller programja transzformálja át ( $e_1..e_4$  jelű FFT-blokk), majd egy súlyozást követően ( $e_5, e_6, e_7, e_8$  jelű SC-blokk) kerül sor egy hipotézis vizsgálatra ( $e_{13}$  jelű HT-blokk). Az egyes eleminek tekintett műveletek főbb idő jellemzőit a [6] alapján először át kell skálázni órajel ciklusok számára a PIPE tervező rendszer számára a 3.9. táblázat szerint. Az újraindítási idő a [6] alapján  $R=130$ .

Elemi művelet megnevezése [6]	EOG azonosítója	Futási ideje	Átszámított órajel ciklusszám (PIPE)
FFT	$e_1, e_2, e_3, e_4$	99,2ms	99
SC	$e_5, e_6, e_7, e_8$	37,2ms	29*
HT	$e_{13}$	111ms	111

3.9. táblázat Az elemi műveletek futási ideje a [6] alapján

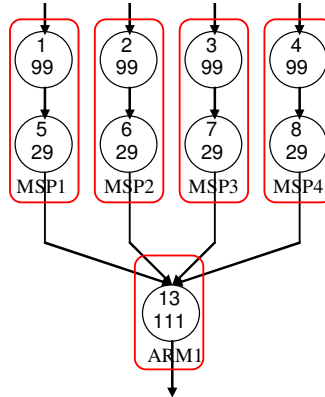
**\*Megjegyzés:** mivel az eredeti munkában található táblázatban megadott adatok és az ütemezésben megadott adatok között ellentmondás volt, az ütemezéshez felhasznált értéket vettem figyelembe.

A [6] alapján elkészített elemi műveleti gráfot a 3.10. ábra mutatja. Az elemi műveletek ebben a példában szoftverrel megvalósított rutinok, a tényleges megvalósítás során a műveletvégző egység(ek) mikrokontrollerek (ezek fogják majd végrehajtani a szoftveres rutinokat). Az, hogy melyik rutin melyik kontrollerbe kerüljön, illetve, hogy hány mikrokontroller kerüljön ténylegesen felhasználásra, az optimalizálás során dől el.



3.10. ábra EOG hangforrás lokalizációhoz

Először kommunikáció nélkül lefutattva a tervező rendszert ( $R=130$ ,  $L=245$ ) a 3.11. ábra szerinti eredményeket kaptuk az egyes műveletek ütemezése és alokálása után. Ez az eredmény megegyezik a cikkben kapott alokációs eredménnyel.



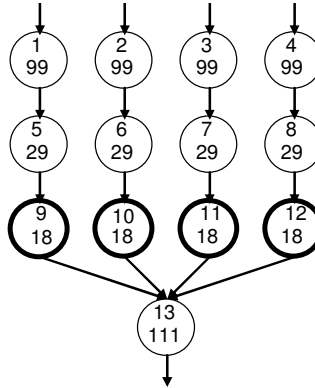
3.11. ábra Allokált gráf kommunikáció nélkül

A gráfot tovább alakítva, az  $e_9$ ,  $e_{10}$ ,  $e_{11}$ ,  $e_{12}$  jelű kommunikációs csomópontok (csatornák) beszúrásával a 3.12. ábra szerinti EOG-t kapjuk. A kommunikáció idejének becslését az előzőekben ismertetett eljárás szerint végezzük. A meghatározott paraméterek a 3.10. táblázat szerint alakultak.

A továbbítandó adatmennyiség [6]	$256 \cdot (2+2) = 1024$ bájt
A választott kommunikációs interfész	CAN2.0A, $1\mu\text{s}$ bitidővel
Keretek száma	$1024 / 8 = 128$
Az átviendő bitek száma	$\Sigma = 128 \cdot [80+56] = 17\,408$
A becsült kommunikációs idő	$T_k = 17,408\text{ms}$
A kommunikációs csomópont végrehajtási ciklusszáma	$t_k = 18$

3.10. táblázat A kommunikációs csomópont jellemzőinek meghatározása

A kommunikációval kiegészített EOG-t a 3.12. ábra mutatja. Az egyes elemi műveletek allokáció előtti darabszámát, ciklusidejét és azonosítóit a 3.11. táblázat foglalja össze.



3.12. ábra EOG a beszűrt kommunikációs műveletekkel

Elemi művelet eredeti megnevezése	Műveletvégző jelölése	EOG azonosítója	$t_i$ [órajel ciklus]	Elemi műveletek darabszáma
FFT	MSP	$e_1, e_2, e_3, e_4$	99	4
SC	MSP	$e_5, e_6, e_7, e_8$	29	4
<b>CAN</b>	<b>CAN</b>	<b><math>e_9, e_{10}, e_{11}, e_{12}</math></b>	<b>18</b>	<b>4</b>
HT	ARM	$e_{13}$	111	1

3.11. táblázat A hangforrás lokalizáló feladat elemi műveleteinek paramétereit

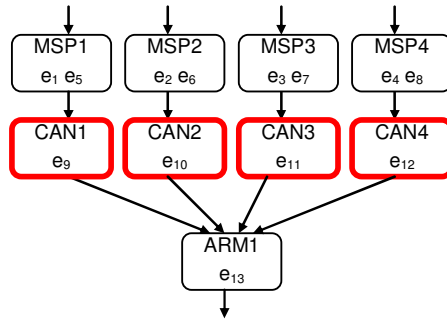
A tervezéskor megadott paraméterek:  $R=130$ ,  $L=260$

Az allokáció után az egyes műveletvégzők darabszámát a 3.13. táblázat mutatja.

Műv.végző	Elemi művelet	$t_i$	darab
MSP	FFT	99	4
	SC	29	
<b>CAN</b>	<b>CAN</b>	<b>18</b>	<b>4</b>
ARM	HT	111	1

3.12. táblázat Erőforrás használat az allokáció után

A végeredményként kapott allokált műveleti gráfot a 3.14. ábra mutatja. Érdekes megfigyelni, hogy az allokáció alapján négy külön kommunikációs csatorna került felhasználásra.



3.13. ábra Allokált műveleti gráf hangforrás lokalizációhoz (R=130, L=260)

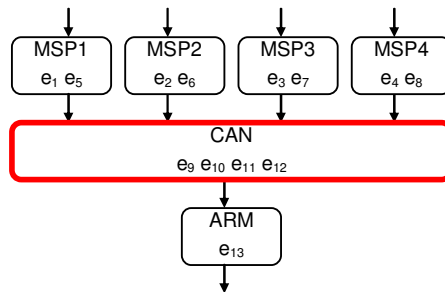
A tervezéskor megadott paraméterek módosítása után: R=130, L=360

Az allokáció után az egyes műveletvégzők darabszámát a 3.13. táblázat mutatja.

Műv.végző	Elemi művelet	$t_i$	darab
MSP	FFT	99	4
	SC	29	
<b>CAN</b>	<b>CAN</b>	<b>18</b>	<b>1</b>
ARM	HT	111	1

3.13. táblázat Erőforrás használat az allokáció után

A végeredményként kapott allokált műveleti gráfot a 3.14. ábra mutatja.



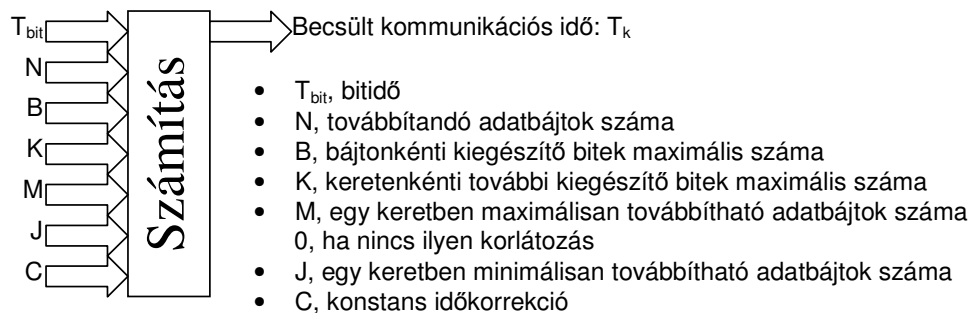
3.14. ábra Allokált műveleti gráf hangforrás lokalizációhoz (R=130, L=360)

Az eredményekből jól látszik, hogy a tervező rendszer a megadott paraméterek alapján CAN interfészeket be tudta úgy ütemezni, hogy egyetlen busz elég legyen a mikrokontrollerek közötti kommunikációra. Az ehhez szükséges L növelésének lehetőségét, illetve annak hatását részletesen a 4. fejezet mutatja be.

A fejezetben bemutatott módszert és algoritmust, illetve használatukat az 1. tézisben fogalmaztam meg.

### 3.4. Első 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], [S4], [S5] és [S6].

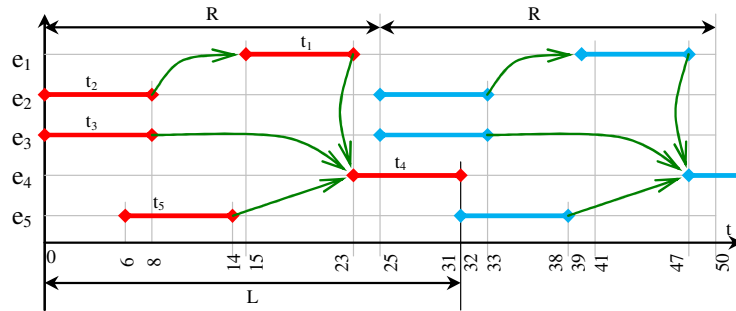
## 4.A lappangási idő növelésének hatása az egyes elemi műveletek újrafelhasználhatóságára

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 a 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.

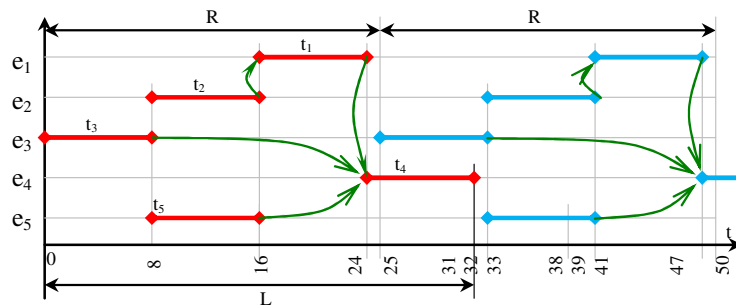
A probléma illusztrálására mutat példát a 4.1. ábra. Az ábrán piros szín jelöli az első adat-, kék a második adat feldolgozását. Az irányított nyilak az adatfüggőséget, a részeredmények továbbításának irányát ábrázolják. Az egyszerűség kedvéért legyenek azonosak az  $e_1 \dots e_5$  műveletek. Az  $e_2$  és  $e_3$  elemi műveletek a 4.1. ábra ütemezése alapján pontosan egyszerre indulnak és így egyszerre fejeződnek be. Emiatt semmiképp sem valósíthatók meg ugyanazzal a műveletvégző egységgel. Az átlapolás megszűnik, ha például az  $e_2$  műveletet  $t_3$  idővel később indítjuk egy újraindítási perióduson (R) belül. Ezt az esetet mutatja a 4.2. ábra. Megfigyelhető, hogy az  $e_2$  művelet késleltetése és az adatfüggőségek miatt a lappangási idő kiinduláskori értéke nem tartható.

A 4.1. ábra esetén az újraindítási idő,  $R = 25$  időegység, az egyes műveletvégzők végrehajtási ideje,  $e_i = 8$  időegység. Ebben az esetben egy újraindítási perióduson belül maximum háromszor lehet egy ilyen elemi műveletet végrehajtani ugyanazzal a műveletvégző egységgel. Ehhez azonban az adatfüggőségek miatt a lappangási időt (L) meg kell növelni, vagyis később jelenik meg az első eredmény a rendszer kimenetén.



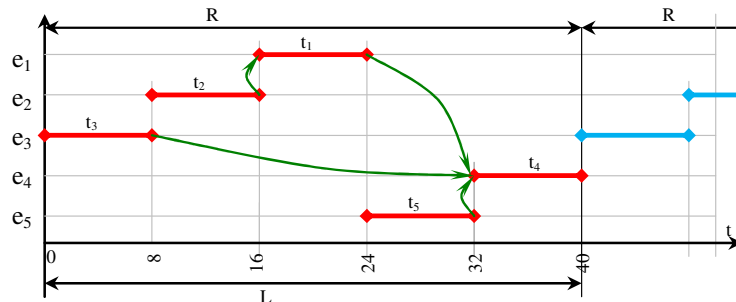
4.1. ábra Példa egy pipeline rendszer ütemezésére

Mivel az adott műveletvégző legfeljebb három elemi műveletet képes elvégezni a megadott újraindítás ( $R=25$ ) esetén, ezért legkevesebb két darab műveletvégző elég lehet a feladat megvalósításához az újraindítási idő megváltoztatása nélkül. A 4.2. ábra ütemezése alapján egy műveletvégzőbe összevonható például az  $e_1$ ,  $e_2$ ,  $e_3$  és egy másik műveletvégzőbe az  $e_4$ ,  $e_5$  elemi művelet, ehhez azonban a lappangási időt egy időegységgel meg kellett növelni ( $L=32$ ).



4.2. ábra Pipeline ütemezés megnövelt ( $L=32$ ) lappangási idő esetén

Ha lehetőség van az újraindítási időt is megnövelni, akkor elérhető a legkevesebb műveletvégzőt tartalmazó megoldás is, ami a példában egyetlen műveletvégzőt jelent. Ezt az esetet mutatja a 4.3. ábra. Fontos megjegyezni, hogy ez a szélsőséges eset az átlapolt működés hiánya miatt már nem pipeline működést eredményez.



4.3. ábra Ütemezés megnövelt újraindítás és lappangás ( $L=R=40$ ) esetén

#### 4.1. A költség csökkenéséhez szükséges 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 négyénél 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.

$$\sum_k t_k \leq R$$

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

$$k \cdot t_k \leq R$$

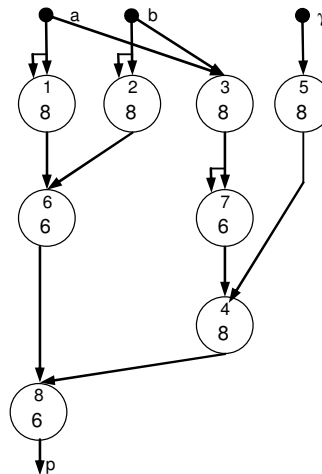
#### 4.1.1. A lappangási idő növelés gyakorlati kivitelezése PIPE tervező rendszerben

Mivel a PIPE tervező rendszer eredeti változata nem tette lehetővé a lappangási idő megadását, ezért először úgy oldottam meg a kívánt érték elérését, hogy az inputként megadott EOG-ba az eredeti gráffal párhuzamosan egy plusz csomópontot vettem fel, amelynek végrehajtási idejét a kívánt lappangási időnek megfelelő értékre állítottam. Ez az idő természetesen csak akkor van hatással a lappangási időre, ha nagyobb, mint az eredeti (beszúrás előtti) műveleti gráf lappangási ideje. Ennek a beszúrt plusz műveletnek a végrehajtási idejét módosítva lehet növelni a lappangási idő mértékét. A módszer általánosan is alkalmazható más HLS eszközök esetén is.

A későbbiek folyamán – éppen az értekezésben leírt módszer alkalmazása miatt - a PIPE ütemező modulja úgy módosult, hogy külső paraméterrel felülbíráható legyen az előző modulok által kiszámított lappangási idő. Ez lényegesen leegyszerűsítette a további szimulációs vizsgálatokat.

#### 4.1.2. Hatásvizsgálat (F1 mintafeladat)

A lappangási idő növelésének hatását elsőként, az F1 jelű feladat (cosinus egyenletet megoldó hálózat a [3] irodalom alapján) felhasználásával vizsgáltam. A vizsgált F1 hálózat elemi műveleti gráfját mutatja a 4.4. ábra.



4.4. ábra F1 elemi műveleti gráfja

Követve a [3] irodalom járulékos feltételezéseit, a 4.1. táblázat mutatja, hogy a rendelkezésre álló műveletvégző egységek ( $P_1$ ,  $P_2$ ,  $P_3$ ,  $P_4$ ) az elemi műveleti gráf mely elemi műveleteit képesek megvalósítani. Erre a pótlólagos feltételre csak az eredmények összehasonlítása miatt van szükség, egyébként komplex műveletvégzők alkalmazása esetén ez a feltételezés részben vagy egészben elhagyható.

Műveletvégző	Elemi művelet	$t_i$	Megvalósítandó elemi műveletek [db]
$P_1$	$e_1, e_2, e_3, e_4$	8	4
$P_2$	$e_5$	8	1
$P_3$	$e_6, e_7$	6	2
$P_4$	$e_8$	6	1

4.1. táblázat Az F1 elemi műveleti gráf műveleteinek jellemzői

A lappangási idő induló értéke:  $L = 31$ . Ezt a PIPE rendszerrel határoztam meg az eredeti gráfra  $R=10$  újraindítási idő esetén.

Mivel az elemi műveletek végrehajtási ideje: 8 és 6 időegység, ezért várhatóan legalább  $R>12$  idő felett jelentkezhet kedvező hatás, hiszen  $R>12$  esetén már van olyan műveletvégző, amely képes lehet több műveletet is elvégezni.

A vizsgált újraindítási idő tartomány:  $R = \{10, \dots, 49\}$

A lappangási időt a műveletvégzők végrehajtási idejének megfelelő lépésekben változtattam. A maximális lappangási időt a kiindulási érték kétszeresére választottam.

Így a vizsgált lappangási idők:  $L = \{31, 37, 39, 43, 62\}$

A 4.2 és 4.3 táblázatok összefoglalják az egyes lappangási és újraindítási időkhöz tartozó műveletvégzők darabszámát és a számított költségeket.

F	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	L=31					L=37					L=39
					C	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	C	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	C
10	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
11	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
12	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
13	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
14	4	1	2	1	58	4	1	1	1	52	4	1	1	1	52
15	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
16	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
17	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
18	3	1	2	1	50	4	1	2	1	58	4	1	2	1	58
19	3	1	2	1	50	4	1	2	1	58	4	1	2	1	58
20	3	1	2	1	50	4	1	2	1	58	3	1	2	1	50
21	3	1	2	1	50	4	1	2	1	58	3	1	2	1	50
22	3	1	2	1	50	3	1	1	1	44	3	1	2	1	50
23	2	1	2	1	42	2	1	1	1	36	2	1	2	1	42
24	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36
25	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36
26	2	1	1	1	36	2	1	2	1	42	2	1	1	1	36
27	2	1	1	1	36	2	1	2	1	42	2	1	1	1	36
28	2	1	1	1	36	2	1	1	1	36	2	1	2	1	42
29	2	1	2	1	42	2	1	1	1	36	2	1	2	1	42
30	2	1	2	1	42	2	1	1	1	36	2	1	1	1	36
31	3	1	2	1	50	2	1	1	1	36	2	1	1	1	36
32	2	1	2	1	42	2	1	1	1	36	2	1	1	1	36
33	2	1	2	1	42	2	1	2	1	42	2	1	1	1	36
34	2	1	2	1	42	2	1	2	1	42	2	1	1	1	36
35	2	1	2	1	42	2	1	2	1	42	2	1	2	1	42
36	2	1	2	1	42	2	1	2	1	42	2	1	2	1	42
37	2	1	2	1	42	2	1	2	1	42	2	1	2	1	42
38	2	1	2	1	42	2	1	2	1	42	2	1	2	1	42
39	2	1	2	1	42	2	1	2	1	42	2	1	2	1	42
40	2	1	2	1	42	2	1	2	1	42	2	1	1	1	36
41	2	1	2	1	42	2	1	1	1	36	2	1	1	1	36
42	2	1	2	1	42	2	1	1	1	36	2	1	1	1	36
43	2	1	2	1	42	2	1	1	1	36	2	1	1	1	36
44	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36
45	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36
46	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36
47	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36
48	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36
49	2	1	1	1	36	2	1	1	1	36	2	1	1	1	36

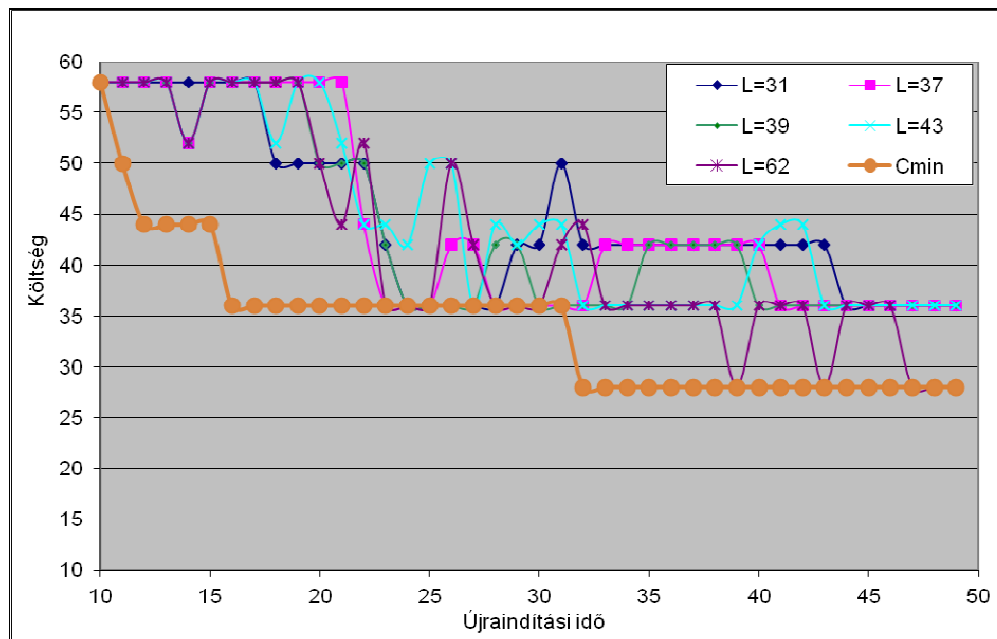
4.2. táblázat F1 feladat műveletvégző igényei különböző L=31, 37, 39 esetén

Elvi minimális darabszám

R	L=43					L=62									
	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	C	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	C	P <sub>1 min</sub>	P <sub>2 min</sub>	P <sub>3 min</sub>	P <sub>4 min</sub>	C <sub>min</sub>
10	4	1	2	1	58	4	1	2	1	58	4	1	2	1	58
11	4	1	2	1	58	4	1	2	1	58	3	1	2	1	50
12	4	1	2	1	58	4	1	2	1	58	3	1	1	1	44
13	4	1	2	1	58	4	1	2	1	58	3	1	1	1	44
14	4	1	1	1	52	4	1	1	1	52	3	1	1	1	44
15	4	1	2	1	58	4	1	2	1	58	3	1	1	1	44
16	4	1	2	1	58	4	1	2	1	58	2	1	1	1	36
17	4	1	2	1	58	4	1	2	1	58	2	1	1	1	36
18	4	1	1	1	52	4	1	2	1	58	2	1	1	1	36
19	4	1	2	1	58	4	1	2	1	58	2	1	1	1	36
20	4	1	2	1	58	3	1	2	1	50	2	1	1	1	36
21	4	1	1	1	52	3	1	1	1	44	2	1	1	1	36
22	3	1	1	1	44	4	1	1	1	52	2	1	1	1	36
23	3	1	1	1	44	2	1	1	1	36	2	1	1	1	36
24	2	1	2	1	42	2	1	1	1	36	2	1	1	1	36
25	3	1	2	1	50	2	1	1	1	36	2	1	1	1	36
26	3	1	2	1	50	3	1	2	1	50	2	1	1	1	36
27	2	1	1	1	36	2	1	2	1	42	2	1	1	1	36
28	3	1	1	1	44	2	1	1	1	36	2	1	1	1	36
29	2	1	2	1	42	2	1	1	1	36	2	1	1	1	36
30	3	1	1	1	44	2	1	1	1	36	2	1	1	1	36
31	3	1	1	1	44	2	1	2	1	42	2	1	1	1	36
32	2	1	1	1	36	3	1	1	1	44	1	1	1	1	28
33	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
34	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
35	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
36	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
37	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
38	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
39	2	1	1	1	36	1	1	1	1	28	1	1	1	1	28
40	2	1	2	1	42	2	1	1	1	36	1	1	1	1	28
41	3	1	1	1	44	2	1	1	1	36	1	1	1	1	28
42	3	1	1	1	44	2	1	1	1	36	1	1	1	1	28
43	2	1	1	1	36	1	1	1	1	28	1	1	1	1	28
44	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
45	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
46	2	1	1	1	36	2	1	1	1	36	1	1	1	1	28
47	2	1	1	1	36	1	1	1	1	28	1	1	1	1	28
48	2	1	1	1	36	1	1	1	1	28	1	1	1	1	28
49	2	1	1	1	36	1	1	1	1	28	1	1	1	1	28

4.3. táblázat F1 feladat műveletvégző igényei különböző L= 43, 62 esetén

Az eredmények ábrázolását egyetlen diagramon a 4.5. ábra mutatja. A legalsó görbe az adott R esetén elérhető minimális költséget jelöli. Megfigyelhető, hogy R=10 esetén a minimális költség megegyezik bármelyik vizsgált lappangási időhöz tartozó megoldás költségével. Ez azt jelenti, hogy R=10 esetén nem érhető el javulás a javasolt módszer alkalmazásával, míg R=14 esetén már igen. Ugyancsak látható, hogy R=39 mellett például a lappangást növelve csökkennek a megvalósítási költségek, míg L=62 lappangás mellett már a minimális költségű állapot is elérhetővé vált. Ugyanakkor azt is megfigyelhetjük, hogy jelen példa esetében R=23 és L=37 mellett az adott újraindításhoz tartozó optimális megoldást sikerült elérni (csak egyetlen  $M_1$  műveletvégzővel kell több a minimális megoldáshoz képest).



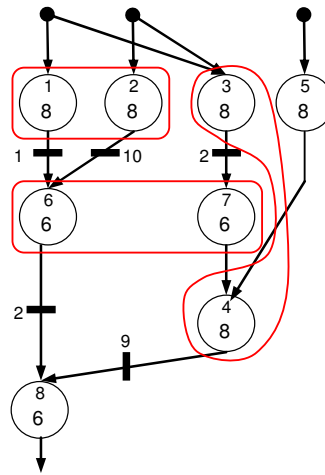
4.5. ábra F1 feladat költségfüggvényei

A  $P_1$  típusú műveletvégzők számának csökkenéséhez legalább akkora mértékben kell R értékén növelni, hogy legyen olyan  $P_1$ -hez tartozó elemi művelet, amelyik nem lapol át egy másikkal. Ehhez legalább  $R > 16$  újraindítási idő kell.

A 4.6. ábra mutatja, hogy például  $R = 25$ ,  $L=39$ -nél az  $e_1$  és  $e_2$  elemi művelet azonos műveletvégzőbe került, így csak 2db  $P_1$  típusú műveletvégző kell.

Egyes esetekben észrevehető, hogy a lappangási idő növelése nagyobb költségű megoldást eredményez, mint várnánk. Ennek oka az alkalmazott módszerben van, ugyanis nem módosítottuk a PIPE eredeti ütemezőjét, csak a lappangási időt növeltük meg. A megnövekedett szabadsági fokon belül a tényleges indítási időpontok rögzítését a PIPE algoritmusára bíztuk. Az allokáció az ütemezést követő lépés, így

egy valamilyen szempontból kedvező ütemezés nem biztos, hogy az allokáció szempontjából is az.

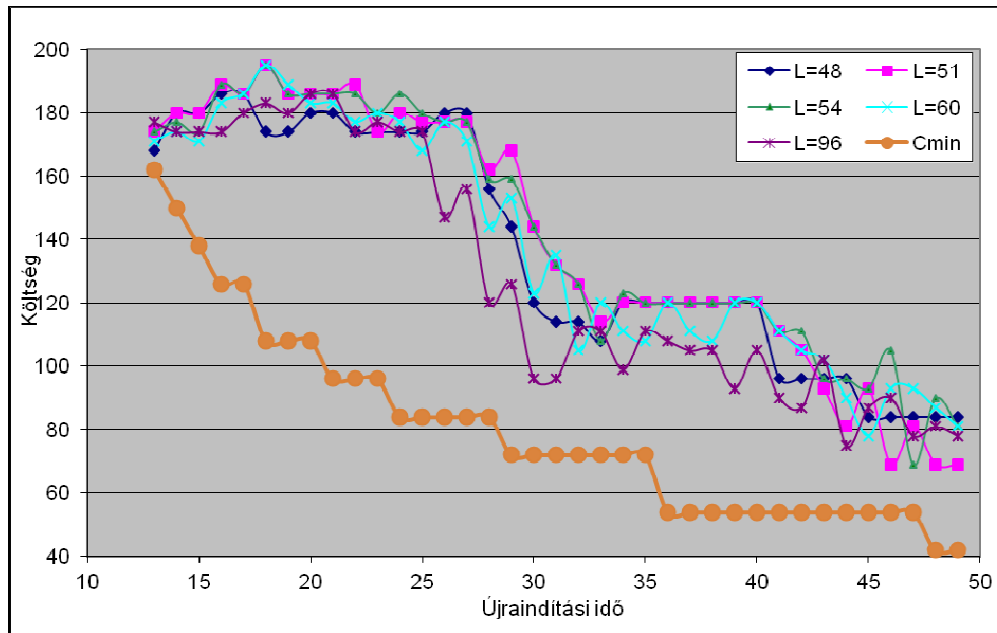


4.6. ábra Az F1 feladat elemi műveleti gráfja allokáció után ( $R=25$ ,  $L=39$ )

A lappangási idő növelésének előnyös hatása jobban érzékelhető, ha az egyes műveletvégzők többféle elemi művelet elvégzésére képesek, mert ekkor átkonfigurálhatók a pillanatnyi igényeknek megfelelően, így a műveletvégzők darabszáma tovább csökkenhet. Az előbbi példánál maradva növeljük a komplexitását a műveletvégzőknek. Tegyük fel, hogy rendelkezésünkre áll olyan  $P_1$  típusú műveletvégző, mely képes ellátni az  $e_1 \dots e_5$  műveleteket és van egy másik  $P_2$  típusú műveletvégző, amely képes az  $e_6, e_7, e_8$  elemi műveletek elvégzésére. Az elemi műveleti gráf ettől nem változik meg, azonban az allokáció és ütemezés során kedvezőbb eredmény is elérhető.

R	M <sub>1</sub>	M <sub>2</sub>	L=31			L=37			L=39			L=43			L=62			
			M <sub>1</sub>	M <sub>2</sub>	C	M <sub>1</sub>	M <sub>2</sub>	C	M <sub>1</sub>	M <sub>2</sub>	C	M <sub>1</sub>	M <sub>2</sub>	C	P <sub>1min</sub>	P <sub>2min</sub>	C <sub>min</sub>	
10	5	3	58	5	3	58	5	3	58	5	3	58	5	3	58	1	2	20
11	5	3	58	5	3	58	5	3	58	5	3	58	5	3	58	1	2	20
12	5	3	58	5	3	58	5	3	58	5	3	58	5	3	58	1	2	20
13	5	2	52	5	2	52	5	2	52	5	2	52	5	2	52	1	2	20
14	5	2	52	5	2	52	5	3	58	5	2	52	5	2	52	1	2	20
15	5	2	52	5	2	52	5	2	52	5	2	52	5	2	52	1	2	20
16	5	3	58	5	2	52	5	2	52	5	2	52	5	3	58	1	2	20
17	5	2	52	5	3	58	5	3	58	5	2	52	5	3	58	1	2	20
18	5	2	52	5	3	58	4	3	50	4	3	50	4	3	50	1	1	14
19	4	2	44	5	3	58	4	3	50	4	3	50	4	3	50	1	1	14
20	4	2	44	5	3	58	4	2	44	4	2	44	4	2	44	1	1	14
21	3	2	36	3	2	36	4	2	44	4	2	44	4	2	44	1	1	14
22	3	2	36	3	2	36	3	2	36	3	2	36	5	2	52	1	1	14
23	4	2	44	4	2	44	4	2	44	3	2	36	3	2	36	1	1	14
24	3	1	30	4	2	44	3	2	36	4	1	38	3	2	36	1	1	14
25	3	1	30	3	2	36	3	2	36	3	2	36	3	2	36	1	1	14
26	3	2	36	3	2	36	3	2	36	3	2	36	3	2	36	1	1	14
27	3	2	36	3	2	36	2	1	22	3	2	36	3	2	36	1	1	14
28	3	2	36	3	1	30	3	1	30	3	2	36	2	1	22	1	1	14
29	3	2	36	3	1	30	3	1	30	2	1	22	3	1	30	1	1	14
30	3	2	36	3	1	30	3	1	30	2	1	22	3	1	30	1	1	14
31	3	2	36	3	1	30	3	1	30	3	2	36	3	1	30	1	1	14
32	3	2	36	3	1	30	3	1	30	3	2	36	3	1	30	1	1	14
33	3	2	36	3	1	30	3	1	30	2	1	22	3	1	30	1	1	14
34	3	2	36	3	1	30	3	1	30	2	1	22	2	1	22	1	1	14
35	3	2	36	3	1	30	3	1	30	2	1	22	3	1	30	1	1	14
36	3	2	36	3	1	30	3	1	30	3	1	30	2	1	22	1	1	14
37	3	2	36	2	1	22	3	1	30	3	1	30	2	1	22	1	1	14
38	3	2	36	2	1	22	3	1	30	3	1	30	2	1	22	1	1	14
39	3	2	36	3	1	30	2	1	22	3	1	30	2	1	22	1	1	14
40	3	2	36	3	1	30	2	1	22	3	1	30	2	1	22	1	1	14
41	3	2	36	2	1	22	2	1	22	3	1	30	2	1	22	1	1	14
42	3	2	36	2	1	22	2	1	22	3	1	30	2	1	22	1	1	14
43	3	2	36	2	1	22	2	1	22	2	1	22	2	1	22	1	1	14
44	3	2	36	2	1	22	2	1	22	2	1	22	2	1	22	1	1	14
45	3	2	36	2	1	22	2	1	22	2	1	22	2	1	22	1	1	14
46	3	2	36	2	1	22	2	1	22	2	1	22	2	1	22	1	1	14
47	3	2	36	2	1	22	2	1	22	2	1	22	2	1	22	1	1	14
48	3	1	30	2	1	22	2	1	22	2	1	22	2	1	22	1	1	14
49	3	1	30	2	1	22	2	1	22	2	1	22	2	1	22	1	1	14

4.4. táblázat Az F1 feladat műveletvégző igénye többfunkciós műveletvégzők esetén

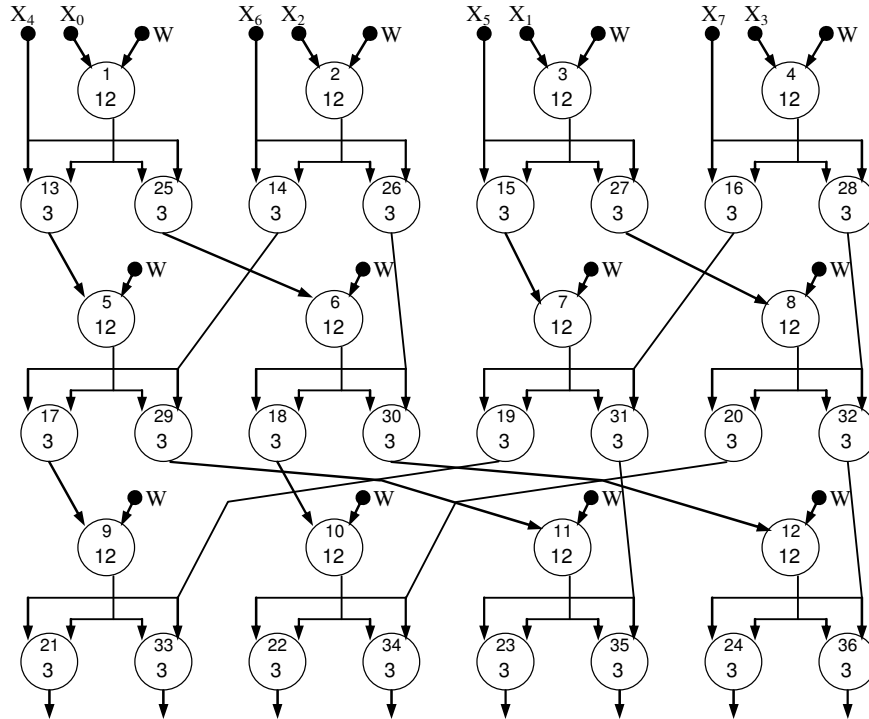


**4.7. ábra** Az F1 feladat költségfüggvényei többfunkciós műveletvégzők alkalmazása esetén

A 4.4. táblázat és a 4.7. ábra grafikonja jól mutatja, hogy már kismértékű lappangási idő növelés esetén is jelentősen lecsökkent a szükséges műveletvégzők darabszáma. Amíg a négy különféle műveletvégző esetén (előző példa)  $L=62$ ,  $R=39$  esetén az elérhető legkisebb költség 28 egység volt, addig most kisebb lappangás esetén,  $L=39$  és  $R=27$  esetén a költség már 22 egység vagyis ebben az esetben egy gyorsabban újraindítható, kisebb lappangású és olcsóbb megoldást kaptunk az univerzális műveletvégzők jobb kihasználtsága révén.

### 4.1.3. Hatásvizsgálat (F2 mintafeladat)

A lappangási idő növelésének hatását másodikként, az F2 jelű feladat (nyolcpontos FFT algoritmus a [3] irodalom alapján) felhasználásával vizsgáltam. A vizsgált F2 hálózat elemi műveleti gráfját mutatja a 4.8. ábra.



4.8. ábra Az F2 feladat elemi műveleti gráfja

Követve a [3] irodalom járulékos feltételezéseit, a 4.5. táblázat mutatja, hogy a rendelkezésre álló műveletvégző egységek ( $P_1$ ,  $P_2$ ,  $P_3$ ) az EOG mely elemi műveleteit képesek megvalósítani. Erre a pótlólagos feltételre csak az eredmények összehasonlítása miatt van szükség, egyébként komplex műveletvégzők alkalmazása esetén ez a feltételezés részben vagy egészben elhagyható.

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

4.5. táblázat Az egyes műveletvégzők tulajdonságai

A lappangási idő indulóértéke:  $L = 48$ . Ezt a PIPE határozta meg az eredeti gráfra  $R=13$  újraindítási idő esetére.

A vizsgált újraindítási idő tartomány:  $R = \{13, \dots, 49\}$

A vizsgált lappangási idők:  $L = \{48, 51, 54, 60, 96\}$

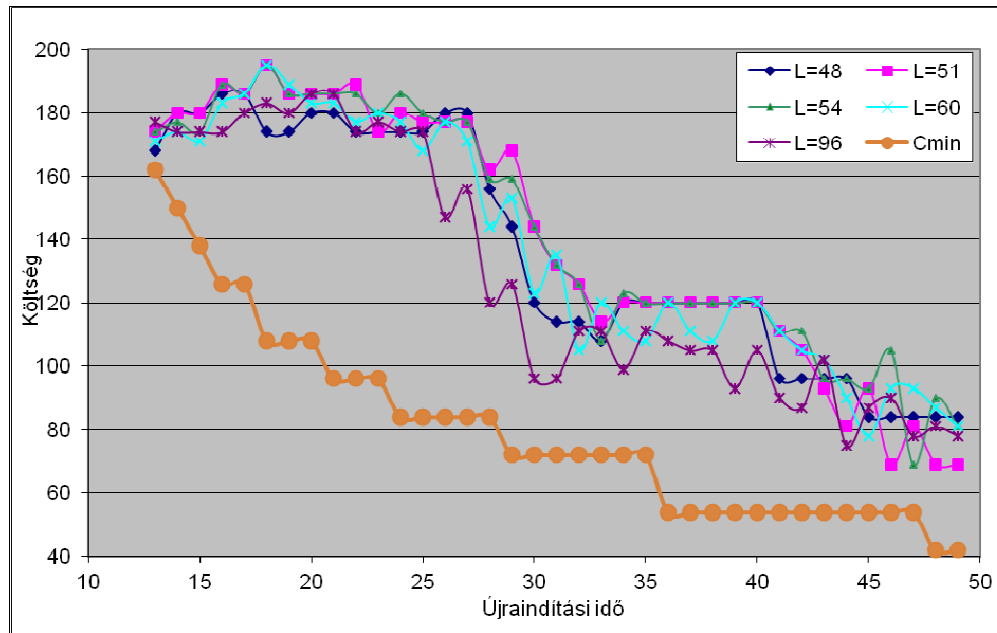
A 4.6. táblázat és a 4.7. táblázat összefoglalja az egyes időkhöz tartozó műveletvégzők darabszámát és a számított költségeket.

$R$	$P^1$	$P^2$	$P^3$	L=48				L=51				L=54			
				$C$	$P^1$	$P^2$	$P^3$	$C$	$P^1$	$P^2$	$P^3$	$C$	$P^1$	$P^2$	$P^3$
13	12	4	4	168	12	5	5	174	12	5	5	174			
14	12	6	6	180	12	6	6	180	12	5	6	177			
15	12	6	6	180	12	6	6	180	12	5	5	174			
16	12	7	7	186	12	7	8	189	12	7	8	189			
17	12	7	7	186	12	7	7	186	12	7	7	186			
18	12	5	5	174	12	9	8	195	12	9	8	195			
19	12	5	5	174	12	8	6	186	12	8	6	186			
20	12	6	6	180	12	7	7	186	12	7	7	186			
21	12	6	6	180	12	6	8	186	12	6	8	186			
22	12	5	5	174	12	7	8	189	12	7	7	186			
23	12	5	5	174	12	5	5	174	12	6	6	180			
24	12	5	5	174	12	6	6	180	12	6	8	186			
25	12	5	5	174	12	5	6	177	12	6	6	180			
26	12	6	6	180	12	5	6	177	12	5	6	177			
27	12	6	6	180	12	5	6	177	12	5	6	177			
28	10	6	6	156	11	5	5	162	10	6	7	159			
29	9	6	6	144	11	6	6	168	10	6	7	159			
30	7	6	6	120	9	6	6	144	9	6	6	144			
31	7	5	5	114	8	6	6	132	8	6	6	132			
32	7	5	5	114	8	5	5	126	8	5	5	126			
33	7	4	4	108	7	5	5	114	6	6	6	108			
34	8	4	4	120	8	4	4	120	8	5	4	123			
35	8	4	4	120	8	4	4	120	8	4	4	120			
36	8	4	4	120	8	4	4	120	8	4	4	120			
37	8	4	4	120	8	4	4	120	8	4	4	120			
38	8	4	4	120	8	4	4	120	8	4	4	120			
39	8	4	4	120	8	4	4	120	8	4	4	120			
40	8	4	4	120	8	4	4	120	8	4	4	120			
41	6	4	4	96	7	4	5	111	7	4	5	111			
42	6	4	4	96	7	3	4	105	7	4	5	111			
43	6	4	4	96	6	3	4	93	6	4	4	96			
44	6	4	4	96	5	3	4	81	6	4	4	96			
45	5	4	4	84	6	4	3	93	6	3	4	93			
46	5	4	4	84	4	4	3	69	7	3	4	105			
47	5	4	4	84	5	4	3	81	4	3	4	69			
48	5	4	4	84	4	4	3	69	6	3	3	90			
49	5	4	4	84	4	4	3	69	5	4	3	81			

4.6. táblázat Az F2 feladat műveletvégző igényei L=48, 51, 54 esetén

r	P <sup>1</sup>	P <sup>2</sup>	P <sup>3</sup>	L=60				L=96				
				C	P <sup>1</sup>	P <sup>2</sup>	P <sup>3</sup>	C	P <sub>1min</sub>	P <sub>2min</sub>	P <sub>3min</sub>	C <sub>min</sub>
13	12	4	5	171	12	5	6	177	12	3	3	162
14	12	5	5	174	12	5	5	174	11	3	3	150
15	12	5	4	171	12	5	5	174	10	3	3	138
16	12	7	6	183	12	5	5	174	9	3	3	126
17	12	7	7	186	12	6	6	180	9	3	3	126
18	12	9	8	195	12	7	6	183	8	2	2	108
19	12	8	7	189	12	6	6	180	8	2	2	108
20	12	6	7	183	12	7	7	186	8	2	2	108
21	12	7	6	183	12	7	7	186	7	2	2	96
22	12	5	6	177	12	4	6	174	7	2	2	96
23	12	6	6	180	12	6	5	177	7	2	2	96
24	12	6	5	177	12	5	5	174	6	2	2	84
25	12	4	4	168	12	5	5	174	6	2	2	84
26	12	5	6	177	10	4	5	147	6	2	2	84
27	12	4	5	171	11	4	4	156	6	2	2	84
28	9	6	6	144	8	5	3	120	6	2	2	84
29	10	6	5	153	8	4	6	126	5	2	2	72
30	8	4	5	123	6	5	3	96	5	2	2	72
31	9	5	4	135	6	5	3	96	5	2	2	72
32	7	3	4	105	7	4	5	111	5	2	2	72
33	8	4	4	120	7	5	4	111	5	2	2	72
34	7	5	4	111	6	4	5	99	5	2	2	72
35	7	4	4	108	7	5	4	111	5	2	2	72
36	8	4	4	120	7	4	4	108	4	1	1	54
37	7	5	4	111	7	4	3	105	4	1	1	54
38	7	4	4	108	7	3	4	105	4	1	1	54
39	8	4	4	120	6	3	4	93	4	1	1	54
40	8	4	4	120	7	4	3	105	4	1	1	54
41	7	4	5	111	6	3	3	90	4	1	1	54
42	7	4	3	105	6	2	3	87	4	1	1	54
43	7	3	3	102	7	3	3	102	4	1	1	54
44	6	3	3	90	5	3	2	75	4	1	1	54
45	5	2	4	78	6	2	3	87	4	1	1	54
46	6	4	3	93	6	3	3	90	4	1	1	54
47	6	4	3	93	5	3	3	78	4	1	1	54
48	6	3	2	87	5	4	3	81	3	1	1	42
49	5	4	3	81	5	3	3	78	3	1	1	42

4.7. táblázat Az F2 feladat műveletvégző igényei L= 60, 96 esetén



4.9. ábra Az F2 feladat költségfüggvényei

A 4.9. ábra a különböző költségfüggvények alakulását ábrázolja. Érdeemes megfigyelni, hogy az  $R=30$   $L=96$ , esetén egész közel került a megoldás költsége az adott  $R$ -hez tartozó elvi minimumhoz.

A lappangási idő növelésének előnyös hatása ezúttal is jobban érzékelhető, ha az egyes műveletvégzők több elemi művelet elvégzésére képesek, mert ekkor átkonfigurálhatók a pillanatnyi igényeknek megfelelően, így a műveletvégzők darabszáma tovább csökkenhet. Az előbbi példánál maradva növeljük a komplexitását a műveletvégzőknek. Tegyük fel, hogy van olyan  $P_1$  műveletvégző, mely képes ellátni az  $e_1 \dots e_{12}$  műveleteket és van egy  $P_2$  műveletvégző, amely képes az  $e_{13} \dots e_{36}$  elemi műveletek elvégzésére. Az elemi műveleti gráf ettől nem változik meg, azonban az allokáció és ütemezés során kedvezőbb eredmény is elérhető.

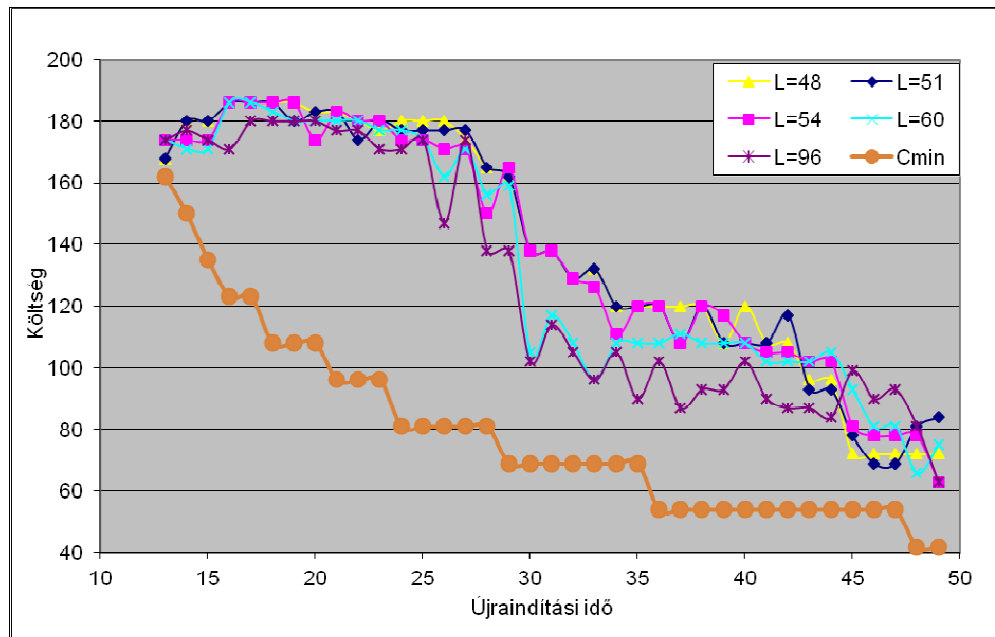
A műveletvégzők darabszámának alakulását többfunkciós műveletvégző alkalmazása esetén a 4.8. táblázat és a 4.9. táblázat mutatja. A táblázatoknak megfelelő költségfüggvényeket a 4.10. ábra mutatja.

$\alpha$	$\alpha^{-1}$	$\alpha^2$	L=48			L=51			L=54		
			$\cup$	$\alpha^{-1}$	$\alpha^2$	$\cup$	$\alpha^{-1}$	$\alpha^2$	$\cup$	$\alpha^{-1}$	$\alpha^2$
13	12	8	168	12	8	168	12	10	174		
14	12	12	180	12	12	180	12	10	174		
15	12	12	180	12	12	180	12	10	174		
16	12	14	186	12	14	186	12	14	186		
17	12	14	186	12	14	186	12	14	186		
18	12	14	186	12	14	186	12	14	186		
19	12	14	186	12	12	180	12	14	186		
20	12	13	183	12	13	183	12	10	174		
21	12	13	183	12	13	183	12	13	183		
22	12	12	180	12	10	174	12	12	180		
23	12	11	177	12	12	180	12	12	180		
24	12	12	180	12	11	177	12	10	174		
25	12	12	180	12	11	177	12	10	174		
26	12	12	180	12	11	177	12	9	171		
27	12	10	174	12	11	177	12	9	171		
28	11	11	165	11	11	165	10	10	150		
29	11	10	162	11	10	162	11	11	165		
30	9	10	138	9	10	138	9	10	138		
31	9	10	138	9	10	138	9	10	138		
32	8	11	129	8	11	129	8	11	129		
33	8	12	132	8	12	132	8	10	126		
34	8	8	120	8	8	120	7	9	111		
35	8	8	120	8	8	120	8	8	120		
36	8	8	120	8	8	120	8	8	120		
37	8	8	120	7	8	108	7	8	108		
38	8	8	120	8	8	120	8	8	120		
39	7	8	108	7	8	108	8	7	117		
40	8	8	120	7	8	108	7	8	108		
41	7	8	108	7	8	108	7	7	105		
42	7	8	108	8	7	117	7	7	105		
43	6	8	96	6	7	93	7	6	102		
44	6	8	96	6	7	93	7	6	102		
45	4	8	72	5	6	78	5	7	81		
46	4	8	72	4	7	69	5	6	78		
47	4	8	72	4	7	69	5	6	78		
48	4	8	72	5	7	81	5	6	78		
49	4	8	72	5	8	84	4	5	63		

4.8. táblázat Az F2 feladat műveletvégző igénye többfunkciós műveletvégzők esetén L=48, 51, 54

R	P <sup>1</sup>	P <sup>2</sup>	L=60			L=96			
			C	P <sup>1</sup>	P <sup>2</sup>	C	P <sub>1min</sub>	P <sub>2min</sub>	C <sub>min</sub>
13	12	10	174	12	10	174	12	6	162
14	12	9	171	12	11	177	11	6	150
15	12	9	171	12	10	174	10	5	135
16	12	14	186	12	9	171	9	5	123
17	12	14	186	12	12	180	9	5	123
18	12	13	183	12	12	180	8	4	108
19	12	12	180	12	12	180	8	4	108
20	12	12	180	12	12	180	8	4	108
21	12	12	180	12	11	177	7	4	96
22	12	12	180	12	11	177	7	4	96
23	12	11	177	12	9	171	7	4	96
24	12	11	177	12	9	171	6	3	81
25	12	10	174	12	10	174	6	3	81
26	11	10	162	10	9	147	6	3	81
27	12	9	171	12	10	174	6	3	81
28	11	8	156	9	10	138	6	3	81
29	11	9	159	9	10	138	5	3	69
30	7	7	105	6	10	102	5	3	69
31	8	7	117	7	10	114	5	3	69
32	7	8	108	7	7	105	5	3	69
33	6	8	96	6	8	96	5	3	69
34	7	8	108	7	7	105	5	3	69
35	7	8	108	6	6	90	5	3	69
36	7	8	108	7	6	102	4	2	54
37	7	9	111	6	5	87	4	2	54
38	7	8	108	6	7	93	4	2	54
39	7	8	108	6	7	93	4	2	54
40	7	8	108	7	6	102	4	2	54
41	7	6	102	6	6	90	4	2	54
42	7	6	102	6	5	87	4	2	54
43	7	6	102	6	5	87	4	2	54
44	7	7	105	6	4	84	4	2	54
45	6	7	93	7	5	99	4	2	54
46	5	7	81	6	6	90	4	2	54
47	5	7	81	6	7	93	4	2	54
48	4	6	66	5	7	81	3	2	42
49	5	5	75	4	5	63	3	2	42

4.9. táblázat Az F2 feladat műveletvégző igénye többfunkciós műveletvégzők esetén L=60, 96



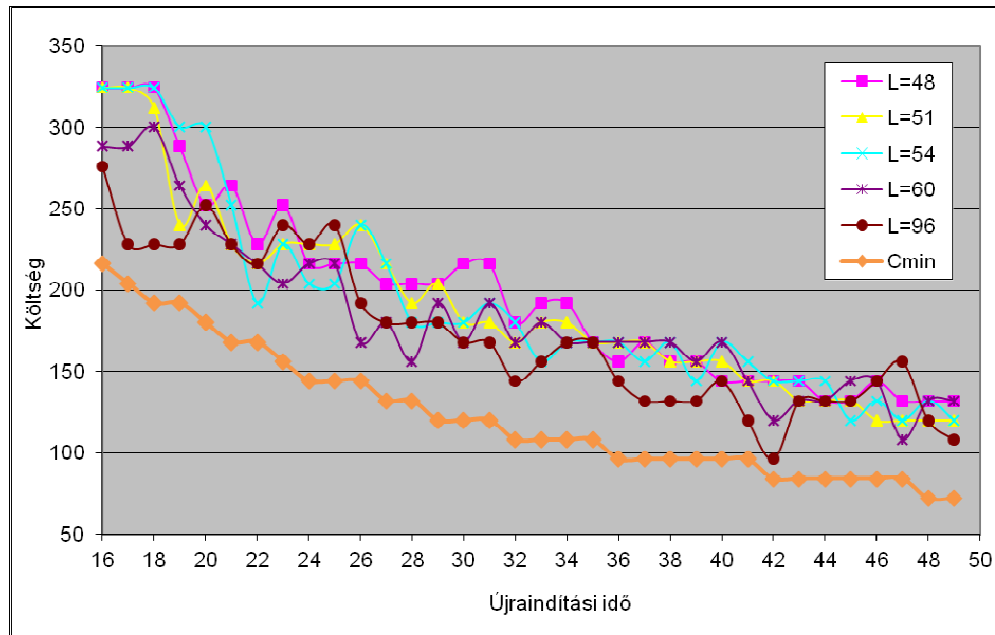
**4.10. ábra** Az F2 feladat költségfüggvényei többfunkciós műveletvégzők alkalmazása esetén

Ha rendelkezésünkre áll olyan komplex műveletvégző elem, amely a kiindulásként megadott elemi műveleti gráf bármely műveletének elvégzésére képes, akkor a műveletvégzők számának további csökkenését várhatjuk a megoldás optimalizálása során. Az F2 feladat esetében ez azt jelenti, hogy van egyetlen  $P_1$  műveletvégzőnk, amely képes elvégezni az  $e_1 \dots e_{36}$  elemi műveletek bármelyikét.

A különböző újraindítási és lappangási időkhöz tartozó műveletvégző darabszámokat és költségeket a 4.10. táblázat foglalja össze. A táblázathoz tartozó költségfüggvény ábrázolását a 4.11 ábra mutatja.

$\alpha$	$\alpha'$	L=48		L=51		L=54		L=60		L=96		$P_{\min}$	$C_{\min}$
		$\cup$	$\cap$	$\cup$	$\cap$	$\cup$	$\cap$	$\cup$	$\cap$	$\cup$	$\cap$		
16	27	324	27	324	27	324	24	288	23	276	18	216	
17	27	324	27	324	27	324	24	288	19	228	17	204	
18	27	324	26	312	27	324	25	300	19	228	16	192	
19	24	288	20	240	25	300	22	264	19	228	16	192	
20	21	252	22	264	25	300	20	240	21	252	15	180	
21	22	264	19	228	21	252	19	228	19	228	14	168	
22	19	228	18	216	16	192	18	216	18	216	14	168	
23	21	252	19	228	19	228	17	204	20	240	13	156	
24	18	216	19	228	17	204	18	216	19	228	12	144	
25	18	216	19	228	17	204	18	216	20	240	12	144	
26	18	216	20	240	20	240	14	168	16	192	12	144	
27	17	204	18	216	18	216	15	180	15	180	11	132	
28	17	204	16	192	15	180	13	156	15	180	11	132	
29	17	204	17	204	15	180	16	192	15	180	10	120	
30	18	216	15	180	15	180	14	168	14	168	10	120	
31	18	216	15	180	16	192	16	192	14	168	10	120	
32	15	180	14	168	15	180	14	168	12	144	9	108	
33	16	192	15	180	13	156	15	180	13	156	9	108	
34	16	192	15	180	14	168	14	168	14	168	9	108	
35	14	168	14	168	14	168	14	168	14	168	9	108	
36	13	156	14	168	14	168	14	168	12	144	8	96	
37	14	168	14	168	13	156	14	168	11	132	8	96	
38	13	156	13	156	14	168	14	168	11	132	8	96	
39	13	156	13	156	12	144	13	156	11	132	8	96	
40	12	144	13	156	14	168	14	168	12	144	8	96	
41	12	144	12	144	13	156	12	144	10	120	8	96	
42	12	144	12	144	12	144	10	120	8	96	7	84	
43	12	144	11	132	12	144	11	132	11	132	7	84	
44	11	132	11	132	12	144	11	132	11	132	7	84	
45	11	132	11	132	10	120	12	144	11	132	7	84	
46	12	144	10	120	11	132	12	144	12	144	7	84	
47	11	132	10	120	10	120	9	108	13	156	7	84	
48	11	132	10	120	11	132	11	132	10	120	6	72	
49	11	132	10	120	10	120	11	132	9	108	6	72	

4.10. táblázat Az F2 feladat műveletvégző igénye komplex műveletvégző esetén



4.11 ábra Az F2 feladat költség függvényei komplex műveletvégzők használata esetén

A 4.11 ábra költségfüggvényeiből látszik, hogy az F2 feladatban amennyiben egyetlen komplex műveletvégző típust használunk, a lappangás növelésével jól megközelíthetjük az adott újraindításhoz tartozó elvi minimális költségű megoldást (például  $L=96$ ,  $R=17$  vagy  $R=42$ ).

Összességében megállapítható, hogy a nagyobb lappangási időhöz tartozó költségfüggvény görbék alacsonyabbak, de nem minden esetben. Ennek oka az ütemezésben keresendő, mert ha egy elemi művelet indítási ideje rögzítésre kerül, akkor az hatással lehet a többi elemi műveletvégző ütemezésére és allokálására egyaránt. A bemutatott példákban a PIPE tervezőrendszer erő vezérelt (force-directed) ütemezőjét használtam minden esetben. A felhasznált műveletvégzők legkisebb darabszámát komplex műveletvégzők alkalmazása esetén érhetjük el a műveletvégző univerzalitásának köszönhetően. Ennek azonban ára van, hiszen egy komplex egységnek csak bizonyos képességeit tudjuk kihasználni az allokáció során. Az értekezésben alkalmazott költségfüggvényt alkalmazva ez a költség úgy jelenik meg, hogy a műveletvégző által elvégezhető műveletek közül mindig a leglassabb végrehajtási idővel számolunk. A vizsgált R és L tartományokban megfigyelhető, hogy míg a legkevesebb felhasznált műveletvégző a 4.7. táblázat alapján 11 darab, addig a 4.10. táblázat alapján 9 darab, a legkisebb költséget (63 egység) a köztes megoldás eredményezte, ahol kétféle műveletvégzőt alkalmaztunk. Bár a komplex egység eredményezte a legkevesebb műveletvégző felhasználását, a vizsgált tartományban a minimális költsége ennek a legmagasabb (96) egység. Ez érthető, hiszen a komplex

műveletvégzőt a leghosszabb időt igénylő elemi művelet végrehajtási idejével vesszük figyelembe a költségfüggvény számításakor. A későbbiekben esetleg célszerű megfontolni a költségfüggvény hangolását a komplex műveletvégző egységekhez.

**Tapasztalatok:**

1. A lappangási idő növekedése miatt az elemek mobilitása megnő. A megnövekedett mobilitás miatt az ütemező és az allokáló algoritmus futási ideje hosszabb lesz, mert megnövekszik az egyes szabadsági fokok száma.
2. Ha  $R \leq 2 \cdot \min\{t_i\}$ , akkor nem várható jelentős csökkenés, mert a többszörözés és a gyakori újraindítás miatt nem lehet többször felhasználni egyik műveletvégzőt sem.
3. Az előző szabálynak némileg ellentmond az a tény, hogy a lappangási idő kismértékű növelése miatt az ütemezés megváltozik és ez is okozhat feladatfüggő költségcsökkenést.
4. Ha egy  $e_i$  elemi művelet esetén  $R \leq t_i$ , akkor az  $e_i$  elemi művelet többszörözése [3] miatt nem várható javulás az  $e_i$  elemi műveletvégzők számának csökkenésében a lappangás növelésének hatására.

## 4.2. A javasolt algoritmus a lappangási idő növelésének hatásvizsgálatára

Az algoritmus tömör megadásához az alábbi jelöléseket használom fel. Ezek közül néhány már korábban is szerepelt, az egyszerűbb áttekinthetőség miatt újra megadom.

R: újraindítási idő

L: lappangási idő

$L_{\min}$ : a HLS tervező eszköz által meghatározott lappangási idő (adott R alkalmazása esetén)

$L_{\max}$ :  $2 \cdot L_{\min}$ , vagy a felhasználó által megadott tetszőleges  $L_{\min}$ -nél nagyobb érték

$e_j$ : elemi művelet, az EOG egy elemi csomópontja, j index 1-től a csomópontok számáig terjed

$E(e_1, \dots, e_j, \dots, e_m)$ : Az EOG-ben található elemi műveletek halmaza

$\Phi_F(f_1, f_2, \dots, f_k, \dots, f_n)$ : Az elemi műveletek teljes fedőrendszere, amelynek  $f_k$  blokkjai tartalmazzák az EOG összes k típusú elemi műveletét. (azért nem partíció, mert előfordulhat, hogy több műveletvégző is képes ugyanazon elemi művelet végrehajtására)

$t_k$ : az  $f_k$  blokkba tartozó elemi műveletek végrehajtási ideje (ez minden a blokkba tartozó műveletre azonos).

$n_k$ : az  $f_k$  blokkba tartozó elemi műveletek darabszáma

$P_i$ : műveletvégző típus, amelyhez  $\Phi_F$ -ből képezhető  $\Phi_i$  részleges fedőrendszer tartozik. Ennek blokkjai tartalmazzák a rendre a  $P_i$  műveletvégzővel végrehajtható elemi műveleteket elemi művelet típusok végrehajtására

$M_q$ : az allokált gráf q-adik műveletvégző példánya (típusa valamelyik  $P_i$  lehet)

$M(M_1, \dots, M_q, \dots, M_z)$ : Az allokáció után szükséges műveletvégző egységek példányainak halmaza.

$\Pi_M(m_1, \dots, m_i, \dots, m_p)$ : a műveletvégző példányok teljes partíciója, amelynek  $m_i$  blokkja tartalmazza azokat az  $M_q$  műveletvégző példányokat, amelyek  $P_i$  típusúak.

$N_i$ :  $m_i$  blokk számossága, vagyis az  $P_i$  műveletvégzők szükséges darabszáma.

$T_i$ :  $P_i$  által végrehajtható elemi műveletek végrehajtási idejének maximuma

$C_i$ : Az allokáció eredményeként felhasználandó  $P_i$  műveletvégzők költsége:

$$C_i = N_i \cdot T_i$$

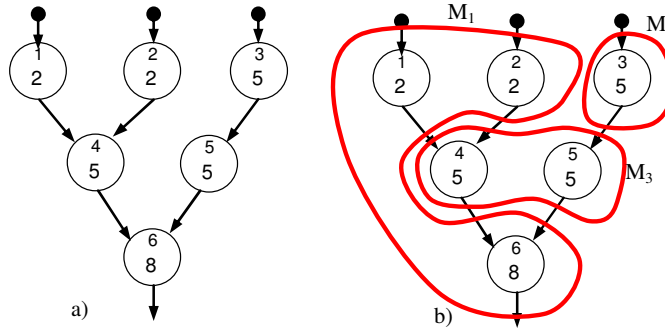
$C_d$ : Elvárt minimális költség (leállási feltételhez)

C: A megvalósítás teljes költsége:  $C = \sum_i C_i$

$N_{\min}$ : az a legkevesebb darabszám, amelynél kevesebb  $P_i$  típusú műveletvégzővel már nem valósítható meg a feladat a megadott R esetén

$C_{\min}$ : Minimális költség  $C_{\min} = \sum_i N_{\min} \cdot T_i$

A 4.12. ábrán egy egyszerű elemi műveleti gráf illetve egy allokált műveleti gráf esetére - alkalmazva az előbbieken bevezetett jelöléseket – a következő paraméterek határozhatók meg.



4.12. ábra EOG (a) és allokált műveleti gráf (b)

Műveletvégző típus	elvégezhető elemi műveletek	Művelet neve	művelet végrehajtási ideje
P <sub>1</sub>	e <sub>1</sub> , e <sub>2</sub>	ADD	2
	e <sub>6</sub>	MUL	8
P <sub>2</sub>	e <sub>3</sub> , e <sub>4</sub> , e <sub>5</sub>	SIN	5

4.11. táblázat Műveletvégzők és elemi műveletek tulajdonságai

**Az elemi műveleti gráf paraméterei:**

Az elemi műveletek halmaza:  $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$

Az elemi műveletek teljes fedőrendszere:  $\Phi_F = \{f_1, f_2, f_3\} = \{(e_1, e_2)(e_3, e_4, e_5)(e_6)\}$

Az egyes blokkokba tartozó elemi műveletek végrehajtási ideje:  $t_1 = 2, t_2 = 5, t_3 = 8$

Az egyes blokkokba tartozó elemi műveletek darabszáma:  $n_1 = 2, n_2=3, n_3=1$

**Az allokált műveleti gráf paraméterei:**

Az allokálás után szükséges műveletvégző egységek halmaza:  $M = \{M_1, M_2, M_3\}$

A műveletvégző példányok teljes partíciója:  $\Pi_M = \{m_1, m_2\} = \{(M_1)(M_2, M_3)\}$

P<sub>1</sub> típusú műveletvégzők szükséges darabszáma:  $N_1 = 1$

P<sub>2</sub> típusú műveletvégzők szükséges darabszáma:  $N_2 = 2$

P<sub>1</sub> által végrehajtható elemi műveletek végrehajtási idejének maximuma:  $T_1 = 8$

P<sub>2</sub> által végrehajtható elemi műveletek végrehajtási idejének maximuma:  $T_2 = 5$

P<sub>1</sub> műveletvégzők költsége:  $C_1 = N_1 \cdot T_1 = 8$

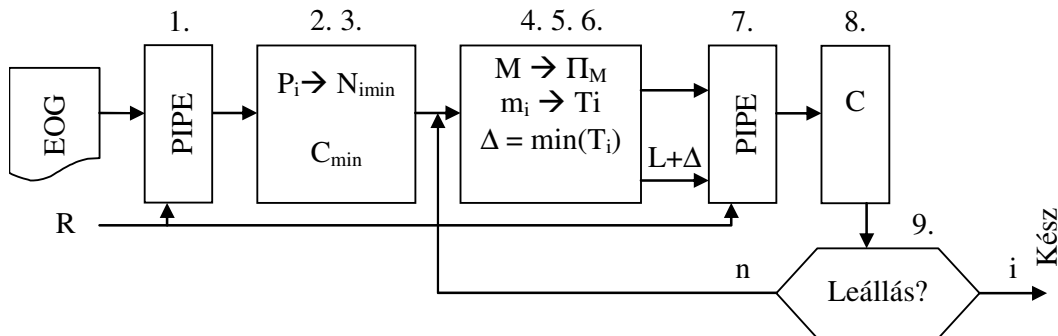
P<sub>2</sub> műveletvégzők költsége:  $C_2 = N_2 \cdot T_2 = 10$

A megvalósítás teljes költsége:  $C = 18$

**Az algoritmus javasolt lépései a következők:**

1. Az eredeti EOG-re a megadott R értékkel 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. Létrehozunk az M halmaz alapján a  $\Pi_M$  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  $\Delta$  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  $\Delta$  növekménnyel megnöveljük az aktuális lappangás értékét.
7. PIPE újrafuttatása.
8. C költség kiszámítása, leállási feltétel vizsgálat, majd ismétlés a 4. lépéstől.
9. Leállítás 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 4.13. ábra mutatja.



**4.13 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.

A fejezetben bemutatott módszert és algoritmust, illetve használatukat a 2. tézisben fogalmaztam meg.

### 4.3. Második 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 átbocsajtási 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]

## 5.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 működőképeseek 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 működőképesenek kell lennie, illetve egy ilyen elosztott módon megvalósított funkció végrehajtásában bekövetkező hiba több érintett berendezés leállítását 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<sup>20</sup> + 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 algoritmus - hasonlóan a magas szintű logikai szintézis feladatokhoz – reprezentálható EOG-vel. Az EOG előállítás után a HLS eszközök alkalmazhatóak a gráf alapján történő optimalizálásra. A kizárás szemléltetésére vegyünk egy példát. Egy megmunkáló gép álljon egy szállító szalagból és egy automatizált fúrógépből. A vezérlést úgy tervezzük, hogy maximum két PLC egységgel megvalósítható legyen, de szeretnénk biztosítani, hogy a szalag vészleállító gombjának kezelése ne kerülhessen a fúrógépvezérléssel egy közös vezérlő egységbe, mert annak meghibásodása vagy

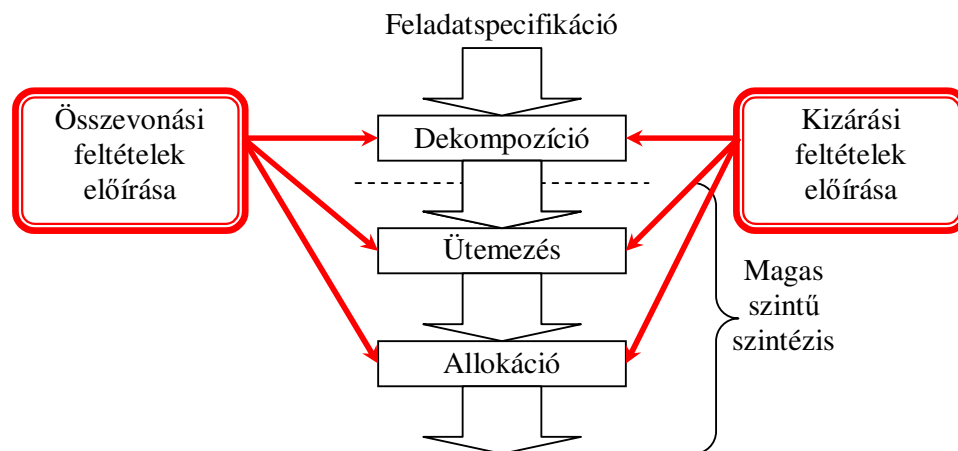
---

<sup>20</sup> PLC: Programmable Logic Controller

karbantartás miatti leállítása esetén a szállítószalag vezérlés leállító gomb nélkül maradhat. A probléma egyik lehetséges megoldása, az, hogy megtiltjuk, hogy a vészgomb és a fúrógépvezérlés egy közös egységbe kerüljön. Egy másik megközelítésben feltételesen megengedhetjük, de csak akkor, ha a teljes vezérlést sikerül egyetlen egységben megvalósítani. Mindkét megoldás esetén biztosított, hogy csak akkor szűnik meg a vészgomb kezelése, ha a szalag vezérlőegységét lekapcsolják, de ekkor a szalag is leáll.

A fentiekben vázolt problémák kezeléséhez tehát új feltételek bevezetése is szükséges. Biztosítanunk kell a lehetőséget a tervező számára, hogy már az előzetes specifikációban meg tudjon adni olyan peremfeltételeket, amelyek a későbbi allokáció során megengedő vagy kizáró feltételként kerülnek figyelembevételre. A kizárási feltételek megadásával biztosíthatjuk, hogy össze nem tartozó funkciók ne kerülhessenek például egy közös vezérlőegységbe.

A magas szintű tervezés folyamatát az 5.1. ábra mutatja. A feladat-specifikációból 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].



**5.1. ábra Összevonások és kizárások előírása a tervezés különböző fázisaiban**

Az 5.1. ábra tartalmazza azokat a tervezési fázisokat, amelyekben az általam megfogalmazott módszerrel a kizárási és összevonási peremfeltételek figyelembe vehetők. A továbbiakban 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.

Az összevonási és kizárások előírása az 5.1. ábra alapján a tervezés három fázisában is lehetséges. További vizsgálatot igényel az, hogy melyik fázisban milyen hatékonysággal lehet ezeket az előírásokat figyelembe venni. A következőkben ezeket a lehetőségeket vizsgálom meg részletesebben.

### **5.1. Összevonási feltételek figyelembevétele**

Amennyiben a dekompozíció során szeretnénk előírni, hogy bizonyos elemi műveletek mindenképpen együtt kerüljenek megvalósításra, úgy a legegyszerűbb mód az, hogy nem választjuk szét ezeket a feladatokat, hanem egyetlen egységként (elemi műveletként) jelenítjük meg az EOG-n.

Az ütemezés során a peremfeltételként megadott összevonások az egyes elemi műveletvégzők mobilitásának meghatározásakor juthatnak szerephez. Ha ugyanis két elemi műveletet mindenképpen szeretnénk összevonni, akkor már az ütemezés során biztosítani kell, hogy ezek időben ne lapolhassák át egymást. Ehhez új ütemező algoritmusra van szükség, vagy a meglévő ütemezést célszerű úgy módosítani, hogy az a nagyobb mobilitású elemi műveletek indítási idejének rögzítésekor vegye figyelembe az összevonásra vonatkozó előírást

Az allokáció során az előre megadott összevonási igényeknek csak azon részét tudjuk figyelembe venni, amelyek az ütemezés miatt biztosan nem fedik át egymást. Ebben az esetben kezdeti lépésként az összevonásra előírt és egymást át nem lapoló elemi műveleteket még az allokáló algoritmusok futása előtt rögzíteni kell. Az allokáló algoritmusnak csak a többi, előre nem rögzített elemi művelettel kell foglalkoznia.

Az előírt összevonások kezelése során a tervező programnak - mindhárom lépésben – jeleznie kell, ha azok valamilyen ok miatt nem teljesíthetők.

## **5.2. Kizárási feltételek lehetséges figyelembe vétele**

A dekompozíció során a kizárási feltételek annyiban befolyásolják a dekomponálás folyamatát, hogy a kizárandó részeket mindenképpen szét kell választaniuk. Amennyiben az alkalmazott tervező rendszer lehetőséget biztosít az egyes műveletek részleges csoportosítására, úgy ügyelni kell arra, hogy az egymást kizáró elemi műveletek különböző csoportokba kerüljenek.

A kizárási feltételek befolyásolhatják az ütemezést is, hiszen az egymást kizáró elemi műveletek időben átlapolhatják egymást, ha nincs más, ezt kizáró függőség (pl. adat precedencia). Így ezeknek az elemi műveleteknek a mobilitása nagyobb lehet, a műveletek időzítésének rögzítése során nem kell törekedni arra, hogy egymás után induljanak, hanem indíthatók akár egyszerre és átlapolva is, hiszen garantáltan különböző műveletvégzőkben lesznek allokálva. Az ütemezési algoritmus módosítása során ezek a feltételek figyelembe vehetők például a nagy mobilitású elemi műveletek indítási idejének rögzítésekor.

Az allokáció során a kizárási feltételek lényegesen befolyásolhatják az allokáció eredményét, ezért a következőkben részletesen foglalkozom ezzel a kérdéssel.

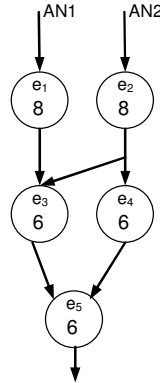
### **5.2.1. A kizárási feltételek megadása allokációhoz**

Az EOG egyes csomópontjainak összevonhatóságát megengedhetjük vagy megtilthatjuk. Speciális esetekben az összevonást további feltételekhez köthetjük. Az összevonhatóság vizsgálata hasonló probléma, mint a sorrendi hálózatok állapotainak összevonása [12] azzal a különbséggel, hogy itt a tervezőnek kell meghatároznia az összevonhatóság, a kizárás és a feltételes összevonhatóság lehetőségét. A feltételek egy része algoritmikusan generálható például az EOG ütemezése alapján, azonban a speciális igények figyelembe vétele intuitív megadást is igényelhet.

A következő példa a módszer szemléltetésére szolgál. Az 5.2. ábra mutatja egy feladat elemi műveleti gráfját. A környezetből két analóg jelet (AN1 és AN2) fogad a rendszer. A bemeneti jeleket az  $e_1$  és  $e_2$  elemi műveletek elvégzése után az  $e_3...e_5$  elemi műveletvégzők alakítják tovább, végül az eredményt  $e_5$  egy kommunikációs interfészen továbbítja a környezetnek. A csomópontokat 1-1 mikrokontrollerrel tervezzük megvalósítani, így az elemi műveleti gráf kiindulásként összesen öt műveletvégzőt tartalmaz (ha nem optimalizálunk semmit). Az egyes kontrollerek csak a beépített kommunikációs vonalaik segítségével kapcsolódnak egymáshoz.

A műveletvégző egységek összevonás szempontjából lényeges legfontosabb tulajdonságait az 5.1. táblázat tartalmazza. A kizárási és összevonási feltételek ebben a példában részben az erőforrások korlátaiból adódnak. A kizárási és összevonási

feltételek megadásához egy kompatibilitási félmátrixot használtam. A kompatibilitási félmátrix minden egyes bejegyzése több különböző állapotot vehet fel az 5.2. táblázat szerint. Az  $e_1$  és  $e_2$  jelű művelet összevonhatósága például az „analóg bemenet” erőforrás korlátossága miatt nem lehetséges, ezért került kizáráásra.



**5.2. ábra A megvalósítandó feladat elemi műveleti gráfja**

Erőforrás	db
Analóg bemenet	1
Kommunikációs vonal	3

**5.1. táblázat A példában alkalmazott műveletvégzők tulajdonságai ( $e_1..e_5$ )**

Jelölés	Állapot
✓	<b>Összevonható</b>
×	<b>Nem vonható össze</b>
☑	<b>Mindenképpen össze kell vonni</b> Előre megadott követelmény jelölése.
☒	<b>Semmiképpen sem vonható össze</b> Előre megadott követelmény jelölése
f	<b>Feltételesen vonható össze</b> A megadott feltétel kiértékelésétől függ, hogy az adott műveletek összevonhatók vagy sem.

**5.2. táblázat A kompatibilitási félmátrix javasolt jelölései**

**Megjegyzések:**

1. Az „összevonható” és a „mindenképpen össze kell vonni”, illetve a „Nem vonható össze” és a „Semmiképpen sem vonható össze” jelölések megkülönböztetése csak a követelmény keletkezésének beazonosítása miatt

került megkülönböztetésre, a kompatibilitási osztályok előállításakor nincs különbség közöttük.

2. A feltétel kiértékelésének végeredménye kétféle lehet: „összevonható”, „Nem vonható össze”.

Az 5.3. ábra kompatibilitási félmátrixa mutatja a megengedett összevonásokat, a kizárásokat és a feltételes összevonásokat. Ez utóbbit jelöli a 3-4 bejegyzés, ami azt jelzi, hogy az  $e_3$ - $e_5$  illetve  $e_4$ - $e_5$  elemi műveletek csak akkor vonhatók össze, ha az  $e_3$ - $e_4$  is összevonható. Ha az előírt feltétel nem teljesül, akkor a kompatibilitási osztályok meghatározása során azt kizárásként kell kezelni.

$e_2$	<input checked="" type="checkbox"/>			
$e_3$	✓	X		
$e_4$	X	✓	✓	
$e_5$	X	X	3-4	3-4
	$e_1$	$e_2$	$e_3$	$e_4$

5.3. ábra Kompatibilitási félmátrix

A [12]-ben bemutatott összevonási eljáráshoz hasonlóan, kiindulásként tételezzük fel, hogy az összes művelet összevonható, majd ezt a halmazt a kizárások figyelembevételével tovább bontjuk. Az összevont műveleteket egy zárójelen belül, vesszővel elválasztva soroljuk fel. Végül meghatározzuk a maximális kompatibilitási osztályokat.

#### Lépés Részeredmények

1. ( $e_1, e_2, e_3, e_4, e_5$ )
2. ( $e_2, e_3, e_4, e_5$ ) ( $e_1, e_3$ )
3. ( $e_3, e_4, e_5$ ) ( $e_2, e_4$ ) ( $e_1, e_3$ )

A kapott osztályok: A B C

#### A kapott eredményekkel kapcsolatos megjegyzések:

1. A meghatározott osztályokat tovább kell alakítani, mert minden elemi műveletet csak egyszer kell elvégezni, ezért a többszörösen lefedett műveleteket valamelyik osztályból el kell hagynunk.
2. Előfordulhat, hogy olyan osztályt kapunk, amelynek minden elemét más osztály is lefedi, ezért érdemes megvizsgálni, hogy egy ilyen osztályt nem lehet-e teljesen elhagyni.

3. Egy osztály elhagyása, vagy egy elemi művelet elhagyása valamelyik osztályból más osztályokra is hatással lehet, amennyiben az feltételként szerepelt, ezért elhagyás esetén meg kell vizsgálni, hogy a többi osztály is zárt marad-e.
4. A többször szereplő műveletek megtalálásához készíthető egy lefedési tábla, amely megmutatja, hogy mely elemi műveletet melyik osztály(ok) fed(ik) le.

Az 5.3. táblázat az előző példa alapján kapott osztályok által lefedett elemi műveleteket mutatja.

Kompatibilitási osztályok	Lefedendő elemi művelet				
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>
A			X	X	X
B		X		X	
C	X		X		

**5.3. táblázat Lefedési tábla**

Az 5.3. táblázatból jól látszik, hogy az e<sub>3</sub> és e<sub>4</sub> elemi műveletek több osztályban is szerepelnek. Az 5.3. ábra kompatibilitási félmátrixa alapján vizsgáljuk meg, hogy teljes osztály elhagyható-e illetve, hogy milyen következménnyel jár a többszörösen lefedett műveletek elhagyása valamelyik osztályból. Mivel diszjunkt osztályokat keresünk, ezért a példánkban két elemi művelet (e<sub>3</sub> és e<sub>4</sub>) hagyható el valamelyik két osztályból (AC vagy AB), ezért a különböző variációk száma négy. Az 5.4. ábra... 5.7. ábra ezt a négy lehetséges esetet szemlélteti. A lefedési táblákban zöld háttérrel jelöltem az elhagyandó elemi műveleteket. A feltételezett elhagyás következményeként - a zártság biztosítása miatt - szükségessé váló új kizárást piros háttér mutatja.

1.eset

Lefedési tábla

	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>
A			X	X	X
B		X		X	
C	X		X		

Módosított kompatibilitási félmátrix

e <sub>2</sub>	<input checked="" type="checkbox"/>			
e <sub>3</sub>	✓	X		
e <sub>4</sub>	X	✓	✓X	
e <sub>5</sub>	X	X	X3-4	X3-4
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>

**5.4. ábra Az osztályok diszjunktá tételének lehetséges esetei (1)**

Diszjunkt osztályok:

- A: e<sub>5</sub>
- B: e<sub>2</sub>, e<sub>4</sub>
- C: e<sub>1</sub>, e<sub>3</sub>

2. eset

	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>
A			X	X	X
B		X		X	
C	X		X		

e <sub>2</sub>	X			
e <sub>3</sub>	✓	X		
e <sub>4</sub>	X	✓X	✓X	
e <sub>5</sub>	X	X	X3-4	X3-4
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>

5.5. ábra Az osztályok diszjunktá tételének lehetséges esetei (2)

A feltételes elhagyás miatt nem teljesül a zártság, hiszen e<sub>4</sub> és e<sub>5</sub> csak akkor lehet együtt, ha e<sub>3</sub> is ugyanazon osztályban van. Emiatt négy osztályt kapunk a zárttá tétel és a feltételek feloldása után.

Diszjunktá tett osztályok:

- A: (e<sub>5</sub>)
- B: (e<sub>4</sub>)
- C: (e<sub>1</sub>, e<sub>3</sub>)
- D: (e<sub>2</sub>)

3. eset

	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>
A			X	X	X
B		X		X	
C	X		X		

e <sub>2</sub>	X			
e <sub>3</sub>	✓X	X		
e <sub>4</sub>	X	✓	✓X	
e <sub>5</sub>	X	X	X3-4	X3-4
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>

5.6. ábra Az osztályok diszjunktá tételének lehetséges esetei (3)

A feltételes elhagyás miatt nem teljesül a zártság! e<sub>3</sub> és e<sub>5</sub> csak akkor lehet együtt, ha e<sub>4</sub> is ugyanazon osztályban van. Emiatt négy osztályt kapunk a zárttá tétel és a feltételek feloldása után.

Diszjunktá tett osztályok:

- A: (e<sub>5</sub>)
- B: (e<sub>2</sub>, e<sub>4</sub>)
- C: (e<sub>3</sub>)
- D: (e<sub>1</sub>)

4. eset

	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>
A			X	X	X
B		X		X	
C	X		X		

e <sub>2</sub>	X			
e <sub>3</sub>	✓X	X		
e <sub>4</sub>	X	✓X	✓	
e <sub>5</sub>	X	X	3-4	3-4
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>

5.7. ábra Az osztályok diszjunktá tételének lehetséges esetei (4)

Diszjunkt osztályok:

- A: e<sub>3</sub>, e<sub>4</sub>, e<sub>5</sub>
- B: e<sub>2</sub>
- C: e<sub>1</sub>

A négy lehetőségből kettő esetében három osztályt kaptunk, két esetben pedig négy osztályt, mert a feltételes elhagyás során valamelyik osztály zártsága sérült. Amennyiben a példában szereplő rendszer megvalósításakor a minimális műveletvégző darabszámra törekszünk, úgy két azonos darabszámot igénylő megoldás közül választhatunk:

Egy lehetséges megoldás:  $(e_3, e_4, e_5)(e_2)(e_1)$

Egy másik lehetséges megoldás:  $(e_1, e_3)(e_2, e_4)(e_5)$

A fenti folyamat nem garantálja, hogy minden esetben megtaláljuk a minimális számú diszjunkt és zárt maximális kompatibilitási osztályokat. Egy adott megoldás értékeléséhez és a felesleges további próbálgatások elkerülése érdekében jól használható a következő bizonyítható elméleti korlát [12/234, 13/264]:

$$M \leq P \leq \min(n, k)$$

Ahol:

- $P$  az éppen összevont, legkevesebb műveletvégzőt tartalmazó gráf műveletvégzőinek száma
- $n$  a kiindulásként adott az EOG 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.

Az  $M$  értékének megállapításakor először az eredeti kompatibilitási félmátrix alapján határozzuk meg a maximális kompatibilitási osztályokat, majd a lefedési tábla segítségével megvizsgáljuk, hogy nem hagyható-e el valamelyik osztály úgy, hogy a többi zárt maradjon. Ha elhagyható, akkor ezt az új  $M' \leq M$  értéket kell a fenti egyenlőtlenség során figyelembe venni.

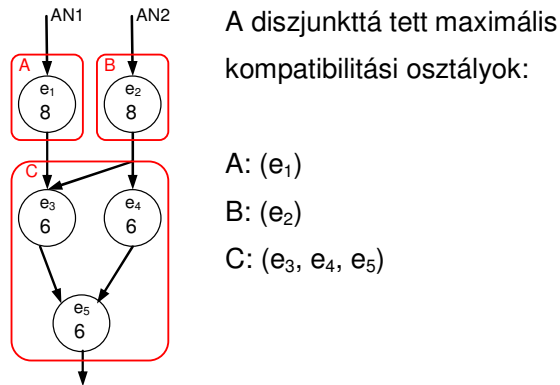
A fenti 2. és 3. esetben  $M = 3$ ,  $P = 4$ ,  $n = 5$ ,  $k = 3$  vagyis:

$$3 \leq 4 \leq \min(5, 3) \rightarrow \text{Nem teljesül, tehát érdemes tovább keresni}$$

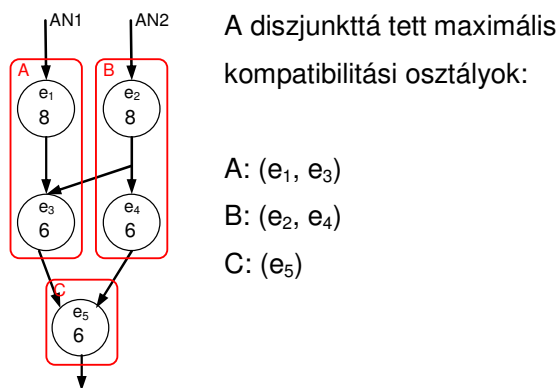
Az 1. és 4. esetben  $M = 3$ ,  $P = 3$ ,  $n = 5$ ,  $k = 3$  vagyis:

$$3 \leq 3 \leq \min(5, 3) \rightarrow \text{teljesül, tehát elfogadható az eredmény}$$

Az 1. eset megoldásához tartozó allokált gráfot mutatja az 5.8. ábra, míg a 4. eset megoldását az 5.9. ábra mutatja. Az ábrákon pirossal keretezve jelöltem a kompatibilitási osztályok alapján azonos egységbe allokált műveleteket.



**5.8. ábra Allokált műveleti gráf (1. eset)**



**5.9. ábra Allokált műveleti gráf (4. eset)**

Figyeljük meg, hogy a fenti műveleti gráfokon teljesülnek az 5.1. táblázat által megadott erőforrás korlátok. Egyik műveletvégzőbe sem jut egynél több analóg jel, valamint a ténylegesen felhasznált kommunikációs vonalak száma sem haladja meg az előírt hármat. (Ezek a feltételek még a kompatibilitási félmátrix megadásakor kerültek figyelembevételre).

### 5.3. A javasolt új allokációs algoritmus

A fentiek értelmében 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 diszjunkttségét.
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 zárttság, akkor vagy nem hagyjuk el az egész osztályt, vagy mérlegelhető, hogy mégis elhagyjuk és inkább más osztályok felbontásával biztosítjuk a zárttságot. Ez a lépés feladatfüggő, ezért próbálgatást is igényelhet.
6. Amennyiben az osztályok nem diszjunktak, úgy diszjunktá tesszük őket. A diszjunktá 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 [12], [13] adhat támpontot a kapott eredmény „jóságáról”.
8. A kapott osztályok alapján az allokációt elvégezzük.

Az algoritmus lépéseit az 5.10. ábra szemlélteti.



**Megjegyzések az előzőekben kidolgozott algoritmussal kapcsolatban**

1. A kezdeti kizáró feltételek generálásához a korábban lefuttatott ütemező algoritmus eredménye jól használható a [3] – ban ismertetett CONCHECK algoritmussal kombinálva.
2. A kidolgozott algoritmus nem foglalkozik a feltételes összevonások és tiltások formális kezelésével. A feltételek megadása során előnyösen alkalmazhatók a Boole-algebra alapműveletei. A megadott kifejezéseket minden lépésben (pl.: zártság vizsgálat) újra és újra ki kell értékelni, ha a kompatibilitási félmátrix bármely bejegyzése módosul. A feltétel végeredménye kétállapotú lehet, vagy teljesül az összevonhatóság, vagy tiltásra kerül.
3. A kompatibilitási félmátrix minden egyes bejegyzése több különböző állapotot vehet fel az 5.2. táblázat szerint. Az algoritmusok futtatása során bizonyos bejegyzések módosulhatnak.
4. A feltételes megjegyzések kiértékelésük után az első két jelölés valamelyikével helyettesítendő.

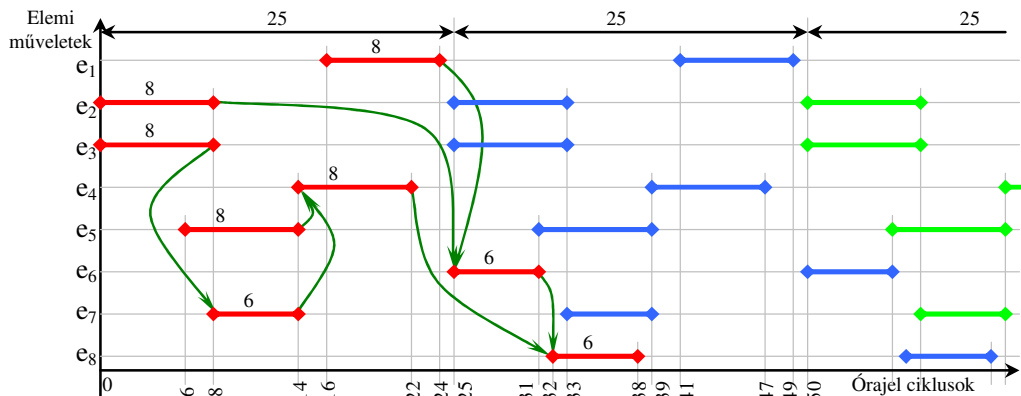
## 5.4. Kompatibilitási félmátrix automatikus generálása az ütemezés alapján

Az előzőekben kidolgozott allokációs algoritmushoz szükség van egy megfelelően kitöltött kompatibilitási félmátrixra. Ennek nagy része az ütemezés során meghatározott adatok alapján automatikusan előállítható, míg egy része a feladatfüggő specifikáció és a tervezői igények alapján adható meg. A módszer szemléltetéséhez példaként vegyük a 4. fejezet F1 feladatát megoldó hálózatot  $R=25$ ,  $L=39$  esetén. A műveletvégzők tulajdonságait a 4.1. táblázat tartalmazza. A PIPE tervezőrendszer által szolgáltatott ütemezési adatokat mutatja az 5.11. ábra. Az ütemezés szempontjából az adatsor 4. és az 5. oszlopában szereplő adatok érdekesek. A 4. oszlop mutatja a művelet végrehajtási idejét, az 5. oszlop pedig a művelet indulásának időpontját.

Részlet az „out.sched.fce” ütemező végeredményét tartalmazó állományból							
e1	MUL	P1	<b>8</b>	<b>16</b>	16	[ ]	[e6]
e2	MUL	P1	<b>8</b>	<b>0</b>	0	[ ]	[e6]
e3	MUL	P1	<b>8</b>	<b>0</b>	0	[ ]	[e7]
e4	MUL	P1	<b>8</b>	<b>14</b>	14	[e7, e5]	[e8]
e5	COS	P2	<b>8</b>	<b>6</b>	6	[ ]	[e4]
e6	ADD	P3	<b>6</b>	<b>25</b>	25	[e1, e2]	[e8]
e7	ADD	P3	<b>6</b>	<b>8</b>	8	[e3]	[e4]
e8	SUB	P4	<b>6</b>	<b>32</b>	32	[e6, e4]	[ ]

5.11. ábra Az ütemezés végeredményének részlete

A kapott eredményeket időfüggvényen ábrázolva szemléltetem az egyes elemi műveletek foglaltságát, ami alapján a kizárási feltételek meghatározhatók. Az 5.12. ábra pirossal, kézzel és zölddel szemlélteti rendre az egymás utáni három újraindításkor érkező bemeneti adatok feldolgozásának időzítését. Mivel  $R=25$ , ezért a 25. órajel ciklusban érkezik a második adatsor. Az első eredmény a 38 órajel ciklustól már elkészült, így a 39. ütemben a kimeneten rendelkezésre áll.



5.12. ábra A műveletek foglaltsága ( $R=25$ ,  $L=39$ )

Az egyidejűleg működő elemi műveletek nem kerülhetnek egy feldolgozó egységbe, ezért ki kell zárni az összevonhatóságukat. Az 5.12. ábra és az eredeti specifikáció alapján meghatározott kizárásokat az 5.4. táblázat foglalja össze.

Elemi művelet	Nem kerülhet egybe az ütemezés alapján	Nem kerülhet egybe a műv.típus miatt
e <sub>1</sub>	e <sub>4</sub>	P <sub>2</sub> , P <sub>3</sub> , P <sub>4</sub>
e <sub>2</sub>	e <sub>3</sub> , e <sub>5</sub>	P <sub>2</sub> , P <sub>3</sub> , P <sub>4</sub>
e <sub>3</sub>	e <sub>2</sub>	P <sub>2</sub> , P <sub>3</sub> , P <sub>4</sub>
e <sub>4</sub>	e <sub>1</sub>	P <sub>2</sub> , P <sub>3</sub> , P <sub>4</sub>
e <sub>5</sub>	e <sub>2</sub> , e <sub>3</sub> , e <sub>7</sub>	P <sub>1</sub> , P <sub>3</sub> , P <sub>4</sub>
e <sub>6</sub>	e <sub>2</sub> , e <sub>3</sub> ,	P <sub>1</sub> , P <sub>2</sub> , P <sub>4</sub>
e <sub>7</sub>	e <sub>2</sub> , e <sub>3</sub> , e <sub>5</sub> , e <sub>8</sub>	P <sub>1</sub> , P <sub>2</sub> , P <sub>4</sub>
e <sub>8</sub>	e <sub>2</sub> , e <sub>3</sub> , e <sub>5</sub> , e <sub>7</sub>	P <sub>1</sub> , P <sub>2</sub> , P <sub>3</sub>

5.4. táblázat Az egyes műveletek kizárása

Az 5.4. táblázatból az 5.13. ábra - feltételeket nem tartalmazó - kompatibilitási félmátrixa határozható meg. A kizárási feltételek „☒” jelölést kaptak, mert a továbbiakban ezeket tekintem előírásnak.

e <sub>2</sub>	✓						
e <sub>3</sub>	✓	☒					
e <sub>4</sub>	☒	✓	✓				
e <sub>5</sub>	☒	☒	☒	☒			
e <sub>6</sub>	☒	☒	☒	☒	☒		
e <sub>7</sub>	☒	☒	☒	☒	☒	✓	
e <sub>8</sub>	☒	☒	☒	☒	☒	☒	☒
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>	e <sub>6</sub>	e <sub>7</sub>

5.13. ábra A cosinus egyenlethez tartozó kompatibilitási félmátrix

Az 5.13. ábra alapján az alábbi maximális kompatibilitási osztályok határozhatók meg:

$$(e_1, e_2)(e_1, e_3)(e_2, e_4)(e_3, e_4) (e_5) (e_6, e_7)(e_8)$$

A    B    C    D    E    F    G

Az osztályok alapján elkészített lefedési táblát mutatja az 5.14. ábra. Látható, hogy a P<sub>1</sub>-hez tartozó elemi műveletek között vannak átlapolódók, ezért további kizárásokra lehet szükség. Az egymást átfedő P<sub>1</sub>-hez tartozó elemi műveleteket tartalmazó osztályokat vizsgálva először azt érdemes ellenőrizni, hogy egész osztály elhagyható-

e. Hagyjuk el a B és C osztályokat. Ezt a kompatibilitási félmátrixban úgy jelölhetjük, hogy „X”-el megjelöljük az  $e_1$ - $e_3$  és  $e_2$ - $e_4$  műveleteket. Mivel a tábla feltételeket nem tartalmaz, ezért a zártság nem sérül ezeknek az osztályoknak az elhagyásával.

Műveletvégző típusa	P <sub>1</sub>				P <sub>2</sub>	P <sub>3</sub>		P <sub>4</sub>
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>	e <sub>6</sub>	e <sub>7</sub>	e <sub>8</sub>
A	X	X						
B	X		X					
C		X		X				
D			X	X				
E					X			
F						X	X	
G								X

5.14. ábra A maximális kompatibilitási osztályokhoz tartozó lefedési tábla

A módosított kompatibilitási félmátrixot mutatja az 5.15. ábra.

e <sub>2</sub>		✓					
e <sub>3</sub>	X	⊗					
e <sub>4</sub>	⊗	X	✓				
e <sub>5</sub>	⊗	⊗	⊗	⊗			
e <sub>6</sub>	⊗	⊗	⊗	⊗	⊗		
e <sub>7</sub>	⊗	⊗	⊗	⊗	⊗	✓	
e <sub>8</sub>	⊗	⊗	⊗	⊗	⊗	⊗	⊗
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>	e <sub>6</sub>	e <sub>7</sub>

5.15. ábra Módosított kompatibilitási félmátrix

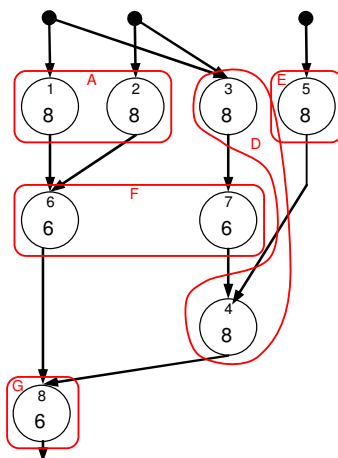
A módosítás után kapott maximális kompatibilitási osztályok a következők:

$$(e_1, e_2)(e_3, e_4) (e_5) (e_6, e_7)(e_8)$$

A      D    E    F    G

A módosítás után kapott osztályok már diszjunktak. A diszjunkt osztályok alapján elvégzett allokáció eredményét mutatja az 5.16. ábra. Az eredményt összehasonlítva a 4.6. ábra allokált grájával megállapíthatjuk, hogy a javasolt új módszerrel is sikerült a korábbi allokációs eredményt megkapnunk. Természetesen, ha a specifikáció során más feltételeket is figyelembe kell venni, akkor az eredményünk már különbözni fog. Az osztályok elhagyásakor több lehetőségből is választhattunk. Ha az osztályok elhagyásakor más lehetőséget választunk, akkor az allokáció eredménye különbözni

fog, de ez jelen esetben a felhasználandó műveletvégző egységek darabszámát (költségét) nem befolyásolja.



**5.16. ábra Allokáció a módosított kompatibilitási félmátrix szerint**

Az 5.14. ábra lefedési táblája alapján az eredeti kompatibilitási osztályokból az A és D osztályokat elhagyva, de a B és C osztályokat változatlanul hagyva ugyanilyen bonyolultságú megoldást kapunk. Ebben az esetben az eredeti kompatibilitási félmátrix az 5.17. ábra szerint alakul.

e <sub>2</sub>	<b>X</b>						
e <sub>3</sub>	✓	⊗					
e <sub>4</sub>	⊗	✓	<b>X</b>				
e <sub>5</sub>	⊗	⊗	⊗	⊗			
e <sub>6</sub>	⊗	⊗	⊗	⊗	⊗		
e <sub>7</sub>	⊗	⊗	⊗	⊗	⊗	✓	
e <sub>8</sub>	⊗	⊗	⊗	⊗	⊗	⊗	⊗
	e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>	e <sub>6</sub>	e <sub>7</sub>

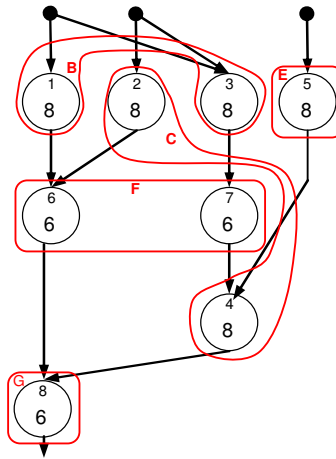
**5.17. ábra Kompatibilitási félmátrix A és D osztályok kizárásával**

A diszjunkt kompatibilitási osztályok:

$$(e_1, e_3)(e_2, e_4) (e_5) (e_6, e_7)(e_8)$$

B      C      E      F      G

A kapott osztályok alapján allokált elemi műveleti gráfot az 5.18. ábra mutatja. Megfigyelhető, hogy a  $P_1$  típusú műveletvégző egységből most is két példányra van szükség.



**5.18. ábra** Allokált műveleti gráf a B és C osztályokkal

A fejezetben bemutatott módszert és algoritmust, illetve használatukat a 3. tézisben fogalmaztam meg.

### **5.5. Harmadik 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]

## 6. 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 téziszhez kidolgozott becslés elsősorban a HLS eszközökhöz készült, de megfelelő paraméterezéssel alkalmazható rendszerfelületesi 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 téziszben 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 téziszben 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 téziszben 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 projekteknél.

## 7. Az értekezésben használt rövidítések jegyzéke

Az alábbiakban ABC szerint rendezve összefoglalom az értekezésben használt rövidítéseket

ALAP	As Least As Possible
ASAP	As Soon As Possible
CAN	Controller Area Network
CLK	Clock
CRC	Cyclic Redundancy Check
DFG	Data Flow Graph
EOG	Elementary Operation Graph
FFT	Fast Fourier Transformation
FPGA	Field Programmable Gate Array
HLS	High Level Synthesis
HW	Hardware
I <sup>2</sup> C	Inter-IC bus
IC	Integrated Circuit
IDE	Identifier Extension
IP	Intelligent processor
OSI	Open System Interconnection
PCI	Peripheral Component Interconnect
PCI-E	Peripheral Component Interconnect-Express
PLC	Programmable Logic Controller
RB	Reserved Bit
SATA	Serial Advanced Technology Attachment
SCL	Serial Clock
SDA	Serial Data
SDI	Serial Data Input
SDO	Serial Data Output
SoC	System on Chip
SOF	Start Of Frame
SPI	Serial Peripheral Interface
SS	Slave Select
SW	Software
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration

## 8. 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](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

**Az értekezés témájához is kapcsolódó publikációim****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", 2012, Periodica Polytechnica, 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

**Könyv, Felsőoktatási tankönyv**

[S4.] Pilászy György, "Gyakorlati anyagok a Beágyazott irányító rendszerek tantárgy gyakorlataihoz - A CAN és LIN protokollok", Elektronikusan közzétett egyetemi jegyzet

**Konferenciaközlemény**

[S5.] 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

[S6.] 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)

[S7.] Komáromi Zoltán, Pilászy György, Móczár Géza, "Központi irányítóegység moduláris, elosztott intelligenciájú irányítástechnikai 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. 172-178. Paper &. Vol. 1

[S8.] Pilászy György, Komáromi Zoltán, Móczár Géza, "Elosztott irányítórendszer napkollektoros energetikai 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

[S9.] 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

#### **Kutatási jelentés (belső) /Tudományos**

[S10.] Arató Péter, Móczár Géza, Pilászy György és társai, "Feladatfüggő felépítésű többprocesszoros célrendszerek szintézis algoritmusainak kutatása.: OTKA T72611.", (2011)