Rīgas Tehniskā Universitāte

Datorzinātnes un informācijas tehnoloģijas fakultāte

Datorvadības, automātikas un datortehnikas institūts

Kursa darbs mācību priekšmetā

Signālu kodēšana un apstrāde

„Daudzkodolu procesori signālu apstrādē”

V.Zagurskis

2012g.

Saturs

Ievads

Daudzkodolu mikroprocesori tiek izmantoti daudzos industrijas segmentos, tādos, kā signālu kodēšana un apstrāde, kosmosa segmentā un citos. Paaugstināts pieprasījums pēc tiem radies tāpēc, ka risināmie uzdevumi vairāk un vairāk pieprasa ātrdarbības resursus, kā arī rodas nepieciešamība pēc programmēšanas uzdevumiem. Procesori izmanto paralelizācijas pieeju un tajos tiek iekļauts vairāk un vairāk paralēlās struktūras, saglabājot nepieciešamos jaudas rādītājus vajadzīgajā līmenī. Eksistē vairāki multiprocesoru risinājumi, sākot ar tradicionāliem daudzkodolu risinājumiem un beidzot ar risinājumiem, kuri sastāv no vairākuma („jūras") aritmetisko-loģisku ierīču kopuma.

Paralēlie procesori attīstās jau ļoti ilgu laika posmu, sākot ar procesoru „Solomon", kurš tika izveidots 1960to gadu vidū. Šos procesorus bija ļoti grūti programmēt un tas nozīmēja, ka tos varēja izmantot tikai inženieri un zinātnieki, kuriem bija nepieciešamās zināšanas un prasmes programmēšanā un šo procesoru arhitektūrā. Šī iemēsla dēļ, daudzi vēlākizveidotie procesori netika atzīti par veiksmīgiem un neieguva popularitāti. Par izņēmumu varētu kalpot „Cray" procesors, kurš izmantoja vektora programmēšanas paņēmienu, kas nebija tik sarežģīts apgūšanā un turpmākajā programmēšanā. Papildus, šiem ierīcēm tika piešķirts skalārais, lielās jaudas procesors, kas palielināja ātrdarbību. Pēdējā laikā starp procesoru ražotājiem pastāv tendence izveidot vienčipa daudzkodolu procesorus.

Šādu pieeju izskaidro divas lietas: Mūra likums (katrus 18 mēn. tranzistoru skaits procesors pieaug 2 reizes) un patērētāju vajadzības pēc tā saucamās „jēlās" ātrdarbības (būtībā – procesora resurss, kurš ir pieejams izmantošanai). Procesoriem jāapmierina šīs divas prasības, vienlaicīgi neparkāpjot pāri nozīmētai jaudas patēriņa barjerai. Vienkodolu procesoriem, lai apmierinātu skaitļošanas vajadzības un vajadzības pēc ātrdarbības, bija jāpaaugstina frekvence, kas izraisīja lielāku jaudas patēriņu un tas nebija pieņemami. Šo risinājumu autori mēģināja „izspiest" lielāku ātrdarbību no instrukciju plūsmas. Rezultātā izveidotie procesori patērēja daudz vairāk enerģijas, to arhitektūra bija pārlieku sarežģīta un nevadāma. Turklāt, šāda pieeja nevarētu apmierināt tā saucamo ITRS plānu, kurš paredzēja, ka, turpinoties pieprasījuma pēc atrdarbības palielinājumam, 2022 gadā izlaisto procesoru ātrdarbībai būtu jābūt 300 reizes lielākai, nekā līdzšinējiem procesoriem. Tieši tādēļ, mikroprocesoru izstrādātāji pievērsās paralēlai skaitļošanai un daudzkodolu procesoriem, lai varētu apmierināt līdzšinējas vajadzības, kā arī sekot ātrdarbības paaugstināšanas plānam. Neskatoties uz to, ka ātrdarbības izmaiņas ir jūtamas, ITRS paredz, ka 2022 gadā procesori saturēs 100 reizes vairāk kodolu, nekā mūsdienīgākie procesori šobrīd. Par svarīgāko daudzkodolu procesoru arhitektūras ieguvumu var

uzskatīt faktu, ka, lai palielinātu šādu procesoru ātrdarbību, ir nepieciešams tikai palielināt paralēlo bloku skaitu, nepalielinot frekvenci, kas rezultējās mazākā jaudas patēriņā un, līdz ar to – mazākos siltuma zudumos. Taču šādai pieejai ir arī būtisks trūkums – attīstoties paralēlam aprīkojumam, paralēlās programmēšanas paņēmieni attīstās daudz lēnāk. Līdz ar to, pagaidām pastāv neatbilstība starp aparatūras un programmatūras platformas līmeņiem, kas liedz pilnā mērā izmantot paralēlās skaitļošanas jaudas.

Vispārējās nozīmes daudzkodolu procesoriem ir liela nozīme digitālo signālu apstrādē. Agrāk, DSA izmantoja šādu arhitektūru: viens vadības kodols un vairākas specifiskas integrētas komponentes, katra priekš sava uzdevuma. Tā kā mūsdienās aparatūras veicamo uzdevumu saraksts ir ļoti garš, tad nav iespējams izveidot aparatūru, vadoties pēc iepriekšizmantotā principa, jo tad specifisko komponenšu skaits būs ļoti liels (piemērām, mobīlā telefona procesors, kuram jāatbalsta dažādas tehnoloģijas, video un audio kodeki, ofisa aplikācijas utt.). Vadoties pēc šādiem atzinumiem, tika nolemts šiem uzdevumiem izmantot vispārējās nozīmes daudzkodolu procesorus, kurus būtu viegli programmēt jebkuru uzdevumu izpildei un kuri piedāvātu augstākminēto „jēlo jaudu" un neizvirzītu nekādas specifiskas prasības izstrādātājam.

Daudzkodolu procesori var tikt klasificēti pēc vairākiem parametriem. Šajā rakstā tie tiks salidzināti pēc pašiem svarīgākiem parametriem:

1. programmatūras klase
2. jauda / ātrdarbība
3. apstrādes elementi
4. atmiņas sistēma
5. akseleratori / integrētās komponentes

Programmatūras klase

Ja tehnika ir domāta kādam specifiskam programmatūras risinājumam, tad aparatūras platformu var pielāgot šim risinājumam. Rezultātā mēs varam izveidot vienkāršāku pēc uzbūves konstrukciju, ar mazākiem jaudas patēriņiem. Ja atcerēties DPS agrākos piegājienus, tad tie vadījās tieši pēc šādām vadlīnijām. Rezultātā sanāca ierīces, optimizētas un pildošas ļoti efektīvi tikai vienu (vai vairākus līdzīgus) uzdevumus. Ja uz šīm ierīcēm palaida kādu programmatūru, kurai šī ierīce nebija domāta, tad ātrdarbības rezultāts bija ļoti vājš.

Eksistē divas apstrādes klases, kurām var piederēt programmatūra:

1. datu apstrādes klase
2. kontroles klase

Datu apstrādes klase

Datu apstrādes programmatūra risina plašu uzdevumu klāstu: grafikas rasterizācija, attēlu un skaņas apstrāde, kā arī bezvadu tīklu nesējfrekvences apstrāde. Šī uzdevumu klase ietver arī vairākus skaņas apstrādes algoritmus. Šo uzdevumu risinājums parasti ir secīga operāciju izpilde attiecībā uz datu plūsmu, kur dati tiek ļoti maz ottreizēji izmantoti, vai vispār netiek izmantoti vairākas reizes. Vajadzīgās operācijas var tikt izpildītas paralēli. Šo uzdevumu izpildei ir nepieciešama ļoti liela ātrdarbības rezerve, jo datu apjomi ir ļoti lieli. Tadējādi, izstrādājot ierīces šādu uzdevumu risināšanai, jāizmanto daudzprocesoru sistēma, starp kuriem varētu sadalīt skaitļošanas (apstrādes) uzdevumus.

Kontroles klase

Pie kontroles klases uzdevumiem var attiecināt: datu kompresiju / dekompresiju, tīklu vadību un tranzakcijas vaicājumu apstrādi. Šīs klases programmu kods nereti satur ļoti daudz loģiskās izteiksmes, nosacījumu pārbaudi un, līdz ar to, šo programmu paralēla izpildīšana ir sarežģīta. Parasti programmām jāseko līdzi lielam parametru apjomam un tās parasti izmanto vienu un to pašu datu kopu vairākkārtīgi. Ierīcēm, kuras risina šāda veida uzdevumu parasti ir daudz mazāks skaitīšanas elementu (procesoru) skaits.

Ir gandrīz neiespējami attiecināt kādu no programmām uz vienu vai otru uzdevumu klasi. Taču, ja sadalīt programmas izpildes secību uz vairākiem posmiem, tad šos posmus var diezgan noteikti attiecināt pie datu apstrādes vai kontroles klasei. Piemēram H.264/AVC video kodeka izpilde pieder pie datu apstrādes klases bloķēšanas filtra piemērošanas laikā, bet video plūsmas kompresijas / dekompresijas laikā šī programma pieder pie kontroles klases. Ir svarīgi sadalīt programmas šādā veidā, lai labāk izprastu programmas arhitektūru un izstrādāt aparatūras daļu atbilstoši katrai no uzdevumu klasēm. Citādi, ja aparatūra izpilda efektīvi tikai datu apstrādes klases komandas, tad ātrdarbības kritums, izpildot kontroles klases uzdevumus, būs ļoti liels.

Jauda / ātrdarbība

Vairākas ierīces, kā arī programmas izvirza striktas ātrdarbības un jaudas patēriņa prasības. Piemēram mobīlajam telefonam ar video atbalstu ir ierobežots jaudas patēriņa parametrs, taču ir nepieciešams panākt atbilstošu ātrdarbību. Ja ātrdarbība ir tradicionāls kritērijs ierīcēm, tad jaudas patēriņš ir salīdzinoši jauns kritērijs, kurš paradījās, jo tirgu aizvien vairāk parādās mobīlās ierīces, kurām ir ierobežota baterijas ietilpība, kā arī ierobežoti izmēri. Turklāt, šī tendence tagad aptver arī ne-mobīlo ierīču klasi. Tā saucamie „datu centri”, kuri tiek izmantoti priekš masīvo apjomu kalkulācijām (cloud computing) un pārsvarā sastāv no daudzkodolu procesorus saturošām komponentēm, patērē milzīgu jaudas apjomu. Šie centri var būt tik lieli, ka

tiem pat ir izdomāts īpašs nosaukums – noliktavas izmēra skaitļošanas mašīnas. Tādēļ, daudzkodolu procesoru, kuri tiek izmantoti šajās sistēmās, jaudas patēriņš spēlē kritisku lomu.

Apstrādes elementi

Šajā sadaļā tiek apskatīti tādas procesoru īpašības, ka arhitektūra un mikroarhitektūra. Būtībā, ar arhitektūru saprotu instrukcijas kopas arhitektūru (Instruction Set Architecture – ISA), kura definē saskarni starp aparatūras un programmatūras platformām. Mikroarhitektūra ir arhitektūras konkrēts implementējums.

Arhitektūra

Pēc savas būtības, jaunu daudzkodolu procesoru ISA arhitektūra ir nekas cits, ka atbilstošās tiem vienprocesoru versijas uzlabotā arhitektūra, kurai izstrādātas modifikācijas paralēlās komandu izpildes rezultāta panākšanai, kā arī iekļautas instrukcijas kodolu sinhronizācijai savā starpā. Šīs arhitektūras priekšrocība ir tajā, ka tai ir izveidoti vairāki risinājumi, kā arī, ka to ir iespējams salīdzinoši viegli programmēt. Šī arhitektūra var būt arī pašrocīgi izveidota vai modificēta.

ISA var tikt klasificēta kā dators ar samazinātu instrukciju kopu (Reduced Instruction Set Computer – RISC) vai dators ar pilnu instrukciju kopu (Complete Instruction Set Computer – CISC). Kaut gan šie divi jēdzieni stipri atšķirās pagātnē, tagad atšķirības ir gandrīz nemanāmas: CISC ierīces strādā gandrīz tāpat, kā RISC analogi pēc dekodēšanas beigām. Kodēšanas / dekodēšanas daļā joprojām pastāv lielas atšķirības, jo CISC sistēmām ir lielāks instrukcju kopums, kā arī lielāks semantiku daudzums. RISC analogiem ir lielāks koda daudzums, jo tiem ir jāemulē sarežģītas CISC instrukcijas ar savām, vienkāršajām instrukcijām. RISC priekšrocība ir tajā, ka, pateicoties vienkāršākām komandām, tās ir viegli kompilēt, kā arī mikroarhitektūras dizains ir vienkāršāks.

Attīstoties ISA arhitektūrai, ierīču un komponenšu izstrādātāji nemitīgi to papildina ar jauniem paplašinājumiem, lai uzlabotu kādu procesu izpildes ātrumu. Piemēram, Intel korporācija pievienoja saviem procesoriem MMX, MMX2 un SSE1-SSE5, lai pāatrinātu ar multimēdiju saistītas operācijas. ARM kompānija pievienoja līdzīgas instrukcijas savam izstrādājumam un nosauca tās par NEON. Šīs instrukcijas ir interesantas ar to, ka rada ļoti labu jaudas patēriņa / ātrdarbības rezultātus veicot vektora transponēšanas operācijas ar vienas instrukcijas palīdzību. Programmatūras kodola izstrādātāji Tensilica padarīja šo izgudrojumu par svarīgāko savā procesorā ar nosaukumu Xtensa CPU, kurā iestrādāja specifiskas instrukcijas speciālajām vajadzībām.

Mikroarhitektūra

Apstrādes elementa (procesora) mikroarhitektūra nosaka, kādas būs šī elementa jaudas patēriņa / ātrdarbības raksturojumi. Katra elementa mikroarhitektūra parasti tiek saistīta ar konkrētu uzdevumu klasi, kuru risina daudzkodolu procesors. Lai arī komerciāli produkti (piemērām, Intel) parasti apvieno vairākus vienādus apstrādes blokus un veido homogēnu arhitektūru, nereti ir nepieciešams apvienot vairākus atšķirīgus skaitļošanas blokus un izveidot heterogēnu arhitektūru. Šīs idejas mērķis ir palielināt ātrdarbību, nepalielinot frekvenci (un, līdz ar to, nepalielinot patērējamo jaudu). Tipiska šādas pieejas infrastruktūra satur galveno vadošo mezglu, kurš vada tam pakārtotos vienkāršākus kodolus. Datu apstrādes klases uzdevumos šāda pieeja var dot lielu ātrdarbību pie salīdzinoši maziem jaudas patēriņiem. Šādas pieejas trūkums ir tajā, ka heterogēnās sistēmas ir daudz sarežģītāk programmēt.

Vienkāršākais apstrādes elements ir secīgais elements. Tāds elements dekodē un izpilda programmas instrukcijas un dinamiski seko līdzi datu nosūtīšanai uz citiem blokiem, kā arī atbild par kontroles uzdevumiem. Šādam elementam ir divi ātrdarbības rādītāji, kurus ir iespējams uzlabot. Ir iespējams izveidot paralēlos konvejerus, lai rodas iespēja izpildīt vairāk par vienu instrukciju paralēli, tadējādi radot superskalāro arhitektūru, kura palielina ātrdarbību. Taču palielinot konvejeru skaitu un izpildamo komandu skaitu, ir jāizstrādā mehānismi, kas pasargās no nekorektas piekļūšanas pie datiem, kā arī vadīs konvejerus. Šo mehānismu sarežģītuma pakāpe ir kvadrātiski atkarīga no izmantoto konvejeru skaita. Cits paņēmiens, kā var palielināt ātrdarbību ir palielināt konvejera soļu skaitu, tadējādi samazinot katra soļa sarežģītumu un spējot palielināt frekvenci. Taču šāda pieeja stipri zaudē ātrdarbībā, ja programmā tiek sastapts sazarojums. Šādiem secīgiem elementiem ir mazs kristāla izmērs, zems enerģijas patēriņš un tos var viegli kombinēt ar lielu daudzumu līdzīgu elementu, ja risināmai problēmai (tai, kuru risinās procesors) ir augsts plūsmas līmeņa paralēlisms (Thread Level Paralelism – TLP) un pāris secīgās sekcijas, kuras ir jūtīgas pret ātrdarbību. Piemēram Nvidia G200 čipam ir 240 paralēlie kodoli un tikai pāris secīgas sekcijas, jo grafikas apstrāde daudzajā mērā balstās uz paralēlo datu apstrādi.

Superskalārā kodola pielāgošana vienas komandu plūsmas izpildei ir tā saucamā „nesakārtotā" pieeja, kura cenšās nesakārtoti ievietot instrukcijas izpildīšanas stekā, lai aizpildītu procesora konvejerus. Tāda dinamiska instrukciju plānošana prasa sarežģītas procesoru veidojošās shēmas, kas, savukārt, negatīvi atspoguļojas patērētajā enerģijā. Šādi projektēti kodoli ir galvenokārt domāti aplikācijām ar ļoti plašu scenāriju sarakstu un lielām ātrdarbības prasībām. Bet, papildus lielam enerģijas patēriņam, šiem procesoriem ir nepieciešama diezgan liela elementu izvietošanas matrica (kas nav pieļaujama, piemērām, mobīlās sistēmās). Šādu procesoru galvenā pielietošanas sfēra ir dažādu domēnu problēmu risināšana – to izmantošana

strikti specifisko uzdevumu risināšanai nedod tiem nekādu parākumu. Tādēļ, ka šie procesori aizņem diezgan lielu platību (dēļ lielā elementu skaita), tos nevar apvienot procesoru matricās. Taču, tie ir ieteicami izmantošanai gadījumos, kad izpildamā programmatūra pieder kontroles klasei, tai ir kritiskās instrukciju izpildes secības un paliels TLN (sk. augstāk). Piemērām ARM Cortex A9 procesors ir domāts nebukiem, utilizē vienu instrukciju plūsmu ar TLN, tādēļ tam der „nesākārtoti" kodoli.

Lai pāatrinātu instrukciju izpildi superskalārajā arhitektūrā, vienlaicīgi nepalielinot elementu skaitu, kuri vajadzīgi, lai izpildītu šo instrukciju rindu, var tikt izmantoti divi paņēmieni: viena instrukcija – daudz datu (Single Instruction – Multiple Data, SIMD) vai izmantojot ļoti garu instrukciju vārdu (Very Long Instruction Word, VLIW). SIMD arhitektūra sadala apjomīgus reģistrus plūsmās tā, lai plūsmas varētu tikt apstrādātas ar vienu instrukciju. Par piemēru var kalpot divu vektoru summēšana. Katrs vektora koordinašu pāris tiek summēts paralēli un neatkarīgi no otra. Šāda pieeja ir efektīva gadījumos, kad tiek izpildīti uzdevumi no datu apstrādes klases un dati ir paralēli. IBM Cell izmanto vairākus SIMD kodolus lai apstrādātu lielus datu apjomus. Taču SIMD arhitektūra ir ļoti neefektīva daudzpusīgo uzdevumu izpildei – tā tiek izstrādāta (instrukcijas) kādas konkrētas problēmas risinājumam.

Lai izvairītos no situācijas, kad viena instrukcija apstrādā vairākas datu plūsmas, var izmantot VLIW arhitektūru. VLIW izmanto vairāku konvejeru arhitektūru, taču neparedz pārsūtīšanas (forwarding), plānošanas (scheduling) un kļūdu atklāšanas loģiku kodolos. Kompilātors sagrupē instrukcijas paķetēs un izpilda tos paralēli, garantējot kļūdu nepieļaušanu. Tadejādi visa sarežģītā loģika tiek pārnesta uz kompilātoru, padarot kodola uzbūvi vienkāršāku. VLIW arhitektūra var tikt izmantota plašā procesoru klāstā, ja tie var apstrādāt vairākus datu punktus ar vairākām instrukcijām, tadējādi radot parākumu par SIMD arhitektūru. Taču, kā jau tika minēts, VLIW ātrdarbība daudzajā mērā saistās ar kompilātoru un, ja kompilātors nespēj atrast vajadzīgā paralēlisma līmeni, VLIW arhitektūra spēcīgi zaudē ātrdarbībā. Jāpiezīmē, kā gan SIMD, gan VLIW ir augstās ražības, energoefektīvās sistēmās, taču kalpo ļoti specifisku aplikāciju darbināšanai, proti, tādu aplikāciju, kur lielākā koda daļa ir neatkarīgās operācijas, kuras vai nu programmētājs, vai nu kompilātors (kā kura gadījumā) var novirzīt uz konvejeriem un izpildīt paralēli. Zemākdotajā tabulā ir summarizēti šo abu arhitektūru īpatnības:

ISA	Priekšrocības	Trūkumi
Novēcojušās	Kompilātoru un programmatūras atbalsts	Iespēju var nepietikt speciālizētajām aplikācijām, kurām nepieciešama augsta ātrdarbība.
Pašrocīgi būvētās	Var tikt optimizētas priekš konkrētā	Šiem lietojumiem var neeksistēt

	lielojuma	kompilātoru vai programmatūras atbalsta
RISC	Vienkāršāks mikroarhitektūras un kompilātoru dizains	Koda apjomi var būt ļoti lieli; neefektīva pieeja dažu lietojumu gadījumā
CISC	Lielāka optimizācija, mazāki koda apjomi	Sarežģīta mikroarhitektūra, sarežģīts kompilātoru dizains
Speciālās	Dod iespēju ļoti spēcīgi optimizēt aplikāciju kodu	Ļoti sarežģīts dizains, pastāv iespēja manuāli programmēt, jo risinājumam nav kompilātora
Mikroarhitektūra	**Priekšrocības**	**Trūkumi**
Secīgās	Zema sarežģītības pakāpe, mazs izmērs, iespēja izvietot vairākus uz vienas matricas	Vienplūsmas ātrdarbība ir vidēja vai zema
Nesākārtotās	Ļoti augsta vienplūsmas ātrdarbība dinamiskās plānošanas dēļ	Sarežģīts dizains, liels jaudas patēriņš, liels izmērs
SIMD	Ļoti efektīva paralēlām / vektoru kalkulācijām	Nepiemērota kontroles klases aplikācijām, var zaudēt ātrdarbībā, ja instrukcijas nevar paralelizēt
VLIW	Var izpildīt vairākas vienkāršās instrukcijas	Ir vajadzīgs sarežģīts kompilātors

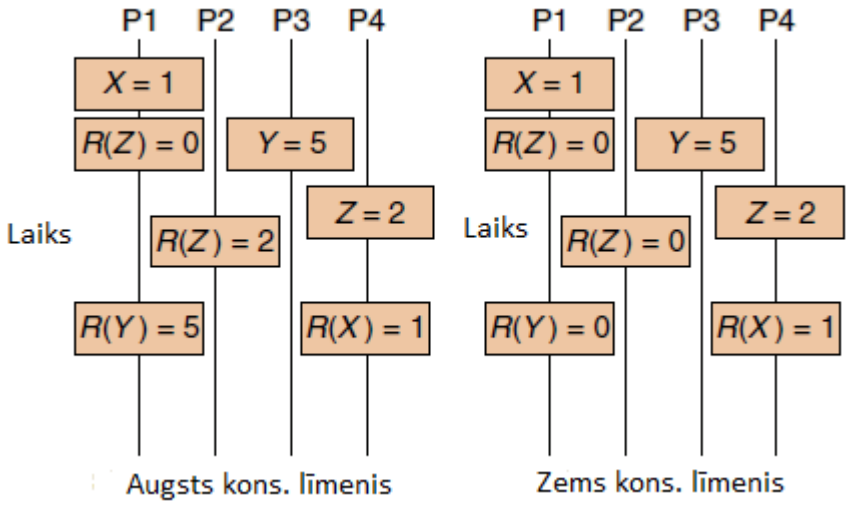
Atmiņas sistēma

Vienprocesoru sistēmās atmiņas sistēma sastāv no pāris kešatmiņas līmeņiem, kuri ir atbildīgi par datu un istrukciju glabāšanu un nodošanu procesoram. Daudzprocesoru sistēmās kešatmiņa kalpo kā viens no līmeņiem un kā lielās atmiņas sistēmas sastāvdaļa. Citas atmiņas sistēmas komponentes ir: konsekventuma modelis, kešatmiņu (tādas ir vairākas – atsevišķa katram kodolam) koherentuma atbalsts un iekščipu savienojumi. Visas šīs komponentes nosaka, kādā veidā kodoli mijiedarbojas savā starpā, paralēlās izpildīšanas iespējas un to, cik daudz kodolus sistēma var reāli atbalstīt.

Konsekventuma modelis

Konsekventuma modelis apraksta, kādā veidā atmiņas operācijas var tikt pārkārtotas koda izpildīšanas laikā. Šis modelis nosaka, kas jāizdara programmētājam (cik daudz pūles jāpieliek), lai uzrakstītu vajadzīgo kodu. Vājākiem modeļiem, programmētājam pašam jāsastāda izpildīšanas secību procesora kodolā, kā arī jādefinē sinhronizācijas protokolus. Taču stiprākiem modeļiem eksistē strikti ierobežojumi, kādā veidā tiek organizēta plānošana, eksistē noteikumi, pie kuriem atmiņas sistēmai var piekļūt citi elementi. Piemērām, secīgais konsekventums nosaka,

ka visiem procesoriem jāredz, ka lasīšanas un rakstīšanas operācijas notiek vienādā kārtībā globālajā līmenī. Tas var ietekmēt ātrdarbību, taču padara programmēšanu par vienkāršāku, jo jebkurā laika brīdī var noteikt, kā izpildīsies paralēlais kods. Zems konsekventuma līmenis nosaka, ka lasīšanas un rakstīšanas operācijas notiek dažādās secībās katrā procesorā. Dēļ šīs situācijas ir nepieciešams izstrādāt papildus loģiskās konstrukcijas, tādas, ka barjeras un žogi, lai nodrošinātu nepieciešamo sinhronizācijas pakāpi. Nākamais attēls raksturo zemu un augsto konsekventuma līmeni:



Kā var redzēt no attēla, tad sistēmai ar augstu konsekventuma līmeni, darbības, kuras ir veicis viens procesors, ir redzamas citiem procesoriem un atmiņai visu laiku ir aktuāls stāvoklis. Piemērām, ja procesors P3 piešķir mainīgaijam Y vērtību 5, tad procesoram P1 vēršoties pie šī atmiņas apgabala, tiek uzrādīta vērtība 5. Tāpat, ja procesors P1 piešķir apgabalam Z vērtību 0, bet procesors P4 vērtību 2, tad procesors P2 vēršoties pie šī apgabala dabū vērtību 2 (aktuālo vērtību). Turpretīm, modelim ar zemu konsekventuma līmeni, darbības, kuras veikuši procesori paliek neredzami vienam priekš otra. Tas nozīmē, ka procesori „neklausa" viens otru un bieži strādā ar neaktuāliem datiem. Piemēram, ja procesors P3 ieraksta apgabalā Y vērtību 5, tad procesors P1 šo vērtību nenolasa, bet nolasa iepriekšējo vērtību. Augsts konsekventuma līmenis tiek nodrošināts ar globālu arbitrāžas mehānismu, kurš sakārto vēršanos pie atmiņas. Sistēmām ar zemu konsekventuma līmeni ir vienkāršāka atmiņas struktūras uzbūve, taču paaugstina rakstāmā koda sarežģītību, jo sistēmas programmētājam pašam jārūpējas par to, lai izmaiņas datos sasniegtu visus procesorus un lai tie strādā ar korektām izmaiņām. Savukārt, sistēmai ar augstu konsekventuma līmeni atmiņas struktūra ir vienkāršāka, taču samazinās ātrdarbība, kura tiek zaudēta dēļ komandu secīgās un nepārprotamās izpildīšanās.

Kešatmiņas konfigurācija

Kešatmiņa kalpo kā ātra lokālā atmiņa ar lielu caurlaidspēju. Kešatmiņa ir kā buferis starp ātru procesoru (vai vairākiem procesoriem) un relatīvi lēnu (lēnāku, nekā procesors) pamatatmiņu, vai operatīvo atmiņu. Kešatmiņa var būt automātiski vai lokāli kontrolēta. Galvenais automātiskās atmiņas pluss ir tajā, ka priekš instrukciju rindas tā ir caurspīdīga un instrukcijām ir pieeja pie viena veida atmiņas. Par galveno trūkumu var minēt to, ka priekš automatiskās kešatmiņas vadības ir nepieciešami tā saucamie tegi – dati, kuri apzīmē katru kešatmiņas rindu (ierakstu). Tie tiek glabāti uz matricas (aizņem vietu), kā arī rada nedeterminētu ātrdarbību. Lokālai atmiņai ir determinēta ātrdarbība, tā neaizņem vietu, jo neglabā tegus, taču, izpildāmajā programmatūrā ir jāparedz konstrukcijas, kuras pārvaldīs atmiņu. Tas ir sarežģīts uzdevums un to nav ieteicams izmantot, ja vien izstrādājamā sistēma nav reāllaika ar striktām ātrdarbības prasībām.

Nepieciešamo kešatmiņas apjomu nosaka izmantojamā aplikācija. Palielināts kešatmiņas apjoms pozitīvi ietekmē ātrdarbību, taču palielināts kešatmiņas apjoms aizņem lielu matricas daļu. Pie tam, lielā atmiņa ir neefektīva aplikācijās, kurās dati netiek izmantoti vairākkārtīgi, tādās kā video kodēšana / dekodēšana. Šajos gadījumos ir labi, ja kešatmiņa var pārslēgties starp dažādiem operēšanas veidiem, tādiem kā ridas pieeja un normālā pieeja. Microsoft Xenon izstrādājumam ir speciāla funkcionalitāte, lai pasargātu (attīrītu) kešatmiņu no tādiem datiem, kuri netiks vairākkārtīgi izmantoti. Kešatmiņas apjomu parasti ierobežo matricas izmēri un plānotā patērējamā jauda. Tāda pieeja tiek izmantota sistēmās, kuras risina kontroles klases uzdevumus, kur dati tiek vairākkārtīgi izmantoti, piemērām – Intel Core i7.

Kešatmiņas līmeņu skaita paaugstināšanos izraisa ar apstrādi saistīto elementu skaita paaugstināšanos. Lieta, kuru jāņem vērā ir cik ciklus galvenā atmiņa ir tālu (atpaliek) no apstrādes elementiem (procesoriem). Jo lielāks ir ciklu skaits, jo vairāk ir vajadzīgi kešatmiņas līmeņi. Pirmais kešatmiņas līmenis ir mazs, ļoti ātrs un parasti ir atsevišķs katram kodolam. Citi kešatmiņas līmeņi ir lielāki, lēnāki un parasti ir koplietošanā visiem kodoliem. Kešatmiņa atļauj kodolam strādāt ar ātru atmiņu, kaut gan operatīvā atmiņa ir vairākus tūkstošus ciklu lēnāka par procesoru. Kā piemērs var kalpot servera-tipa daudzkodolu procesori, tādi kā AMD Phenom, kuriem ir 3 kešatmiņas līmeņi. Kešatmiņas nozīme ir nozīmīga ne tikai darbstaciju un serveru procesoriem, bet arī iebūvējamām sistēmām, tādām kā Texas Instruments OMAP 4430, kuriem operatīvās atmiņas ātrdarbības atšķirība ir vēl lielāka. Kešatmiņa palīdz samazināt šo starpību.

Iekščipu starpsavienojumi

Starpsavienojumi ir atbildīgi par čipā esamo komponenšu savienojumiem, kā arī kešatmiņas koherentuma nodrošināšanu (ja tā eksistē). Eksistē vairāku tipu starpsavienojumi: kopne, sijas tipa savienojums, riņķis un tīkls uz čipa (network-on-chip, NoC). Kopne ir visvienkāršākais savienojuma veids, taču nespēj nodrošināt pietiekošu caulaidspēju, risinājumos, kuros tiek izmantoti vairāki apstrādes elementi. Turpretīm, tīklam uz čipa ir lieliskas paplašinājuma iespējas, taču tā konstrukcija ir ievērojami sarežģītāka.

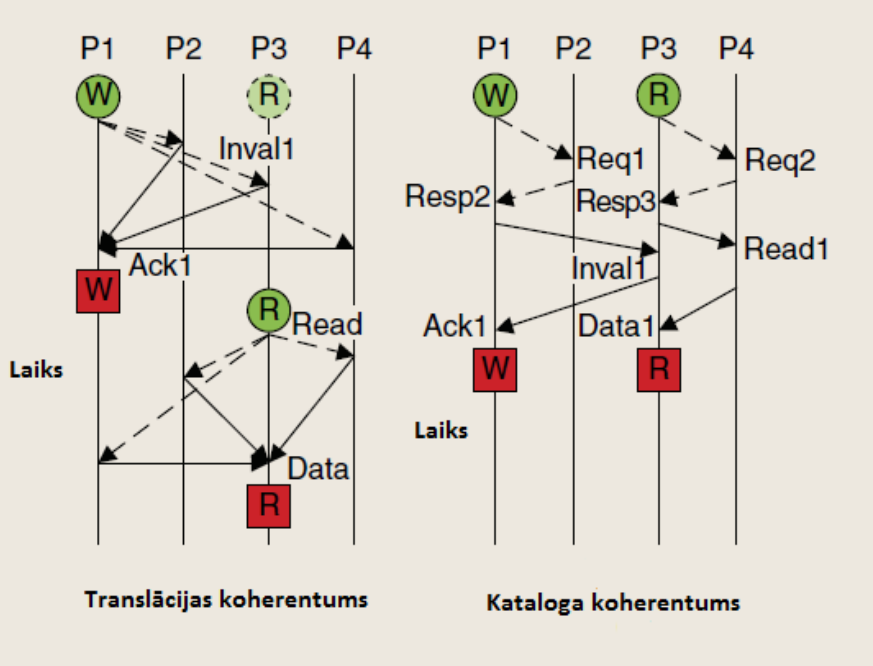
Starpsavienojumi nodrošina kešatmiņas koherentumu. Koherentums ir atmiņas attēls, kurš ir globāli redzams visiem procesoriem un tas tieši ietekmē programmēšanas modeļus, kuri ir atkarīgi no koplietojamās atmiņas. Šāda pieeja tiek plaši izmantota vispārējās lietošanas procesoriem, tādiem, kā ARM Cortex A9. Eksistē divi koherentuma veidi: translācijas koherentums un kataloga koherentums.

Translācijas koherentuma pieeja ir salīdzinoši vienkārša – tā tiek panākta atļaujot tikai vienam procesoram izpildīt tikai vienu darbību, kura ir redzama visiem procesoriem. Ja kāds no procesoriem vēlas izpildīt darbību, tas nosūta pieprasījumu uz visiem citiem procesoriem. Pieprasījums tiek apstrādāts un tiek nosūtīta atbilde vai apstiprinājums (acknowledgement, ACK). Kad procesors saņem visus apstiprinājumus, tas veic darbību. Lielākais šīs pieejas trūkums ir tajā, ka, kamēr viena operācija nav noslēgusies, cita operācija (no cita procesora) nevar sākties, jo translācijas trafiks aizņem visu kopni. Taču ir izskaitļots, ka salīdzinoši mazam procesoru skaitam šī pieeja der. Tā tiek izmantota astoņu kodolu Intel Core i7 procesoros.

Kataloga koherentums tiek izmantots sistēmās ar lielu procesoru skaitu un pieeja atšķirās no augstākaprakstītās translācijas pieejas. Šī pieeja atļauj vairākas koherentās darbības izpildīt vielaicīgi. Visi procesori izpilda vaicājumus globālajam katalogam, kurš satur informāciju par to, kura kešatmiņa ir atbildīga par katra atmiņas apgabala glabāšanu. Katrai adresei tiek piešķirts mezgls, kurā glabājas daļa no globālā kataloga. Ja kādam procesoram ir nepieciešama pieeja, tas izpildīs vaicājumu uz šo mezglu, ar mērķi noskaidrot, kuri procesori pašlaik strādā ar noteikto kešarmiņas apgabalu. Pēc šī vaicājuma izpildīšanas procesori nodod izmantošanas tiesības šim procesoram un tas var darboties ar kešatmiņas apgabalu. Kataloga koherentuma modelis var tikt un tiek pielietots lielās sistēmās ar mazu koherentuma pakāpi, piemērām Tilera TILE64.

Neskatoties, ka kešatmiņas koherentums ir svarīgs, vairākās sistēmās tas netiek iekļauts, lai vienkāršotu sistēmu uzbūvi un samazinātu izmaksas, kā arī lai vienkāršotu izstrādājumu testēšanu. Rezultātā, vairākām daudzprocesoru sistēmām kešatmiņas koherentuma modelis nepastāv, kā piemēru var minēt TI TMS320DM6467 un IBM Cell (!). Atkārtosim, ka sistēmām

bez kešatmiņas koherentuma, šī funkcionalitāte jāimplementē programmatūrā. Nākamais attēls vizuāli parāda atšķirības starp translācijas un kataloga koherentuma modeļiem:



Dažu komerciālo multiprocesoru risinājumu kešatmiņas un citu parametru lielumi un iezīmes ir parādīti tabulā (skat. 1. pielikumu).

Akselerātori un integrētā perifērija

Akselerātori un integrētā perifērija ir parasti ASIC konstrukcijas vai specializētie procesori, kuru funkcionalitāti nevar efektīvi emulēt ar programmatūru. Kā piemērus šo ierīču veidam var minēt grafikas rasterizētājus, kodeku akseleratorus, atmiņas kontrolerus. Šīs ierīces vairākuma gadījumos palielina patērējamo enerģijas daudzumu, bet var būtiski palielināt ātrdarbību. Sistēma, kurā ir iestrādātas šādas ierīces ir TI OMAP4430.

Komerciālas daudzkodolu sistēmas

Komerciālie produkti tiek izlaisti ikvienam tirgus segmentam, sākot ar iebūvētām sistēmām un beidzot ar serveru sistēmām. Daudzkodolu arhitektūras pieeja tiek izmantota tāpēc, lai, nepalielinot patērējamo enerģijas daudzumu.

Pirmais šādu sistēmu veids ir vispārējās lietošanas daudzkodolu procesori. Kodolu mikroarhitektūra ir tradicionāla un bāzējas uz vispārpieņemtiem vienprocesoru uzbūves principiem. Procesoros ir vairākas vienādas kodolu kopijas, kā arī iespaidīgs kešatmiņas apjoms. Šie procesori tiek izmantoti, lai izpildītu darbastaciju un servera segmenta programmatūru, kur enerģijas patēriņš nav noteicošais.

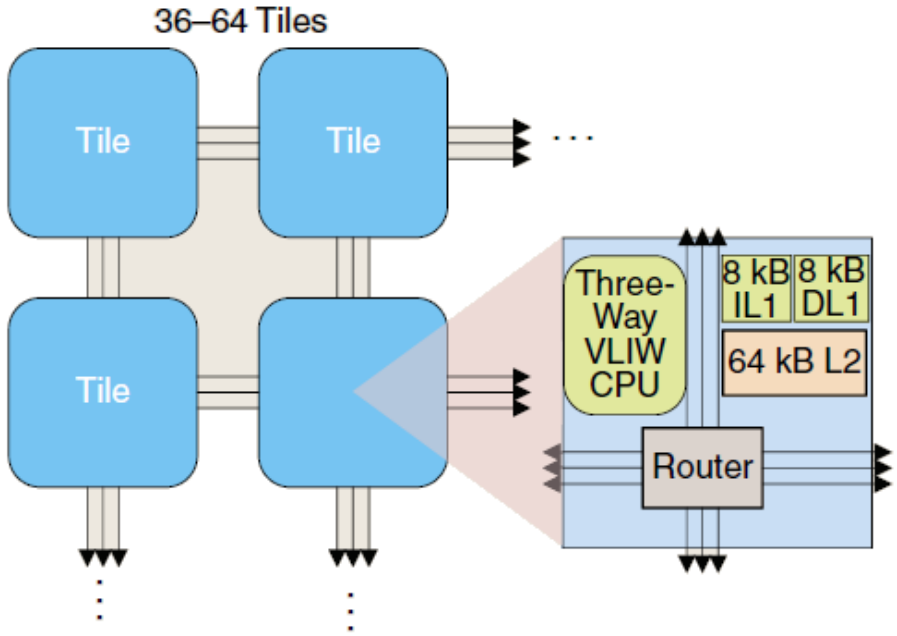
Otrs veids arī ir pieskaitāms pie vispārējās lietošanas sistēmām, taču uz orientāciju uz mobilitāti. Konstrukcija ir līdzīga iepriekšpieminētajam veidam, taču, tie ir orientēti uz kontroles klases uzdevumiem, patērē ļoti maz enerģijas, jo tiek darbināti no baterijām.

Nākošais veids ir specializētās augstās ražības arhitektūras (skat. 2. pielikumu), kuras ir paredzētas lielās ātrdarbības skaitļošanai. Procesoros kodolu skaits ir ļoti liels – parasti pāri 100 gabaliem. Kā piemēru šādām sistēmām var minēt AMD R700 vai NVIDIA G200. IBM Cell sistēma ir heterogēna ar lielu skaitu augsti specializētām skaitļošanas ierīcēm. Šādām ierīcēm parasti ir liels enerģijas patēriņš: no 100W līdz 180W.

Vēl viens augstās ražības arhitektūru veids ir specializētās ierīces kādam noteiktam aplikāciju domēnam (skat. 2. pielikumu). Šo ierīču vairākums ir domāts datu apstrādes aplikāciju domēniem, tādiem kā bezvadu joslas, audio / vizuālie kodeki, kur parasts paralēlisms var tikt izmantots. Līdz ar to, šīs ierīces atbalsta augstu komputācijas mēru. Vairākām ierīcēm ir speciāli izveidots un pielāgots starpsavienojumu tīkls. Šīs ierīces sasniedz augsto komputācijas rādītāju, nepatērējot lielus enerģijas apjomus.

TILERA TILE64 digitālā signālu apstrāde

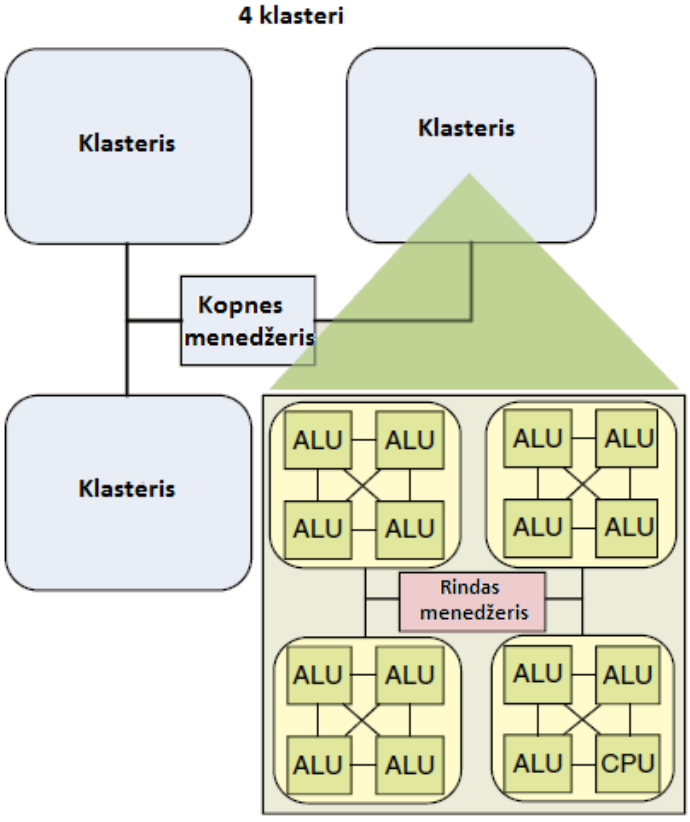
Tilera TILE64 ir uz digitālo signālu apstrādi orientēts procesors, kurā ir izmantota daudzkodolu arhitektūra. Arhitektūra sastāv no 64 parastiem VLIW kodoliem, kuri ir savienoti ar tīklu uz čipa (network on chip, NoC), kurš ir pilnīgi koherents. Tādēļ, ka katrs kodols ir mazs un patērē maz enerģijas, šai sistēmai ir nepieciešams ļoti labs paralēlisma mehānisms programmatūrā. Nākamajā attēlā ir paradīta šīs ierīces shēma:



Jāpiezīmē, ka pilnīgi koherenta savienošana nav vispārpieņemts paņēmiens digitālo signālu apstrādes sistēmu realizācijai, taču šis paņēmiens ļauj procesoram izpildīt vairākus vispārīgās lietošanas lietojumus, kuri izmanto koplietojamo kešatmiņu. Starpsavienojums ir liels tīkls uz čipa, izmanto kataloga koherentuma modeli, ar mērķi sasniegt labu paplašinājuma pakāpi. Tādēļ, ka šis procesors ir papildus orientēts uz vispārīgiem lietojumiem, tam ir koherenti starpsavienojumi un platāki apstrādes elementi, tas patērē 18W enerģijas. Šis rādītājs ir lielāks nekā citiem šīs klases procesoriem.

ELEMENT CXI ECA-64 digitālo signālu apstrāde

Šis procesors pieder pie ierīcēm ar ļoti zemu enerģijas patēriņu. Šī pieeja izmanto savādāku ierīces dizainu, izmantojot dažus kontroles kodolus lai vadītu aritmētiski-loģisko ierīču kopumu (arithmetical logical unit, ALU). Ierīce ir orientēta uz datu klases uzdevumiem, patērējot mazu enerģijas daudzumu. Ierīces programmēšana ir līdziga FPGA (field-programmable gate array) programmēšanai. Attēlā ir parādīta ierīces shēma:



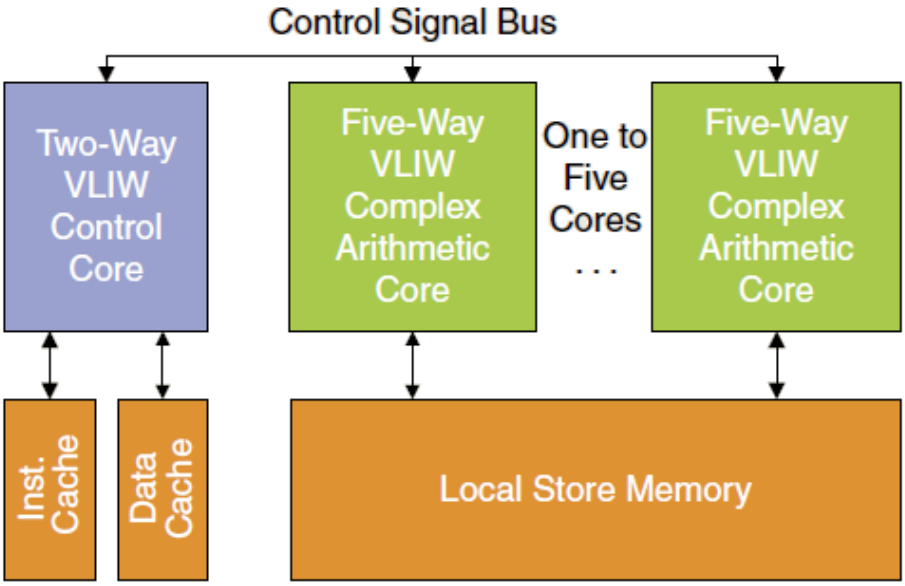
Zemu enerģijas patēriņu izdevās sasniegt, izmantojot heterogēno dizaina pieeju. Procesors ir izveidots no četriem klasteriem, kur katrs klasteris satur 16 apstrādes elementus. Katrs klasters satur vienu RISC kodolu un 15 ALU, kur katrs ALU ir specializēts savādākam pielietojumam. ALU ir datu vadāmie, kas nozīmē, ka tie strādā tikai tad, kad ir pieejami dati, pārējos gadījumos paliekot bezdarbībā, tadējādi ekonomējot enerģiju.

Katram kodolam ir 32KB kešatmiņas, kura tiek vadīta ar programmatūru. Starpsavienojumi ir hierarhiski. Četri kodoli tiek savienoti ar sijām un pēc tam šo četru kodolu grupas tiek savienotas savā starpā ar punkts-punkts rindām, tadējādi formējot 16 kodolu klasterus. Klasteri savienojas savā starpā ar kopni, kura atļauj tiem komunicēt.

SILICON HIVE HIVEFLEX CSP2X00 digitālo signālu apstrāde

Šīs ierīces patērētā jauda sastāda tikai 1/4W. Ierīcei ir kontroles čips, kurš uzbūvēts pēc vispārējās lietošanas divu virzienu VLIW dizaina ar mazo kešatmiņu, kura tiek izmantota lai izpildītu pamatprogrammu. Datu apstrādes uzdevumi tiek ielādēti uz tā saucamiem „sarežģītiem kodoliem". Sarežģītie kodoli ir piecu virzienu VLIW kodoli ar pielāgotiem ALU, lai pāatrinātu metamātiskās operācijas; kodoli ir pieslēgti lielam RAM atmiņas masīvam. Šie kodoli neatbalsta programmas zarošanos, tādēļ tiem ir jāpadod taisns, nekomplicēts kods no kontroles kodola. Tas, ka sarežģītie kodoli neatbalsta zarošanos, pozitīvi ietekmē izmērus un patērēto jaudas daudzumu (tie tiek samazināti).

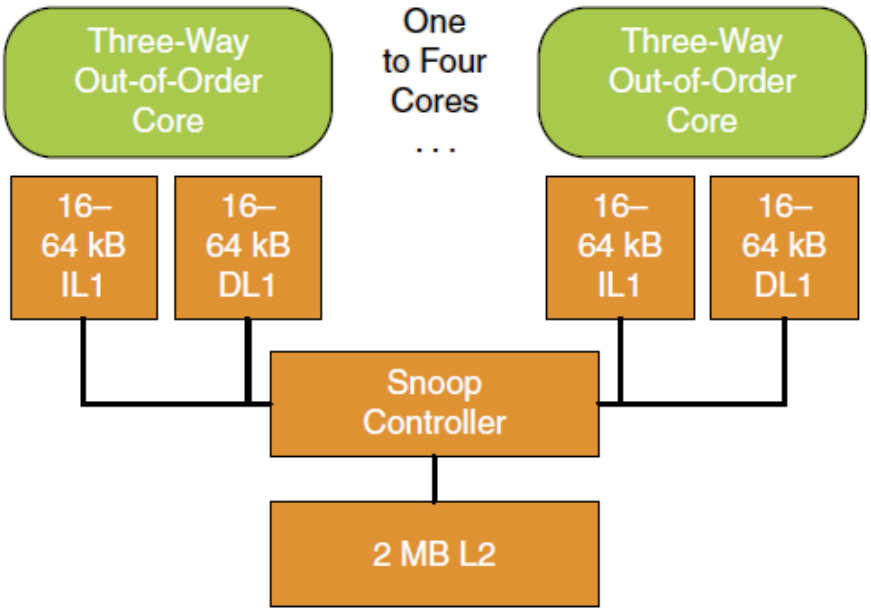
Šai ierīcei ir ļoti vienkārša atmiņas hierarhija. Koherentumu un saderību kontrolē uz kontroles kodola izpildošā programmatūra. Kopne ir galvenokārt domāta komandu nodošanai no kontroles kodola uz sarežģītiem kodoliem, kuri, savukārt, komunicē caur speciālu noliktavu. Attēlā ir paradīta ierīces blokshēma:



ARM CORTEX A9 vispārējās lietošanas mobilā ierīce

ARM Cortex A9 ir vispārējās lietošanas iebūvēts kodols, kuru ir iespējams pielāgot specifiskiem uzdevumiem pirms tā ražošanas. Patērētais enerģijas apjoms nepārsniedz 1W. Ierīce ir domāta izmantošanai smart-telefonos un netbukos, lai veiktu vispārējos uzdevumus. Ierīcē labi tiek galā ar kontroles klases uzdevumiem. Individuālie A9 kodoli ir trīs virzienu,

nesākārtotie kodoli, kuri piedāvā labu vispārējās vajadzības ātrdarbību. Ierīce ir domāta operētājsistēmas palaišanai un citu darbastaciju uzdevumu risināšanai. Starpsavienojums ir pilnīgi koherenta kopne. Tā kā kodolu skaits ir relatīvi mazs, tad tiek izmantots translācijas koherentuma modelis. Kešatmiņas izmērs šīs klases ierīcei ir gana liels. Šāds kešatmiņas izmērs tiek izvēlēts tādēļ, lai nodrošinātu lielas taktēšanas frekvences vienplūsmas kodolos. Datu klases uzdevumus šis čips izpilda neefektīvi, jo ir pozicionēts citam segmentam. Attēlā ir paradīta čipa blokshēma:

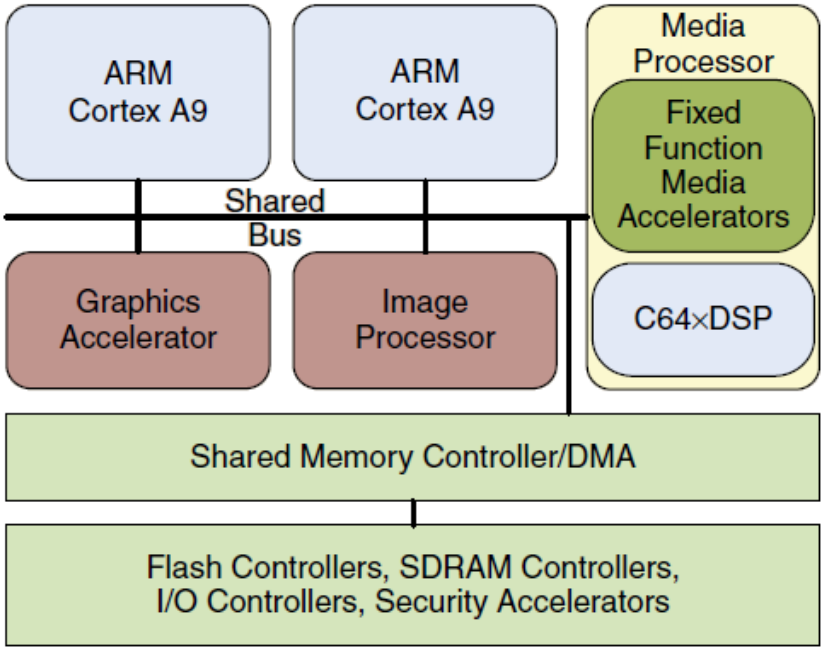


TI OMAP 4430 vispārējās lietošanas sistēma uz čipa

Šī ierīce ir vispārējās lietošanas sistēma uz čipa (System on Chip, SoC), kura ir orientēta uz nākotnes smart-telefoniem un mobīliem interneta ierīcēm (mobile Internet devices, MIDs). Ierīcei ir mazais patērējamais enerģijas daudzums (ap 1W), plašas skaitļošanas iespējas un liels atbalstāmās perifērijas daudzums. Ierīce izmanto divus ARM A9 procesorus vispārējās lietošanas uzdevumiem un C64x digitālo signālu apstrādes mezglu datu intensīviem uzdevumiem. Media uzdevumu apstrādei šai ierīcei ir paredzētas ASIC ierīces, lai paātrinātu šos uzdevumus, nepalielinot (ievērojami) enerģijas patēriņu: GPU (graphics processing unit), attēlu procesors, audio/video kodeku procesors. No perifērijas var minēt uz čipa esošo šifrēšanas mezglu. Būtībā šī ierīce ir nekas cits, ka ASIC mezglu kopums, kurš tiek kontrolēts ar kontroles procesoru. Ja kādu uzdevumu nevar atrisināt ar esošiem ASIC mezgliem, tiek iesaistīti trīs ARM A9 kodoli: tiek patērēts vairāk enerģijas, taču uzdevumu klāsts tiek paplašināts.

Ierīces starpsavienojumi sastāv no pilnīgi koherentās kopnes starp ARM A9 kodoliem. Tas nodrošina vispārējo uzdevumu programmēšanu ar dalīto (koplietojamo) atmiņu. Kopne, kura

apvieno C64x mezglu ar akselerātoriem (ASIC) ir nekoherenta, tādēļ ar šo uzdevumu nodarbojas ARM A9 kodoli. Galvenais atmiņas kontrolieris ir koplietojams. Attēlā ir dota šīs ierīces blokshēma:



NVIDIA G200 grafiskā / augstās ražības platforma

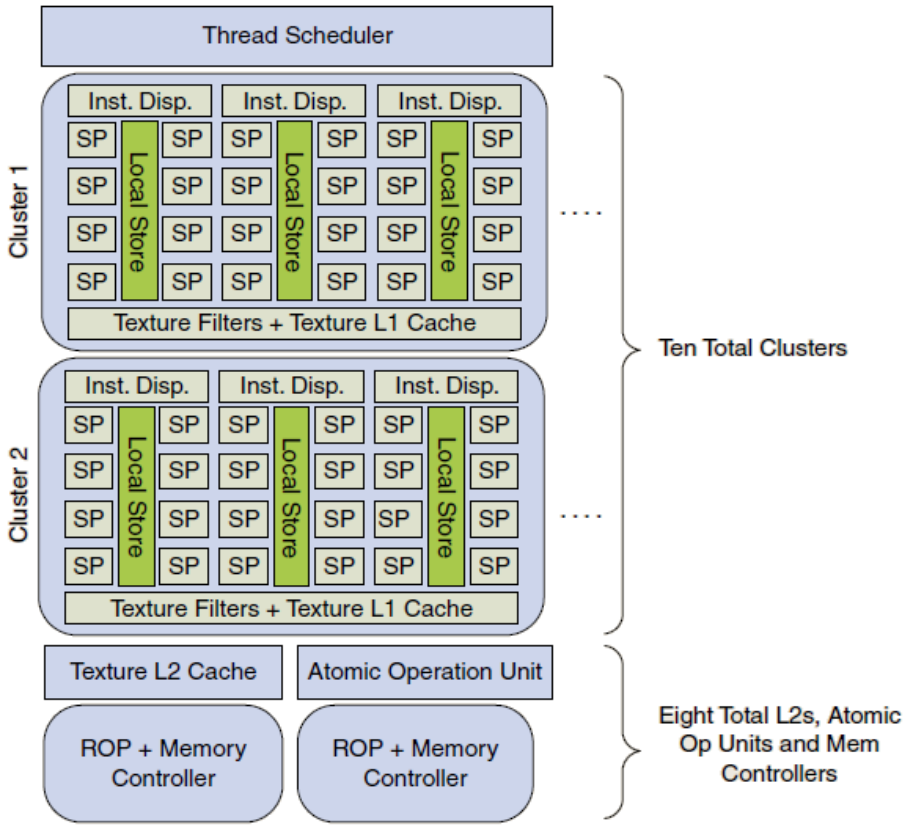
Šis grafisks risinājums ir orientēts uz datu klases uzdevumiem, pārsvārā uz rastra grafiku. Taču, izņemot grafiskus uzdevumus, šī platforma spēj apstrādāt arī citus datu atkarīgus uzdevumus.

Arhitektūra ietver 240 vienvirziena sakārtotus kodolus. Kodoli tiek grupēti pa 24 gabaliem klasteros. Katrai grupai, kura sastāv no 8 kodoliem, ir 16KB liela kešatmiņa. Katrs klasteris (24 kodoli) tiek kontrolēts pēc SIMD principa: katrs kodols izpilda vienādu instrukciju no savādākas plūsmas (thread). Vēršanās pie atmiņas notiek pēc ne-SIMD pieejas: ja viens kodols pieprasa adresi X, tad cits kodols pieprasa adresi Y, nevis X+1. Šī pieeja samazina ātrdarbību, jo parasti atmiņas kontrolieris nevar apvienot šos pieprasījumus. Šī pieeja arīdzen ģeneralizē platformu, padarot to vairāk lietojamu vispārējā rakstura aplikācijām, līdzinot to vairāku instrukciju / vairāku datu mašīnai (multiple instruction / multiple data, MIMD), bet pilnīga līdzība ar MIMD nepastāv, jo notiek ātrdarbības zudumi momentos, kad instrukciju plūsmas un atmiņas pieprasījumi sadalās.

Atmiņas sistēma ir pielāgota datu klases uzdevumiem – tā ir nekoherenta un izmanto mazas armiņas vietnes kešatmiņas vietā (kešatmiņa eksistē, taču tā tiek izmantota nevis vispārējiem lietojumiem, bet gan rastra grafikas kalkulācijām un apstrādēm). G200 platformai ir

ļoti maza atmiņa, kura atrodas uz kristāla – platforma izmanto darbastacijas resursus – tiek izmantota operatīvā atmiņa.

Ierīce ir labi pielāgota datu klases uzdevumiem, tādiem, kā medicinisko attēlu apstrāde vai finansiālo datu apstrāde. Turpretīm, šī ierīce nav labi piemērota kontroles klases uzdevumiem, jo vēršanās pie atmiņas un zarojumu apstrāde izraisa ātrdarbības zudumus. Šī platforma ir ļoti labi piemērota galvenajam uzdevumam – grafikas apstrādei. Attēlā ir dota platformas blokshēma:



Intel Core i7 vispārējās lietošanas procesors

Šis izstrādājums no Intel korporācijas ir vispārējās lietošanas procesors, kurš ir orientēts uz vienādi efektīfu jebkuru uzdevumu izpildīšanu. Līdz ar to, ierīce patērē lielu enerģijas daudzumu – 140W.

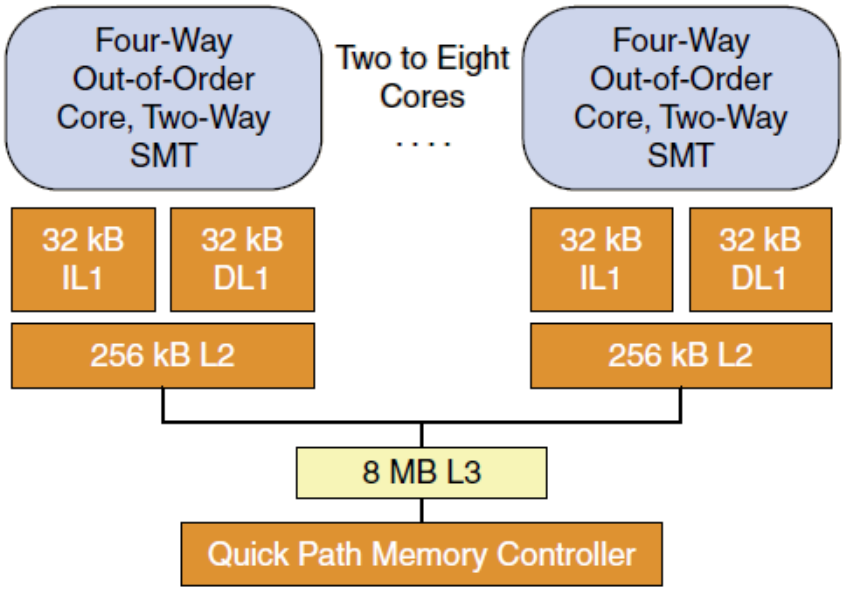
Procesors ir realizēts kā astoņi četru-operāciju, divvirzienu simetriskie daudzuzdevumu (simultaneous multithreadint, SMT) kodoli. Katrs kodols satur 128 bitu SMD mezglu, lai gūtu ātrdarbības pieaugumu no paralēlām operācijām. Procesors atbalsta CISC x86 ISA instrukcijas.

Procesors izmanto koherentu atmiņas sistēmu un lielas kešatmiņas katrā kodolā.

Koherentums ir bāzēts uz translācijas modeli, jo kodolu skaits ir pietiekoši mazs.

Procesors ir orientēts uz plašu lietojumu klases uzdevumu izpildi un apstrādi, ņemot vērā to, ka patērētais enerģijas apjoms nav ierobežojums.

Attēlā ir parādīta Intel Core i7 blokshēma:



	ISA	**Mikroarhitektūra**	**Kodoli**	**Kešatmiņa**	**Koherentums**	**Starpsavien.**	**Saderība**	**Max. pat. enerģija**	**Frekvence**	**OPS/Clock**
AMD Phenom	x86	3-virzienu, nesakārtotā, superskalārā, 128 bitu SIMD	4	64 KB IL1 AND DL1/ CORE, 256 KB L2/CORE, 2-6 MB L3	Katalogs	P2P	Procesors	140W	2.5 GHZ–3.0 GHZ	12–48 OPS/CLOCK
INTEL CORE I7	x86	4-virzienu nesakārtotā, divvirzienu SMT, 128 bitu SIMD	2-8	32 KB IL1 AND DL1/ CORE, 256 KB L2/CORE, 8 MB L3	Translācija	P2P	Procesors	130W	2.66 GHZ–3.33 GHZ	8–128OPS/CLOCK
SUN NIAGARA	SPARC	Divvirzienu sakārtotā, 8-virzienu SMT	8	16 KB IL1 AND 8 KB DL1/ CORE, 4 MB L2	Katalogs	Crossbar	Noliktavas kārtošana	60-123W	900 MHZ–1.4 GHZ	16OPS/CLOCK
Intel Atom	X86	Divvirzienu sakārtotā, divvirzienu SMT, 128 bitu SIMD	1-2	32 KB IL1 AND DL1/ CORE, 512 KB L2/CORE	Translācija	Kopne	Procesors	2-8W	800 MHZ–1.6 GHZ	2–16OPS/CLOCK
ARM Cortex A9	ARM	3-virzienu nesakārtotā	1-4	(16,32,64) KB IL1 AND DL1/CORE, UP TO 2 MB L2	Translācija	Kopne	Vāji saderīga	1W	N/A	3–12 OPS/CLOCK
XMOS XS1-G4	XCORE	Vienvirziena sakārtota, 8-virzienu SMT	4	64 KB LCL STORE/CORE	Nav	Crossbar	Nav	0.2W	400MHZ	4 OPS/CLOCK

2. pielikums

	ISA	Mikroarhitektūra	Kodoli	Kešatmiņa	Koherentums	Starpsavien.	Saderība	Max. pat. enerģija	Frekvence	OPS/Clock
AMD Radeon R700	N/A	5-virzienu VLIW	160 CORES, 16 CORES PER SIMD BLOCK, TEN BLOCKS	16 KB LCL STORE/SIMD BLOCK	Nav	N/A	Nav	150W	750 MHZ	800-1600 OPS/CLOCK
NVIDIA G200	N/A	Vienvirziena sakārtotā	240, EIGHT CORES PER SIMD UNIT, 30 SIMD UNITS	16 KB LCL STORE/EIGHT CORES	Nav	N/A	Nav	183W	1.2 GHZ	240-720 OPS/CLOCK
INTEL LARABEE	x86	Divvirzienu sakārtotā, 4-virzienu SMT, 512-Bitu SIMD	Līdz 48	32 KB IL1 AND 32 KB DL1/ CORE, 4 MB L2	Translācija	Divvirzienu gredzens	Procesors	N/A	N/A	96-1,536 OPS/CLOCK
IBM Cell	Power	Divvirzienu sakārtotā, divvirzienu SMT PPU, divvirzienu IN-ORDER 128-B SIMD SPU	1 PPU, 8 SPU	PPU: 32 KB IL1 AND 32 KB DL1, 512 KB L2; SPU: 256 KB LCL STORE	Nav	Divvirzienu gredzens	Nav	100W	3.2 GHZ	72 OPS/CLOCK
Microsoft Xenon	Power	Divvirzienu sakārtotā, divvirzienu SMT, 128 bitu SIMD	3	32 KB IL1 AND 32 KB DL1/ CORE, 1 MB L2	Translācija	Crossbar	Vāja	60W	3.2 GHZ	6-24 OPS/CLOCK
AMBRIC AM2045	N/A	Vienvirziena sakārtots SR, 3-virzienu sakārtots SRD	168SR, 168SRD	21 KB LCL STORE/EIGHT CORES	Nav	NoC	Nav	6-16W	350MHZ	672 OPS/ CLOCK
ELEMENT CXI ECA–64	N/A	Vienvirziena sakārtota, DATAFLOW CONNECTIONS Līdz 15 rekonfigurējamiem	4 klasteri ar vienkodolu ALU	32 KB OF LCL STORE/ CLUSTER	Nav	Hierarhisks NoC	Nav	1W	200MHZ	64 OPS/CLOCK (16-B)

		ALU								
TI TMS320-DM6467	ARM, C64X	1 ARM9 vienvirziena sakārtots, 1C64X 8-virzienu VLIW	2	ARM9: 16 KB IL1, 8 KB DL1; C64X: 32 KB IL1 AND DL1, 128 KB L2	Nav	Kopne	Vāja	3-5W	ARM: 297–364 MHZ, C64X: 594–729 MHZ	1-9 OPS/CLOCK
TI OMAP 4430	ARM, C64X	2 ARM 3-virzienu nesakārtoti, 1C64X 8-virzienu VLIW	3	N/A	Translācija starp ARM kodoliem	Kopne	Vāja	1W	1 GHZ	6-140 OPS/CLOCK
TILERA TILE64	N/A	3-virzienu VLIW	36-64	8 KB IL1 AND DL1/CORE, 64 KB L2/CORE	Katalogs	NoC	N/A	15-22W	500–866 MHZ	108–192 OPS/CLOCK
HIVEFLEX CSP2X00	N/A	Divvirzienu VLIW kontroles kodols, 5-virzienuVLIW Kompleksais kodols	2-5	2X CONFIGURABLE L1 FOR BASE CORE, LCL STORE FOR COMPLEX CORE	Nav	Kopne	Nav	0.25W	200MHZ	2-22 OPS/CLOCK

Literatūra

[1] International Technology Roadmap for Semiconductors, “International technology roadmap for semiconductors—System drivers,” 2007 [Online]. Available: http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_System-Drivers.pdf

[2] Intel Corp., “Intel core i7-940 processor,” Intel Product Information, 2009 [Online]. Available: http://ark.intel.com/cpu.aspx?groupId=37148

[3] Element CXI Inc., “ECA-64 elemental computing array,” Element CXI Product Brief, 2008 [Online]. Available: http://www.elementcxi.com/downloads/ECA64ProductBrief.doc

[4] ITU-T, “H.264.1: Conformance specification for h.264 advanced video coding,” Tech. Rep., June 2008.

[5] “Intel 64 and IA-32 Architectures Software Developer’s Manual,” *Intel Developer Manuals*, vol. 3A, Nov. 2008.

[6] ARM Ltd., “The ARM Cortex-A9 Processors,” ARM Ltd. White Paper, Sept. 2007 [Online]. Available: http://www.arm.com/pdfs/ARMCortexA-9Processors.pdf

[7] Tensilica Inc., “Configurable processors: What, why, how?” Tensilica Xtensa LX2 White Papers, 2009 [Online]. Available: http://www.tensilica.com/products/literature-docs/white-papers/configurable-processors.htm

[8] A. L. Shimpi and D. Wilson, “NVIDIA’s 1.4 billion transistor GPU: GT200 arrives as the GeForce GTX 280 & 260,” Anandtech Web site, 2008 [Online]. Available: http://www.anandtech.com/video/showdoc.aspx?i=3334

[9] M. Gschwind, H. P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki, “Synergistic processing in cell’s multicore architecture,” *IEEE Micro*, vol. 26, no. 2, pp. 10–24, 2006.

[10] J. Andrews and N. Baker, “Xbox 360 system architecture,” *IEEE Micro*, vol. 26, no. 2, pp. 25–37, 2006.

[11] Advanced Micro Devices Inc. “Key architectural features—AMD Phenom II processors,” AMD Product Information, 2008 [Online]. Available: http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_15331_15917%5E15919,00.html

[12] Texas Instruments Inc., “OMAP 4: Mobile applications platform,” 2009 [Online].Available: http://focus.ti.com/lit/ml/swpt034/swpt034.pdf

[13] Tilera Corp., “Tilepro64 processor,” Tilera Product Brief, 2008 [Online]. Available: http://www.tilera.com/pdf/ProductBrief_TILEPro64_Web_v2.pdf

[14] Texas Instruments, Inc., “TMS320DM6467: Digital media system-on-chip,” Texas Instruments Datasheet, 2008 [Online]. Available: http://focus.ti.com/lit/ds/symlink/tms320dm6467.pdf

[15] Advanced Micro Devices Inc., “Software optimization guide for AMD family 10h processors,” AMD White Papers and Technical Documents, Nov. 2008 [Online]. Available: http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/40546.pdf

[16] Sun Microsystems Inc., “UltraSPARC T2 processor,” Sun Microsystems Data Sheets, 2007 [Online]. Available: http://www.sun.com/processors/UltraSPARCT2/datasheet.pdf

[17] T. Johnson and U. Nawathe, “An 8-core, 64-thread, 64-bit power efficient sparc soc (niagara2),” in *Proc. 2007 Int. Symp. Physical Design ISPD ’07*. New York, NY: ACM, 2007, pp. 2–2.

[18] Intel Corp., “Intel atom processor for nettop platforms,” Intel Product Brief, 2008 [Online]. Available: http://download.intel.com/products/atom/319995.pdf

[19] D. May, “XMOS XS1 architecture,” XMOS Ltd., July 2008 [Online]. Available: http://www.xmos.com/files/xs1-87.pdf

[20] Advanced Micro Devices Inc., “ATI Radeon HD 4850 & ATI Radeon HD 4870—GPU specifications,” AMD Product Information, 2008 [Online]. Available: http://ati.amd.com/products/radeonhd4800/specs3.html

[21] NVIDIA Corp., “NVIDIA CUDA: Compute unified device architecture,” NVidia CUDA Documentation, June 2008 [Online]. Available:

http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf
[22] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan, "Larrabee: A many-core x86 architecture for visual computing," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–15, Aug. 2008.
[23] IBM Corp., "POWER ISA Version 2.05," Power.org documentation, Oct. 2007 [Online]. Available: http://www.power.org/resources/reading/Power-ISA_V2.05.pdf
[24] T. R. Halfhill, "Ambric's new parallel processor: Globally asynchronous architecture eases parallel programming," *Microprocessor Rep.*, Oct. 2006[Online]. Available: http://://www.mdronline.com/mpr/h/2006/1010/204101.html
[25] Nethra Imaging Inc., "Massively parallel proccessing arrays technology overview," Ambric Technology Overview, 2008 [Online]. Available: http://www.ambric.com/technologies_mppa.php
[26] S. Kelem, B. Box, S. Wasson, R. Plunkett, J. Hassoun, and C. Phillips, "Anelemental computing architecture for SD radio," in *Proc. Software Defined Radio Technical Conf. Product Exposition*, 2007.
[27] M. Baron, "Tilera's cores communicate better: Mesh networks and distributed memory reduce contention among cores," *Microprocessor Rep.*, Nov. 2007 [Online]. Available: http://www.mdronline.com/mpr/h/2007/1105/214501.html
[28] A. Agarwal, B. Liewei, J. Brown, B. Edwards, M. Mattina, C.-C. Miao, C. Ramey, and D. Wentzlaff, "Tile processor: Embedded multicore for networking and multimedia," in *Proc. Hotchips 19: A Symp. High Performance Chips*, 2007.
[29] "HiveFlex CSP2000 series: Programmable OFDM communication signal processor," *Silcon Hive Databrief*, 2007 [Online]. Available: http://www.siliconhive.com/Flex/Site/Page.aspx?PageID=8881