Adatfolyam alapú RACER tömbprocesszor és algoritmus implementációs módszerek valamint azok alkalmazásai parallel, heterogén számítási architektúrákra

PhD disszertáció tézisei

Rák Ádám

Témavezető:
Dr. Cserey György

Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Budapest, 2014

1. Bevezetés, kitűzött feladatok

Az utóbbi években a számítástechnikában megindult az az új irány, miszerint már nem a processzorok órajelét növelik, hanem a magok és végrehajtó egységek számát. Ez a trend manapság a hálózati eszközöktől kezdve, az asztali számítógépeken keresztül, még a mobil telefonokban is megjelenik. Ennek fő oka fizikai törvényekre vezethető vissza, ugyanis a mikrocsipek miniatűrizálása és az órajel növelése oda vezetett, hogy egy processzor két távolabbi pontja közötti kommunikáció túl sok órajelnyi ideig tart. E késleltetésben pedig leginkább a csipen levő vezetékezések és fémes összeköttetések játszanak nagy szerepet. Az órajel további növelése így egyszerre nemcsak hogy a szilícium kapcsolási sebessége által meghatározott limitbe ütközik, de még növeli is a késleltetés tapasztalt értékét. A túl nagy késleltetés szükségszerűvé teszi az architektúra fragmentálódását több végrehajtó egységre és magra.

A Moore törvény szerint a digitális integrált elektronikának az egy tranzisztorra levetített gyártási költsége egyre olcsóbb. Ez tovább segíti ezt az irányt, ugyanis egy jól megtervezett sokprocesszoros rendszerben a magok számának növelése egyszerű feladat. Így ugyanolyan áron nem csak egyre több tranzisztort, hanem egyre több magot, azaz a Moore törvénnyel megegyező ütemben növő nyers számítási erőt is kapunk.

A trendet jelenleg a Grafikus Feldolgozó Egységek (GPU) vezetik, melyek a 2014-es év folyamán már elérték, sőt meg is haladták a 5760 mag / mikrocsip értéket. Ilyen többmagos vagy sokmagos rendszerek például: DSP, FPGA, CELL, GPU, de a trend jellemző a multimédiás célspecifikus beágyazott processzo-

rokra is.

A hardver gyors fejlődése mellett azonban felvetődik a kérdés hogy ezeket az architektúrákat hogyan programozzuk hatékonyan. A sokprocesszoros rendszerek nemcsak nagyobb változatosságot mutatnak, mint a hagyományos elődeik, hanem alapvetően új programozásbeli megközelítést igényelnek. Ahhoz, hogy minél több feldolgozó egységet lehessen integrálni egy processzorba, a számítást végző magokat jelentősen leegyszerűsítették, gyakorlatilag eldobták az előző 20 év eredményeinek nagy részét, ami egy processzor mag optimalizálására irányult. Így a határ elmosódik egy egyszerű számoló egység, például Lebegőpontos Egység (FPU) és egy teljes értékű mag között. A sokprocesszoros és hagyományos rendszerek radikális különbségeit példázza, hogy ha sokprocesszoros rendszereket szigorúan soros programfuttatásra használjuk minden párhuzamosság nélkül, akkor gyakran $100-szor$ lassabb sebességeket tapasztalunk mint egy nem párhuzamos CPU esetén. Az ilyen új rendszerek szabványos programozására született az OpenCL programozási nyelv, azonban ez egy alacsony szintű C nyelv, mely az algoritmus párhuzamosításának teljes problémáját a programozóra hárítja.

Egy algoritmus implementációja esetén a célarchitektúra alapos ismerete szükséges. Mely a tapasztalok szerint még OpenCL szabvány nyelv használata esetén is elengedhetetlen, ugyanis a futási időben egy nagyságrendet jelenthetnek a legapróbb optimalizációs megoldások is. A problémát tovább nehezíti, hogy még egy adott gyártón belül is (Nvidia, AMD-ATI) évente változik az architektúra felépítése, két-évente pedig alapvető újratervezésen megy keresztül. Továbbá nem ritka, hogy a gyártó önmaga is

küzd a saját termékének megismerésével és kiaknázásával.

Jelentős igény lenne olyan megoldásokra, melyek az algoritmusok párhuzamos implementációját segítik, matematikailag megalapozott módszerekkel. Ebbe bele tartozik az az irány is, mely gépi tanulással segíti egy új architektúra megismerését.

Ezek alapján feladatomnak tekintettem általános algoritmusosztályok párhuzamosításának vizsgálatát és matematikai analízisét, valamint eredményeim néhány nehéz eseten történő bemutatását.

A fejlődési irány egyértelmű, a processzorok számának exponenciális növekedésére továbbra is számíthatunk az előre látható fejlődési trendekben. Ugyanakkor az egyes, egymást követő architektúrák eltérése nem csak mennyiségi processzorszám, hanem architektúrális változásokban is megmutatkozik. A fejlődés nem csak a pusztán nagyobb számú feldolgozó egység felé vezet, hanem a hatékonyabb és optimalizáltabb működés, az egyidőben elérhető, felületre eső számítási teljesítmény növekedéséhez is. Amennyiben megvizsgáljuk a különböző, akár csak elméletként megszületett párhuzamos architektúrákat, láthatjuk, hogy ezek kisebb-nagyobb kompromisszumok mellett, arra törekednek, hogy az általános célú használat mellett az egységnyi területre jutó számítási teljesítményt maximalizálják. Ezek a kompromisszum és hátrányok az egyes legelterjedtebb architektúrákra a következőek:

A CPU-k klasszikusan hierarchikus cache segítségével elrejtik a memória írás és olvasás nehézségeit, azonban ez, különösen a magok számának emelkedésével, tarthatatlanul növeli a cache memória felületarányát a számítást végző szilicium felülettel szemben. Ez egy jó egyensúly a kevésbé számításigényes fel-

adatok esetén, azonban tudományos, vagy grafikai számításokra meglehetősen pazarló.

DSP: digitális jelfeldolgozó processzor. Ezek az eszközök nagyon hasonlítanak a CPU-khoz, inkább csak a képességekre helyezett hangsúlyban különböznek manapság. Leginkább ezek az eszközök alacsony áramfogyasztás és versenyképes ár mellett hatékonyan képesek jelfeldolgozási algoritmusokat futtatni (FFT, mátrix-vektor műveletek). A chip felülete (azaz a chip gyártási költsége) optimalizációja érdekében gyakran sokkal kisebb cache memóriát tartalmaznak, mint a CPU-k, ezért itt a rendszer memória elérésének mintázata sokkal kötöttebb ha ki akarjuk használni a rendelkezésre álló sávszélességet.

A GPU-k (Graphics Processing Unit) elsősorban grafikára szánt architecturája vektoros SIMD (Single Instruction Multiply Data) nagyon erős megkötést ad a végrehajtási szálakra, miszerint egy csoporton belül minden szálnak ugyanazt a műveletet kell elvégeznie, és szomszédos memóriát kell hogy olvasson. Ennek köszönhetően mind a memória sávszélesség, mind a szilicum felület számításra történő kihasználtsága nagyon nagy. Azonban ez az architektúra teljesen a programozóra hárítja a memória hatékony használatának feladatát, nem rejti el a részleteket és nem oldja meg az ezzel kapcsolatos problémákat, ellentétben a CPU esetével.

Cell BE (Cell Broadband Engine): hibrid architektúra, melyben helyet foglal egy klasszikus PowerPC CPU processzor, összekötve úgynevezett SPU-kel. Az SPU-k nagyon leegyszerűsített számító egységek, melyekben relatíven nagy helyi memória foglal helyet a chip-en. Itt minden technikai apróság megoldása a programozó feladata, a végrehajtó pipeline megfelelő etetésétől,

egészen a memória műveletek belső logikájáig, ugyanis itt csak közvetett memória elérés lehetséges a helyi memórián keresztül.

FPGA (Field Programmable Gate Array): helyben programozható kapu hálózat. Ez egy olyan architektúra mely képes bizonyos tág korlátokon belül tetszőleges logikai áramkört megvalósítani. Általában az itt megvalósított áramkör viszonylag hatékony, ugyanis a chip-en található kapcsolóáramkörök segítségével az FPGA fizikálisan valósítja meg a kívánt áramkört, összeköti alkotóelemekből. Mivel az FPGA-nál a logikai áramkör közvetlenül a feladathoz igazítható, ezért itt lehet a legjobban kihasználni a rendelkezésre álló végrehajtó egységeket. Azonban a hatalmas flexibilitás ára a kicsi végrehajtó egység sűrűség a chip felületén, ugyanis a kapcsolóáramkörök és az univerzális huzalozás nagy helyet foglalnak el.

Systolic Array: ez a klasszikus topologikus tömbprocesszor elrendezés gyakorlatilag csak végrehajtó (számoló) egységeket tartalmaz, összeadó és szorzó áramköröket, melyek általában valami lineáris algebrai műveletet oldanak meg párhuzamosan. Használati köre rendkívül szűk, ugyanis a topológiája a végrehajtandó algoritmusra specifikus. Nem tartalmaz se memória architektúrát, se program vezérlési szerkezetet, ezeket egy másik hozzákapcsolt rendszernek kell nyújtania. Itt a flexibilitást teljesen feláldozzák a hatékonyságnak, ugyanis a végrehajtó egységek jó része 100%-ban ki van használva a számítás során, és a szilicum felülete gyakorlatilag csak végrehajtó egységeket tartalmaz.

CNN: csak szűk feladatkörben hatékony (kisfelbontású képfeldolgozási algoritmusok szürkeárnyalatos képen), azonban ebben a feladatkörben lokális képfeldolgozási operátorokra rendkívül nagy sebességgel és kis áramfogyasztással teljesít. Minden kép-

pont feldolgozása társítva van egy feldolgozó egységhez, azonban a feldolgozás analóg módon zajlik és ehhez nagyon kevés helyi memória áll rendelkezésre. A központi memória elérése a rendszer belső sebességéhez képest rendkívül lassú, és a képpontok digitalizálását is igényli. 2D topologikus kis memóriát igénylő lokális számításokra van optimalizálva.

Ezek alapján feladatomnak tekintettem egy olyan számítási architektúra megtervezését (RACER architektúra), amely nem rendelkezik a korábbi párhuzamos architektúrák hátrányaival, Turing-teljes valamint teljesen általános célú algoritmusok hatékony megoldására használható, szem előtt tartva a felületre eső számítási teljesítmény maximalizálását.

2. A vizsgálatok módszerei

Kutatásaim során több disciplinának az eszköztárát is felhasználtam, ezek közül a legfontosabb volt az automatikus párhuzamosítás elmélete. Az automatikus párhuzamosítás alapja, hogy olyan ciklus szerkezeteket keresünk a programban, melyek újrafogalmazhatóak párhuzamos formában. Ehhez úgynevezett polihedrális leírást használunk, mely során a statikus ciklus szerkezetet sokdimenziós diszkrét téren ábrázoljuk és affin geometriai transzformációkkal alakítjuk megfelelő alakra. Ezt az elméletet egészítettem ki, úgy hogy dinamikus vezérlőszerkezeteket is tudjon kezelni. Ehhez kapcsolódóan adatfolyam és kontrollfolyam programleírást használtam, melyek lehetővé teszik a programkód hatékony automatikus kezelését és optimalizációját. Megismerkedtem a jelenleg két legjobban használt nyílt forráskódú fordítóprogram (GCC, LLVM) belső működésével, és az általuk használt optimalizációs algoritmusokkal.

Munkámhoz szükséges volt részleteiben megismernem, a jelenleg legjobban elterjedt általános célúan programozható GPU architektúrákat:

- NVIDIA GeForce8
- NVIDIA Fermi
- NVIDIA Kepler
- AMD(ATI) R800(Evergreen)
- AMD(ATI) R900(NI Cayman)
- AMD(ATI) R1000(Southern Islands GCN)

Az AMD architektúrákhoz a gyártó rendelkezésemre bocsátotta a gépi kódról, a felépítésről, továbbá az általános célú működéshez szükséges hardver szintű programozásról szóló dokumentációkat. Az NVIDIA esetén, ezeket az információkra visszafejtéssel, és alapos mérésekkel sikerült szert tennem.

A RACER architektúra tervezéséhez általános digitális processzor tervezési ismereteket használtam fel, mint például a pipeline tervezése, digitális szintézis, órajel huzalozási módszerek: H fraktál megolás, rezonáns háló, aszinkron háló. Törekedtem arra, hogy a RACER minél több része már meglévő IP-mag könyvtárból felhasználható legyen, így ezekkel is vázlatosan megismerkedtem. Felhasználtam a nehéz algoritmusok (például

videó tömörítés, ritka-mátrix algebra) implementációja során szerzett tapasztalataimat, kombinálva a tömbprocesszorok és szisztolikus tömbök lokális adatfeldolgozási módszereivel. Részletesen megismertem a memória áramkörök, valamint a processzor memória kommunikáció működését és limitációit, melyek kiküszöbölése fontos szerepet kap a RACER architektúrában. Megismerkedtem a digitális áramkörök hatékony viselkedés szintű szimulációjával és magas szintű tervezésével VHDL és Verilog nyelven.

Egy GPU-ra optimalizált kvantum kémiai két-elektron integrál számító szoftverben eredményeim egy része felhasználásra került. Az algoritmus masszív párhuzamosítása és GPU-ra történő optimalizációja érdekében implementáltam egy specializált fordító programot. A fordítóprogram szimbolikus levezetéssel képes az integrálokat kifejteni, a levezetésben felhasznált alapul vett algoritmusok, melyeket alaposan megismertem:

- Boys
- Pople-Hehre
- Obara-Saika-Schlegel
- Head-Gordon-Pople
- McMurchie-Davidson
- Az ezekre épülő MD-PRISM, HGP-PRISM

Ezen algoritmusok megértéséhez részletesen megismerkedtem a numerikus kvantum kémia matematikai eszközeivel és

módszereivel, különös tekintettel a Gauss bázisok integráljainak számolására, és az általánosított bra-ket matematikájára. Megismerkedtem az integrálokat felhasználó elektrosztatikus potenciál számoló módszerekkel:

- Önkonzisztens Mező (Self Consistent Field : SCF) Hartree-Fock
- Sűrűség Függvény Elmélet (Density Function Theory : DFT)
- Moller-Plesset Perturbációs Elmélet (Moller-Plesset Perturbation Theory : MPPT)
- Konfiguráció interakció (Configuration Interaction : CI)
- Csatolt Csoport számítás (Coupled Cluster computation : CC)

3. Új tudományos eredmények

1. Tézis: *Megmutattam, hogy sokmagos rendszerek programozása során a klasszikus statikus polihedrális reprezentáción felül, megvalósítható a dinamikus ciklusok és dinamikus vezérlő szerkezetek polihedrális reprezentációja is. Dinamikus polihedronok esetén formalizmust adtam arra, hogy hogyan érdemes kezelni a memória hozzáférési mintázatot. Definiáltam azokat az algoritmus osztályokat, melyeket hatékonyan lehet kezelni módszereimmel. [7, 8, 9]*

Új matematikai formalizmust adtam a programok dinamikus vezérlő szerkezetének magas szintű kezelésére. Melyben a dinamikus vezérlő szerkezeteket visszavezettem végtelen statikus szerkezetekre és speciális dinamikus függőségekre. A végtelen határok azért szükségesek, mert statikus szempontból parametrikusak a dinamikus határok, így felülbecsülhetőek végtelennel. Mivel a függőségek dinamikusak lehetnek, ez a párhuzamosítási lépést a futási időben teszi szükségessé. Így ebben az elméletben a program futásának része az úgynevezett „scheduling„ mely során eldől, hogy mely számítások milyen erőforráson és mikor hajtódnak végre. H.264 videó-enkóder implementáció részleteinek példáival demonstráltam a bemutatott elméletet.

2. Tézis: *Egy új adatfolyam elvű párhuzamos számítógépes architektúrát (RACER) terveztem, melyben a funkciók az eddigi megoldásoknál hatékonyabban vannak szétosztva a memória és a feldolgozó egységek között és ennek a megvalósításához a párhuzamosítás kiterjed a memóriára is. [14]*

Megtervezem a RACER adatfolyam vezérelt számítási architektúra működéséhez szükséges elemeket. Az architektúrában mind a program, mind az adat folyamszerűen halad végig egy tömbprocesszorban. A program által előre kialakított szerkezet dolgozza fel az utána jövő adatfolyamot. Az adatfolyamnak dinamikus vezérlése is lehetséges, elágazás, ciklus, egyesítés, szétosztás. Nagyon fontos egység az ehhez kapcsolódó memória rendszer, mely eltérően a hagyományos memóriától, tartalmaz számítást végző elemeket is, különös tekintettel a összehasonlítási aritmetikákra. Így a memóriában megvalósítható az aktuális algoritmusnak megfelelő rendezés, mely biztosítja a tömbprocesszor folytonos adattal történő ellátását.

2.1. Megmutattam, hogy vezérlőelektronika és huzalozás egyszerűsége és lokalitása miatt, integrált áramköri megvalósítás esetén a felület 57-72%-át a feldolgozó aritmetikai egységek foglalják el, ezzel adott felületen az egyik legmagasabb aritmetikai sűrűség érhető el a GPU architektúrákhoz viszonyítva.

A feldolgozó egységekhez az Aeroflex Gaisler FRFPU-1 IP core-t választottam. Egy feldolgozó egység megvalósításához 100000 kapu felhasználását számoltam 65nm és 90nm technológia használata mellett. Becslést adtam az útvonal meghatározó egységek megvalósításához szükséges kapuk számára is. Ezek az egységek nagyságrendileg a chip felület 20%-át foglalják el. A kapott adatok alapján a Cadence InCyte Chip Estimator programmal becslést adtam a teljes chip méretére és fogyasztására. Összehasonlítottam a RACER így becsült csúcsteljesítményét a

GPU csúcsteljesítményével, hogy bemutassam a lehetséges teljesítményelőnyt, amit a nagyobb számú feldolgozó egységek biztosítanak. Becsléseim alapján a RACER architektúra számítási egységekkel lefedett területe a teljes terület 57 és 72 százaléka között van.

2.2. Megmutattam hogy a RACER képes hatékonyan megoldani a Mandelbrot és Conway's életjáték algoritmusok végrehajtását is, miközben általános architektúra marad. Az implementált algoritmusokkal demonstráltam az architektúra működőképességét, valamint bizonyítottam, hogy az architektúra Turing-teljes.

Megtervezem egy lehetséges konkrét implementációt, és szimulációban bizonyítottam az architektúra működőképességét. A szimuláció alacsony szintű, de még nem gépi kód. Képes inputot fogadni, mely a program gráfreprezentációban történő adatfolyam leírása. Így a javasolt architektúrán ellenőrizhető lett konkrét algoritmusok futása.

3. Tézis: *Az általam kidolgozott módszereket felhasználtam kvantumkémiai integrál számítási algoritmusok gyorsítására és párhuzamosítására, különös tekintettel a Single Instruction Multiply Data (SIMD) architektúra kompatibilitásra. GPU architektúrára meta algoritmust (BRUSH) terveztem, ami kiválasztja az integrálokhoz tartozó optimális számítási útvonalat. Megmutattam, hogy speciális kontrakciók esetén konstans behelyettesítés és terjesztés hatékony az ilyen architektúrákon. [12, 1]*

Megtervezem és implementáltam egy kvantum kémiai integrátorra specializált fordítóprogramot, mely lehetővé teszi a SIMD típusú masszív párhuzamos architektúrák hatékony kihasználását. Kvantum kémiában a legfontosabb numerikus számítás, a két-elektron integrálok számítása, melyek hatékony párhuzamosításához fejlesztettem egy fordítóprogramot. A fordító bemenete a konkrét integrál probléma, melyet az eddig használt módszerekkel ellentétben statikusan kifejt, azaz az összes dinamikus vezérlőszerkezetet fordítási időben végrehajtja. Az optimális számítási utakat előre kiszámolja és kiválasztja. Az így kapott számítási gráfból, mely óriási számú aritmetikai műveletből áll, készül el a hardver specifikus gépi kód. Ezen átalakítások során különös figyelmet fordítottam az architektúra sajátságainak kihasználására. Ilyen például a többszintű memória szerkezet használata (regiszter-lokális-globális) az átmeneti értékek tárolására, vagy a SIMD párhuzamos feldolgozó magokhoz való optimalizációja.

4. Eredmények alkalmazási területei

Munkám során elkészült elméleti eredményeket a gyakorlati hasznosítás motiválta. A bemutatott algoritmusok valós alkalmazási területeken felmerülő problémákra adnak megoldást.

Az első téziscsoport eredményei algoritmus implementációk sokprocesszoros architektúrákra (GPU, FPGA) történő fordítását segíti, így a fejlesztő számára kevésbé szükségese az adott architektúra részletes ismerete.

Második téziscsoportom egy olyan architektúrát mutat be,

amelyik egy fizikai megvalósítás esetén számos területen kimagasló teljesítményt mutathat fel és kerülhet alkalmazásba. Ezek a területek a teljesség igénye nélkül: grafikus 3D megjelenítés, sugárkövetés (raytracing), stuktúrálatlan griden számolás, számítógépes játékok, nagy adatbázisok kezelése, lényegében minden GPU-n hatékonyan megoldható probléma.

A harmadik téziscsoportban bemutatott egy célalgoritmus általános feladatra használható. A kvantumkémiai integrálszámítás GPU-n történő gyorsítása a szintetikus molekulatervezést gyorsíthatja fel jelentősen lecsökkentve a néha több hetes futási időt.

5. Köszönetnyilvánítás

Köszönöm a Pázmány Péter Katolikus Egyetem, Információs Technológiai Kara Interdiszciplináris Műszaki Tudományok Doktori Iskolájának és a Doktori Iskola vezetőinek, Dr. Roska Tamás Professzor Úrnak valamint Dr. Szolgay Péter Professzor Úrnak a bátorítást és a tanácsokat, amely erőt adott a munkám során.

Szeretnék köszönetet mondani Cserey Györgynek, aki segített és támogatott mindenben, aki mindig töretlen lelkesedéssel és útmutatással irányított tanulmányaim során.

Köszönöm kollégáimnak akik tanáccsal láttak el, és akikkel mindig megbeszélhettem ötleteimet: Soós Gergely Balázs, Feldhoffer Gergely,

Tar Ákos, Veres József, Jákli Balázs, Sárkány Norbert, Tornai Gábor, Koller Miklós, Reguly István, Józsa Csaba, Horváth András, Stubendek Attila, Gergely Domonkos, Radványi Mihály, Fülöp Tamás, Zsedrovits Tamás, Nemes Csaba, Gaurav Gandhi.

Külön köszönöm a mindig segítőkész Vida Tivadarné mosolyát és áldozatos munkáját valamint a PPKE dékáni hivatalának és tanulmányi osztályának a gyakorlati és hivatalos dolgokban kedves segítségüket.

The support of grants Nos. TÁMOP-4.2.1/B-11/2-KMR-2011-0002 and TÁMOP-4.2.2/B-10/1-2010-0014 are gratefully acknowledged.

Nagyon köszönöm Annának, hogy sűrű feladatai mellett is időt szakított beszélgetéseinkre.

Végül, de mégis elsősorban köszönöm Édesanyámnak és Édesapámnak és családunknak a szeretetét, törődését és segítségét, amellyel mindig támogattak.

6. Publikációs lista

6.1. A szerző folyóiratbeli publikációi

[1] **A. Rák** and G. Cserey, „The BRUSH algorithm for two-electron integrals on GPU," *MATCH Communications in Mathematical and in Computer Chemistry*, 2014. submitted.

[2] **A. Rák** and G. Cserey, „Macromodeling of the memristor in SPICE," *IEEE Transactions on Computer-Aided Design of*

Integrated Circuits and Systems, vol. 29, no. 4, pp. 632–636, 2010.

[3] **A. Rák**, G. Gandhi, and G. Cserey, „Chua's circuit topology evolution using genetic algorithm," *International Journal of Bifurcation and Chaos*, vol. 20, no. 3, pp. 687–696, 2010.

[4] G. B. Soós, **A. Rák**, J. Veres, and G. Cserey, „GPU boosted CNN simulator library for graphical flow based programmability," *EURASIP Journal on Advances in Signal Processing*, 2009. Article ID 930619, 11 pages doi:10.1155/2009/930619.

[5] **A. Rák**, G. B. Soós, and G. Cserey, „Stochastic bitstream based CNN and its implementation on FPGA," *International Journal of Circuit Theory and Applications*, vol. 37, no. 4, pp. 587–612, 2009.

6.2. A szerző nemzetközi konferencia publikációi

[6] G. Cserey, **A. Rák**, B. Jákli, and T. Prodromakis, „Cellular neural networks with memristive cell devices," in *Proceedings of 17th IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2010*, (Athens, Greece), pp. 938–941, Dec. 2010.

[7] **A. Rák**, G. Feldhoffer, G. B. Soós, and G. Cserey, „Standard C++ Compiling to GPU with Lambda Functions," in *Proceedings of 2010 International Symposium on Nonlinear Theory and its Applications (NOLTA 2010)*, (Krakow, Poland), 2010.

[8] **A. Rák**, G. Feldhoffer, G. B. Soós, and G. Cserey, „Standard c++ compiling to GPU," in *3rd HUNGARIAN-SINGAPOREAN WORKSHOP on SYSTEMS BIOLOGY and COMMUNICATION SYSTEMS*, (Budapest, Hungary), 2010.

[9] **A. Rák**, G. Feldhoffer, G. B. Soós, and G. Cserey, „CPU-GPU hybrid compiling for general purpose: Case studies," in *Proceedings of 12th International Workshop on Cellular Neural Networks and their Applications, CNNA 2010*, (Berkeley, USA), Feb. 2010.

[10] G. J. Tornai, G. Cserey, and **A. Rák**, „Spatial-temporal level set algorithms on CNN-UM," in *Proceedings of 2008 International Symposium on Nonlinear Theory and its Applications, NOLTA 2008*, (Budapest, Hungary), pp. 696–699, 2008.

[11] G. B. Soós, **A. Rák**, J. Veres, and G. Cserey, „GPU powered CNN simulator (SIMCNN) with graphical flow based programmability," in *Proceedings of 11th International Workshop on Cellular Neural Networks and their Applications, CNNA 2008*, (Santiago de Compostela, Spain), pp. 163–168, 2008. Cited: **2**.

6.3. A szerző egyéb publikációi

[12] **A. Rák**, and Feldhoffer, G., and Soós, G.B. and Höltzl, T., and Oroszi, B. and Cserey, György, „Eljárás és rendszer integrál kiszámításának párhuzamos architektúra szálára való leképezésére." Hungarian and PCT patent, 2012. 2013.

[13] G. Cserey and **A. Rák**, „High accuracy time-to-digital converter on FPGA." Hungarian patent, 2009.

[14] **A. Rák** and G. Cserey, „Számítógépes architektúra és feldolgozási eljárás." Hungarian patent, 2012.

[15] **A. Rák**, G. Cserey, and B. Jákli, „Eszköz és eljárás mért jel időbeliségének meghatározására." PCT patent, 2013.