Optimális és közelítő algoritmusok a hardver–szoftver partícionálásra

Kidolgoztuk a rendszerszintű szintézis egyik legfontosabb optimalizációs problémájának, a hardver–szoftver partícionálásnak egy új gráfelméleti modelljét. Az irodalomban szereplő korábbi modellekkel szemben matematikailag jól kezelhető és gyakorlatban használható algoritmusok készíthetők hozzá.
A megfogalmazott partícionálási feladat ugyan NP–teljes, mivel speciális esetként tartalmazza az ismert, NP–teljes ún. hátizsák–feladatot, de a modell egyszerűsége lehetővé teszi, hogy gyakorlati példákon egzakt optimumot keressünk. Megfogalmaztuk a feladatot egészértékű optimalizációs problémaként (ILP), majd egy általános ILP–megoldóval megoldottuk.
Bár több száz komponensből álló rendszert sikerült így optimálisan partícionálni, a futási idő gyorsan növekszik a probléma méretével. Ezért az általános ILP megoldó helyett egy saját, branch–and–bound technikán alapuló egzakt algoritmust adtunk, melyet a konkrét problémára optimalizáltunk. Az új módszerrel nagyméretű benchmark feladatokon átlagosan kétszeres gyorsítást értünk el.
A branch–and–bound algoritmus további előnye, hogy egy kezdeti közelítő megoldást felhasználva lényegesen gyorsabban találja meg az optimumot, mint minden előzetes információ nélkül. Ez indokolta, hogy a partícionálási problémára különböző közelítő heurisztikákat készítsünk, illetve létező heurisztikákat használjunk fel a branch–and–bound kezdeti megoldásának előállítására. Az alkalmazott heurisztikák az alábbiak:

- Genetikus algoritmus, mely abban tér el a szokásos genetikus algoritmusoktól, hogy a populáció a korlátokat sértő egyedeket is tartalmazhat, melyeket azonban az értékelő függvény büntet. Így a genetikus operációk korlátozás nélkül alkalmazhatók, ami hatékonyabb optimalizálást tesz lehetővé.
- Kombinatorikus, a minimális vágás keresésén alapuló algoritmus, melynek lényege, hogy az eredeti gráfból különböző súlyozásokkal előálló segédgráfok minimális vágásai közül a korlátoknak megfelelő legjobb vágást választja ki.
- Csúcsok összevonásán (clustering) alapuló algoritmus, melynek lényege, hogy a gráf csúcsait egy heurisztikus sorrendben összehúzza addig, míg olyan méretű gráfhoz jut, amely az előbb leírt egzakt megoldó algoritmussal optimálisan is hatékonyan megoldható.

A heurisztikákat a gyakorlatban is implementáltuk, és az ezekkel indított branch-and-bound megoldó számos példán összehasonlítva az előzőeknél hatékonyabbnak bizonyult.

Új komponens alapú hardver-szoftver együttes tervezési módszertan

Az új módszertan lehetővé teszi összetett hardver-szoftver rendszerek tervezésének korai szakaszában egy szimulálható prototípus egyszerű létrehozását. Ez a megközelítés a szoftver technológia területén elterjedt komponens alapú tervezést általánosítja hardver komponensekre is, így a tervezőnek csak a funkcionalitásra kell koncentrálnia. Az új koncepció része, hogy a hardvertervezésben megszokott nem funkcionális követelmények (pl. valós idejű korlátok) is kifejezhetők. A partícionáló algoritmus

feladata, hogy az előírt követelmények betartásával a legkedvezőbb implementációkat kiválassza a megadott viselkedéshez.
A komponens alapú partícionálás újfajta konzisztencia-problémákat is felvet, melyek felismerésére és kezelésére módszereket adtunk.
A teljes rendszer implementálásra került egy JAVA alapú vizuális fejlesztőeszközbe, amely lehetővé teszi hardver és szoftver komponensek grafikus magas szintű kompozícióját, a kialakult rendszer szimulációját, valamint automatikus partícionálást és konzisztencia–ellenőrzést végez.
A fenti módszertan és fejlesztőeszköz alkalmazhatóságát esettanulmánnyal támasztottuk alá.

Szisztematikus eljárás magas szintű programnyelven megirt algoritmusból uniformizált hardver felépítés közvetlen generálására

A kutatás során megvizsgáltuk, hogy a gyakorlatban jól bevált szoftvertechnológiai specifikációs nyelvek (pl. UML) miként terjeszthetők ki a rendszerszintű szintézis feladatok formális leírására. A szoftver tervezési folyamatban az implementációt közvetlenül megelőző fázisban (pl. *use-case* leírások) már rendelkezésre állnak mindazok az ismeretek, amelyek alapján egy magas szintű nyelven (pl. C) a programírás megkezdhető. A rendszer szintű szintézis folyamatban célszerű a lehető legkésőbbi fázisban eldönteni, hogy egy adott rendszerelem megvalósítása hardver vagy szoftver történik-e. Emiatt az UML-ből kiinduló specifikációs folyamat során azonos módon kezelhetők a rendszerelemek a későbbi döntéstől függetlenül. A magas szintű nyelvi leírásig tehát a későbbiekben hardverként megvalósuló rendszerelemek esetén is alkalmazható az UML-ből kiinduló specifikációs folyamat. Az eltérés a további tervezési lépésekben van: a magas szintű nyelvi leírás alapján nem a programot kell implementálni, hanem a hardver tervezését kell elvégezni.
Olyan módszert dolgoztunk ki, amely alkalmazás-specifikus hardvert határoz meg magas szintű (C) programozási nyelven írt forráskód alapján. Digitális sorrendi hálózatokat feleltet meg minden egyes utasításnak, műveletnek és változó elérésnek. Az így kapott állapotgépek egymást indítják az eredeti forráskódnak megfelelően. Az eredeti erős sorrendiséget a fordítási időben elvégzett kódvizsgálat oldja fel úgy, hogy független műveleti blokkok párhuzamosan kerülnek indításra. Erőforrás-foglalási és ütemezési lépések segítik az adott feltételeknek legjobban megfelelő sebesség-gazdaságosság kompromisszum megtalálását. A kidolgozott modell és eljárás főbb funkcionális jellemzői:
- A rendszer-szintű tervezés során a szoftver és a hardver rendszerelemek egyaránt egy közös leíró nyelv segítségével specifikálhatók
- Kiválasztható szoftver specifikációs részeket hardverré lehet lefordítani anélkül, hogy a forráskódban lényeges változtatást kellene tenni.
- A hardver rendszerelemek uniformizáltsága és a szoftver leírásból való közvetlen származtathatósága révén a hardver tervezés folyamata jelentősen egyszerűsödik.

A szakirodalomban található hasonló célú C-alapú módszerek a felhasználóktól különböző, nem szabványos C ismeretet várnak el. Csak egy részhalmazát engedik a C

változótípusoknak használni, a forráskódot a hardver fordítás előtt módosítani kell, hogy az adott fordító feltételeinek megfeleljen. Különleges utasításokat kell használni a párhuzamos végrehajtás elérése érdekében. Az optimumkeresés sebesség-áramkörfelület viszonylatban nincs erőforrás foglalással és ütemezéssel támogatva. Az általunk kidolgozott eljárás lényeges jellemzői:
- Nincs szükség az eredeti forráskód szintaxisának módosítására a fordítás előtt
- Az alkalmazott változókezelés megadja a lehetőséget akár összetett adatszerkezetek használatára is (pl. mutatók, vagy rekordok tömbjei).
- A független műveleti blokkok párhuzamos futtatását a fordítási időben elvégzett kódvizsgálat biztosítja.
- Erőforrás foglalás és ütemezés révén lehetővé válik a sebesség és gazdaságosság közötti kompromisszum beállítása.

A módszer állapotgépeken alapul, amelyek az egyes C utasításoknak felelnek meg, mint *while, do, for, if.* Ezen állapotgépek (utasítás egységek) vezénylik a megfelelő tevékenységeket, mint pl. feltételek kiértékelése, ciklusok indítása a kiértékelt feltétel-kiértékelése eredménye alapján, és végül a következő utasítás egység indítása. Az eljárások (alprogramok) definiciója változókkal együtt procedúra egységek létrehozásával történik, melyek egy állapotgépből és a helyi változók illetve hívási paraméterek számára tárolóhelyekből állnak. Más műveletek, mint például összehasonlítás, összeadás, stb műveleti egységekre képeződnek le, melyek szintén állapotgép alapúak és az utasítás egységekével azonos módon aktivizálhatók. A utasítás, műveleti és procedúra egységek az eredeti forráskód alapján láncot alkotnak a *trigger* és *ready* kivezetéseik segítségével. Ez az egyszálú láncolat a fordítási időben zajló kódvizsgálat segítségével úgy módosul, hogy a függetlenül aktivizálható műveletsoroknak megfelelő láncszegmensek párhuzamosan kerülnek bekötésre. Az eljárás végén egy VHDL kód generálódik automatikusan, amelynek alapján a szoftverrel meghatározott célhardver megvalósítható.

Új módszer komplex és adaptálható funkcionális egységekből (IP-kből) történő rendszerszintézisre

Új szisztematikus eljárást fejlesztettünk ki előre megadott és kiadódó komplex funkcionális egységekből (IP-kből) történő rendszerfelépítésére, amely egyaránt lehetővé teszi az időbeli átfedésen alapuló allokáció optimális és közelítő végrehajtását, biztosítva a lehető legkevesebb funkcionális egység felhasználását.
A funkcionális egységeket az általuk megvalósítható művelet típusok definiálják. Az új algoritmus egyrészt használható előre megadott funkcionális egységekre történő allokációra, másrészt segíti az adott feladat szempontjából optimális funkcionális egységek megtalálását figyelembe véve a költséget és a funkcionális egységek ismételt felhasználását (reuse). Az algoritmus az időbeli átfedést inkompatibilitási relációként kezeli, miáltal a maximális kompatibilitási osztályokból kiindulva az optimális megoldást szolgáltatja. A maximális kompatibilitási osztályok célszerű redukciójára kidolgozott algoritmus segítségével gyors közelítő megoldás is nyerhető.

A módszer részét képezi az az új szisztematikus eljárás is, amely automatikusan megvalósítja a magas szintű szintézis során kiadódó tetszőleges bonyolultságú pipeline működésű funkcionális egységek és a közöttük lévő adatkapcsolatok VHDL nyelven történő leírását. Ezáltal lehetővé válik az FPGA áramkörökre vagy VLSI építőelemekre történő közvetlen szintetizálás kommersz „silicon compiler" jellegű tervező rendszerekkel. A puffer, multiplexer és demultiplexer egységeket megvalósító entitások automatikusan keletkeznek, a funkcionális egységeket leíró entitások váza generálódik, a funkcionalitást a tervezőnek kell definiálnia. Az adatfolyam specifikációja strukturális leírással, a központi elven működő vezérlésé pedig viselkedési leírással történik. Az így kapott egység egyetlen komplex műveleteként is értelmezhető hierarchikus szintézis esetén.

Új algoritmust fejlesztettünk ki a feltételes elágazások tetszőleges mélységű kezelésére pipeline rendszerek esetén. Az elemi műveletek reprezentációját kiterjesztettük feltételes elágazást tartalmazó struktúrák leírására szolgáló vezérlő jellemzőkkel, amelyek alapján algoritmust dolgoztunk ki a feltételes ágak tetszőleges mélységű hierarchiájának detektálására. Kiterjesztettük az elemi műveletek kompatibilitását számító algoritmust alternáló feltételes ágakban lévő pipeline műveletek esetére. Ennek eredményeképpen szakirodalmi eljárásokhoz képest kisebb számítási igényű általánosabb eljárást kaptunk. Az új kompatibilitási számításmódnak megfelelően módosítottuk a korábbi OTKA támogatással létrehozott PIPE tervező rendszerünk genetikus és force-directed ütemező algoritmusait, amelyek így már hatékonyan kezelik a tetszőleges mélységű feltételes elágazásokat. Az új algoritmusokat használó PIPE rendszerrel a korábbi változathoz képest 9-61%-os költségcsökkenést lehetett elérni a nemzetközi szakirodalomban használt vezérlés-domináns benchmark feladatokon.