

Dsp verwerkt grote stromen gegevens

In deel 1 in PT ES 2000/9 is de programmeerbare embedded cpu besproken, die met name geschikt is voor de controle-achtige delen van een applicatie. In deze editie komt de programmeerbare dsp aan de orde. Deze processor blinkt uit in het verwerken van grote stromen van gegevens, en is goed toepasbaar in media-applicaties.

GER SCHOEBER, Ordina Media, Son

Programmeerbare dsp-cores zijn ontworpen voor het uitvoeren van taken met harde realtime beperkingen. Een verschil met programmeerbare cpu-cores met de load-store-architecturen is dat we nu rekening moeten houden met het verwerken van grote arrays. De load-store-architecturen met de centrale plaats van de register-file zijn eerder toegespitst op het verwerken van scalaire variabelen. In dsp-architecturen zullen dus ook grotere RAM-geheugens en het datatransport van en naar die geheugens een belangrijke plaats innemen.

We herkennen een drietal basisconcepten achter de architectuur van programmeerbare dsp's. Het eerste basisconcept is de *vermenigvuldiger-accumulator* (MAC). Die vormt het aritmetisch hart,

geoptimaliseerd voor de voornaamste dsp-operatie: de som van producten. Verder kent de dsp een zogenoemde *gewijzigde Harvard-architectuur* met juist meerdere databussen en meerdere geheugens. Naast de al genoemde MAC is er ook een ALU aanwezig. De ALU is typisch bedoeld voor minder regelmatige algoritmes waarin beslissingen en condities voorkomen (if – then – else).

Vermenigvuldiger-accumulator

De MAC is goed toepasbaar in filtering, correlatie, spectrale analyse, enzovoort. Het streven is om elke klokcyclus een product en een sommatie uit te voeren. Dit is een groot verschil met de eerder besproken embedded cpu. Een MIPS-processor heeft daar bijvoor-

beeld negentien operaties voor nodig! Figuur 1 toont de basisstructuur. Soms wordt er ook in een additionele output voorzien om het product rechtstreeks als output beschikbaar te maken. Ook kan er een additionele input zijn als een rechtstreekse ingang naar de *adder*.

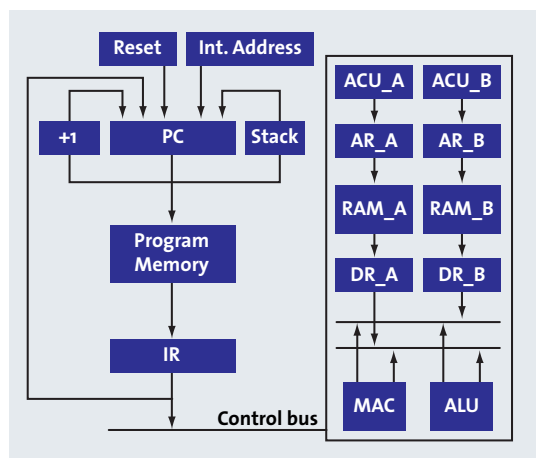
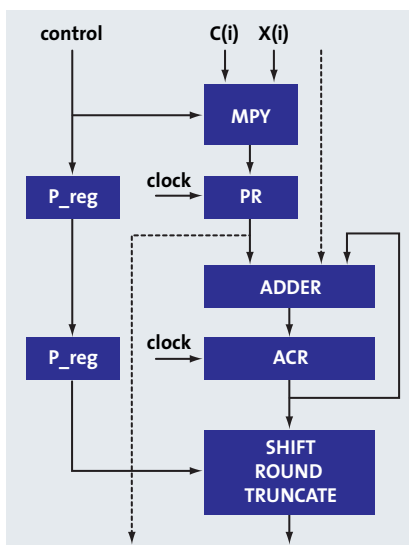
Een andere variant is over twee accumulatorregisters te beschikken. Dit kan nuttig zijn voor een butterfly-berekening, waarin eerst $(a + b)$ berekend moet worden en vervolgens $(a - b)$. Het is snel duidelijk dat een klassieke Von Neumann-architectuur niet voldoet. In deze architectuur wordt geen onderscheid gemaakt tussen instructies en data. Beide verblijven in hetzelfde geheugen en worden via dezelfde bus getransporteerd. Daarbij gebeuren alle operaties in deze architectuur sequentieel.

Gewijzigde architectuur

Een alternatief is de *gewijzigde Harvard-architectuur*. Hierin zijn het controlepad en het datapad van elkaar gescheiden. Vervolgens kent deze architectuur twee databussen en twee grote RAM-geheugens. Elk geheugen heeft een eigen *address computation unit* (ACU). In figuur 2 is deze architectuur weergegeven, waarin AR staat voor adresregister, DR voor dataregister en IR voor instructieregister. De beide adresgeneratoren moeten voldoende krachtig zijn om in elke klokcyclus een volgend adres te kunnen berekenen. Deze architectuur is bij uitstek geschikt om bijvoorbeeld een delay-lijn van een FIR-filter te implementeren.

Hierbij is het wel van belang goed af te spreken hoe de data in het geheugen wordt opgeslagen. Dit bepaalt in belangrijke mate de implementatie van een delay-lijn. Dit kan er toe leiden dat volledige blockmove-operaties nodig zijn. Een alternatief is het gebruik van pointers. Hierbij blijft de data op zijn plaats en geeft een pointer het begin aan van een delay-lijn.

1. Basisstructuur met een productregister (PR) en een accumulatorregister (ACR).



2. Elk geheugen heeft een eigen *address computation unit* (ACU). In deze figuur is deze architectuur weergegeven, waarin AR staat voor adresregister.

ALU

Om echter ook het hoofd te kunnen bieden aan bijvoorbeeld sorterende filters, interpolatie, absolute-waarde-berekeningen en logaritmische conversies, is naast een vermenigvuldiger-accumulator (MAC) ook een ALU nodig. In het begin werd getracht de ALU met de MAC samen te voegen, maar al snel bleek dat beide toch een verschillend karakter hebben, en al snel werd de ALU een aparte volwaardige unit die volledig parallel kan werken aan de vermenigvuldiger-accumulator.

Vaak zien we in de controller-architecturen terug dat een aantal operaties in parallel kunnen plaatsvinden. Een belangrijk kenmerk van een dsp-controller is de hardware-support voor loops in combinatie met loopfolding. Loopfolding is niets anders dan software-pipelining toegepast op loopbodies (figuur 3). Toegepast op het voorbeeld van de productaccumulatie betekent dit dat verschillende executies van de loopbody elkaar kunnen overlappen mits de nodige resources voorhanden zijn. Tijdens de productaccumulatie van het huidige sample kunnen we reeds het volgende sample lezen en ook reeds het daaropvolgende adres berekenen. Merk op dat de pipeline vóór de loop moet worden opgebouwd (preamble) en na de loop weer moet worden afgebouwd (postamble).

Assembler

Omwille van efficiëntie-redenen worden de meeste dsp's nog op assembly-niveau geprogrammeerd. Dit is echter zeer tijdrovend. Hierdoor ontstaat er een steeds grotere behoefte aan goede compilers. Met name het overzien van al het parallelisme is zeer lastig. Andere kenmerken, typisch voor dsp's zijn de gedistribueerde registers en de onregelmatige interconnects. Ook dit maakt het lastig voor dergelijke architecturen een goede compiler te schrijven.

De structuur van een dsp-compiler bestaat grofweg uit twee delen. Het *front-end* bestaat uit lexicale analyse, syntaxisanalyse en semantische analyse. Dit geeft een intermediaire representatie van de code. Deze representatie is nog onafhankelijk van de specifieke processor hardware. De *code-generatie* is de meest kritische stap. Code-generatie zelf kan weer opgedeeld worden in drie fasen: code-selectie, register-allocatie en scheduling. *Code-selectie* is het proces waarbij de

intermediaire representatie bedekt wordt met zogenaamde *register transfer patterns* (RTP). Een RTP is een primitieve atomaire operatie waarbij inputs worden gelezen van één of meerdere geheugenelementen van de processor, vervolgens een bewerking op deze data plaatsvindt, en ten slotte het resultaat geschreven wordt in één of meer geheugenelementen.

Register allocatie beslist in welke registers programvariabelen worden opgeslagen.

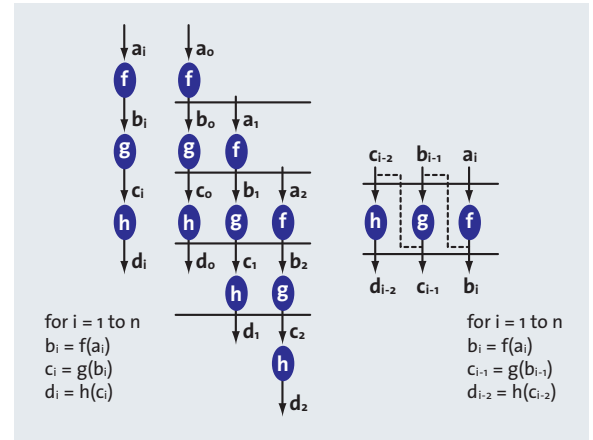
Scheduling ten slotte beslist welke operaties samen in één instructie worden uitgevoerd en bepaalt tevens de volgorde tussen deze instructies.

Dsp's halen veelal een verhoging van de rekenkracht uit parallelisme en niet puur alleen uit de klokfrequentie zoals bij programmeerbare cpu-cores. Belangrijkste reden hiervoor is de vermogensdissipatie. Om dit parallelisme zoveel mogelijk in een enkele instructie aan te kunnen sturen zijn de *very long instruction word*-architecturen (VLIW) bedacht. De bedoeling ervan is *instruction level parallelism* (ILP) toe te laten zonder de regelmaat in de architectuur te verliezen. ILP maakt gebruik van een aantal ontwerptechnieken om de executie te versnellen door verschillende risc-achtige operaties parallel uit te voeren.

Kenmerken van de VLIW-architecturen zijn de aanwezigheid van meerdere *functionele units* (FU's) die parallel aan elkaar kunnen opereren. FU's zijn hierbij vaak gepipelined en werken veelal met operanden uit registers. Soms overstijgt het aantal FU's een dusdanig aantal dat om deze allemaal tegelijkertijd aan te sturen het instructiewoord wel heel breed zou worden (denk aan de Trimedia). Vaak wordt hiervoor een beperking ingebouwd tot een bepaald aantal *issue slots*. De breedte van het instructiewoord en de daarbij horende code size vormen het voornaamste onderdeel van VLIW-architecturen, met name voor embedded applicaties.

Parallelisme verhogen

Een belangrijke rol voor een dsp-compiler is het ILP te detecteren en hiervoor een zo kort mogelijk *schedule* te genereren. In tegenstelling tot superscalar-processoren waar scheduling-hardware gebruikt wordt om op run-time parallelisme te detecteren gebeurt dit in geval van een VLIW dus op compile-time. De controle-hardware in een pro-



3. Een belangrijk kenmerk van een dsp-controller is de hardware-support voor loops in combinatie met loopfolding. Loopfolding is niets anders dan software-pipelining toegepast op loopbodies.

cessor wordt daardoor eenvoudiger.

Een andere vaak toegepaste techniek om het parallelisme te verhogen binnen een FU is subwoord-parallelisme. Bij de Trimedia heet dit custom-operators. Hierbij worden bijvoorbeeld vier afzonderlijke databytes samengepakt in één 32-bit woord. Daarna worden in één FU vier identieke operaties uitgevoerd op de aparte bytes. Een dergelijke instructie wordt aangeduid met *quadadd* (viermaal ADD). Dit soort custom-instructies wordt echter niet door een compiler herkend en vereist het handmatig herschrijven van de applicatie. Subwoord-parallelisme zien we veelal toegepast worden voor graphics-processoren. Hier is een trend naar 128 bit brede datapaden. Daarin worden dan vier 32-bit floating point getallen gepakt. Een typisch voorkomende operatie bij graphics is een 4x4 matrix vectorvermenigvuldiging. Deze VLIW's worden soms aangeduid onder de naam mediaprocessoren, die in feite programmeerbare dsp's zijn. Tot dezelfde groep worden ook programmeerbare cpu's met multimedia-extensies gerekend, zoals Intel MMX of PA-RISC van Hewlett Packard. □

Achtergrond

Auteur Ger Schoeber is consultant software- en systeem-architectuur voor Ordina Media. Ordina Media is de nieuwe naam voor High Tech Automation in Son. Het bedrijf profileert zich als strategische partner bij de totstandkoming van de nimmer tot rust komende stroom van nieuwe, spectaculaire mediaproducten zoals *flat tv*, *DVD rewritable* en interactieve tv.

Met dank aan Jef van Meerbergen en Bart Mesman (Philips Research/EESI). Uit hun materiaal is dit artikel tot stand gekomen. Het volgende artikel zal de applicatiedomein-specifieke processoren belichten.