Co-design van hardware en software

HSE heeft de Toekomst

Omdat in ingebedde systemen hardware en software zeer nauw met elkaar verbonden zijn, lijkt het nuttig dat geheel te kunnen specificeren op een manier die los staat van de implementatie. Deze wijze van ontwerpen krijgt langzamerhand enig bekendheid als hardware/software co-design of ook wel hardware/software-engineering (HSE). Het idee is om de splitsing van het systeem in elektronica en programmatuur zo lang mogelijk uit te kunnen stellen met als doel, uiteraard, optimalisatie van prestatie en kostprijs. Dit ideaal is vooralsnog beperkt haalbaar maar in een tweede, lossere zin staat HSE ook voor parallel ontwerpen van hardware en software. Vooral de integratie van beide moet hiermee beter te beheersen zijn. Door exacte specificatie van hardware en software is het systeem bovendien als geheel te simuleren.

Hans van Thiel

Het is een bekend gezegde in de elektronica dat alles wat in hardware kan worden uitgevoerd ook in software is te implementeren - en omgekeerd. Vuistregel daarbij is dat hardware sneller is maar duurder en tijdrovender in ontwerp en produktie.

Ingebedde systemen bestaan per definitie uit programmeerbare IC's en daarop draaiende programmatuur maar ook hierin zijn twee polen te onderscheiden. Hardware-georiënteerde systemen zijn gebouwd op digitale signaal processoren (DSP) of zelfs speciaal ontworpen Application Specific Instruction Processors (ASIPS). Dergelijke systemen zijn in de regel ontworpen om een beperkt aantal toepassingsgerichte algorithmen uit te kunnen voeren. Software-gerichte systemen implementeren hun functionaliteit in programmatuur. De hardware is niet specifiek voor de toepassing en zeker als ook nog een Real Time Operating System (RTOS) wordt benut om standaard functies te realiseren blijft de hardware op de achtergrond. Tussen deze twee polen in staan de heterogene systemen. Voor deze systemen zal de ontwikkelaar misschien niet èn hardware èn software van de grond af aan moeten ontwerpen, maar hij of zij zal beide wel zorgvuldig in overweging moeten nemen en op elkaar afstemmen.

Systeembeschrijving

In het ideale geval heeft de ontwerper een model ter beschikking om de systeemfunctionaliteit te kunnen beschrijven zonder met de implementatie in software en hardware rekening te hoeven houden. Aan dergelijke modellen werd en wordt veel onderzoek verricht maar het nadeel voor praktische toepassing is nog altijd de complexiteit en de daarbij behorende uitgebreidheid. Het opstellen van specificaties is bovendien vaak zo ingewikkeld en moeilijk dat de correctheid ervan niet goed is vast te stellen. Het voordeel dat deze methoden bieden boven de tradiotionele, op ervaring van het ontwerpteam gebaseerde werkwijze wordt daarmee een stuk minder. Dit neemt niet weg dat zij in principe beter zijn en in de toekomst zeker belangrijker zullen worden.

Een zo'n model is dat van Concurrent Sequential Processes (CSP), dat, zoals de naam al zegt, gebaseerd is op processen en hun synchrone of asynchrone communicatie. Een ander gebruikt eindige toestandsautomaten oftewel Finite State Machines (FSM). Deze leggen een systeem vast als een verzameling toestanden en de stappen waardoor die toestanden in elkaar overgaan. Met name ingebedde systemen met een regelende functie zijn soms te specificeren als samenwerkende FSM's. Data Flow Graphs (DFG) zijn gebaseerd op datastromen en dus beter geschikt voor digitale signaalverwerkingssystemen (zie referentie 1).

Voor de genoemde modellen - en andere, niet genoemde - bestaan talen of grafische formalismen. StateCharts en Esterel, SDL, Cossap en SPW zijn er enkele. Volgens referentie 2 zijn Esterel en StateCharts allebei synchrone talen die verschillende hierarchiën en entiteiten zoals activiteiten, toestanden en modulen ondersteunen en ligt de kracht van Esterel in de mogelijkheden om er de tijd in uit te drukken. Beide zouden minder geschikt zijn voor gedistribueerde systemen. Cossap en SPW zijn commerciële omgevingen die op DSP-verwerking zijn gericht. De Specification and Description Language (SDL) is een ITU-standaard (International Telecommunications Union) die echter beperkt is in rekenkundige uitdrukkingskracht en in timing. Tenslotte is ook de Unified Modelling Language (UML) bestemd om een systeem te beschrijven, onafhankelijk van de implementatie.

Meer talen

Als alternatief voor één systeemtaal kan de ontwerper er voor kiezen een (voorlopige) splitsing te maken in software en hardware en die te beschrijven in een programmeertaal als C en een hardware-beschrijvingstaal als VHDL (Very high speed integrated circuit Hardware Description Language). Het voordeel van co-design in C en VHDL is dat beide talen goed zijn gedefinieerd en worden ondersteund door tools die zich in de praktijk hebben bewezen. Het parallel ontwerp geeft een goed inzicht, verkort de ontwerptijd en geeft bovendien de mogelijkheid to co-simulatie.

Figuur 1, uit referentie 2, toont het ontwerpschema. In onderlinge samenhang wordt de functionaliteit op het hoogste niveau geschreven in VHDL en C. De C-code wordt gecompileerd en geassembleerd naar de executeerbare binaire code die de software vormt. De VHDL-beschrijving wordt omgezet in RTL (Register Transfer Level) VHDL en vervolgens naar VHDL op poortniveau. Uit deze beschrijving kan de chip daadwerkelijk worden gegenereerd.

De hardware/software-engineering oftewel co-design is aangegeven door de middelste blokken. De instructieset van de processor bepaalt hoe de compiler de C-code moet omzetten. Voor een ASIP zal uiteraard een aparte compiler moeten worden gegenereerd. De assembleertaal van de processor en de RTL VHDL-beschrijving horen beide bij het (klokcyclusniveau) model van de processor. Het onderste blok geeft het ingebedde systeem weer als daadwerkelijk geïmplementeerde elektronica met de bijbehorende programmatuur.

Simulatie

Een groot voordeel van de geschetste ontwerpmethode is dat het een tot op klokcyclusniveau gespecificeerd model oplevert van de processor waarmee systeemsimulaties kunnen worden uitgevoerd.

Dat kan niet direct, maar hardware en software kunnen wel zodanig op een computer worden

gesimuleerd dat beide processen met elkaar kunnen communiceren. Deze co-simulatie, schematisch aangegeven in figuur 2, kan verlopen via op het gastsysteem aanwezige standaardprotocollen maar ook in een speciaal programma worden geïmplementeerd. Zowel C als VHDL hebben de mogelijkheid om procedures in andere talen aan te roepen, wat dit vergemakkelijkt. Schematisch verloopt de communicatie via de co-simulatiebus.

Met name control-systemen bestaan behalve uit software en elektronica uit mechanische componenten die door de eerste worden aangestuurd. Voor mechanische (sub)systemen zijn uitgebreide simulatiemogelijkheden beschikbaar, bijvoorbeeld in het Matlab-pakket. Met het in figuur 2 aangegeven co-simulatieprincipe kan men mechanische subsystemen tegelijk met de elektronica en de software mee simuleren.

Voorbeeld van HSE

Een combinatie van de aangegeven ontwerpmethoden staat beschreven in referentie 3. Het gaat hierbij om het ontwerp van een ATM Network Interface Card (NIC). De kaart moest een protocolstack van vier lagen implementeren, nl. ATM (Asynchronous Transfer Mode), AAL (ATM Adaptation Layer), IP (Internet Protocol) en TCP (Transmission Control Protocol). De TCP-laag verzorgt een betrouwbare communicatie tussen twee processen met verschillende snelheden. De IP-laag maakt datagrammen van de TCP-segmenten en de AAL-laag zet die om naar pakketten die tenslotte in de onderste laag worden omgezet in standaard ATM-cellen van 53 byte. Aan de ontvangende kant verloopt het proces (conform het lagenmodel) in omgekeerde richting.

Het co-ontwerp van de kaart werd uitgevoerd volgens het schema van figuur 3 en combineerde het gebruik van drie tools. Allereerst werd het hele systeem gespecificeerd in de reeds genoemde Specification and Description Language (SDL). Voor zowel de specificatie als de simulatie van het systeem werd een tool Object Geode gebruikt. De hardware/software-splitsing werd vervolgens uitgevoerd op de SDL-specificatie met een tool Cosmos. Cosmos produceert uit SDL-specificaties een gedistribueerd model van softwarepocessen die gespecificeerd zijn in C en hardwareprocessen gespecificeerd in VHDL. De co-simulatie van deze modellen werd uitgevoerd met VCI. Alle drie de tools schijnen overigens nog min of meer experimenteel te zijn en worden zeker niet algemeen gebruikt buiten onderzoekstoepassingen. SDL is een ITU-standaard.

De onderzoekers ontwierpen met Cosmos vijf verschillende HW/SW-partities die ze alle vijf ook simuleerden. De snelheden van de gegene-

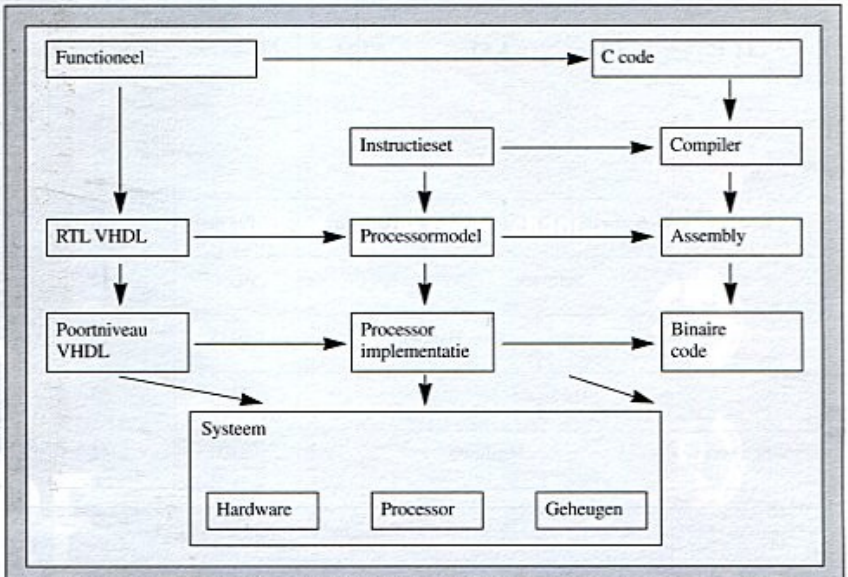

Fig. 1. C-VHDL co-ontwerp volgens Jerraya et al. (referentie 2).

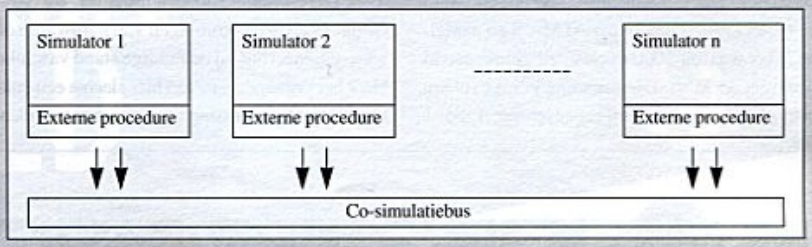

Fig. 2. Co-simulatie van modulen volgens Jerraya et al. (referentie 2)

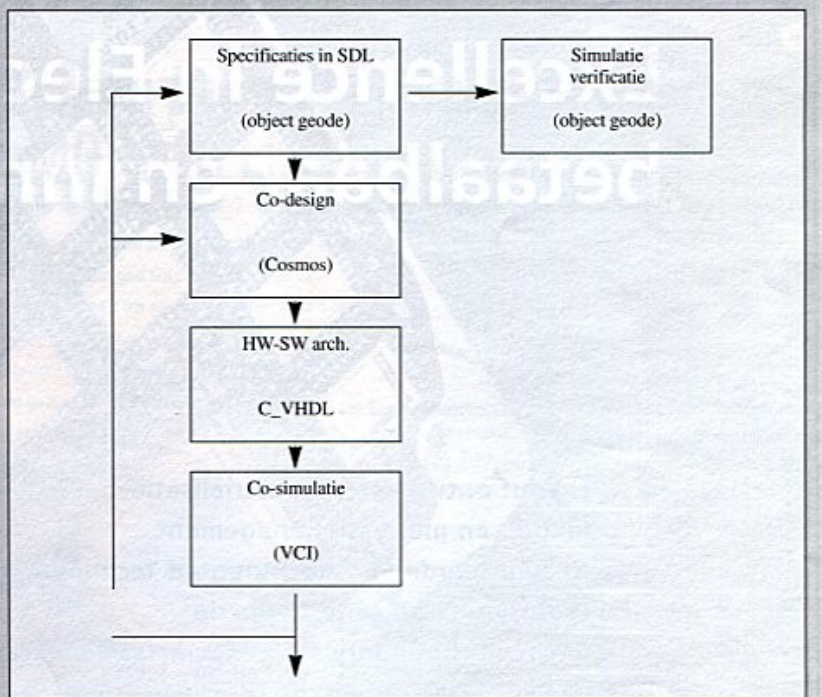

Fig. 3 Het ontwerp van een ATM Network Interface Card volgens Daveau et al. (referentie 3). Tussen haken staan de gebruikte tools aangegeven.

TCP	IP	AAL	ATM	*Klokcycl. / cel*	*Mbit / s*
Software			Hardware	19890	6
Software				10550	12
Software		Hardware		6445	19
Hardware				2110	60
Software	Hardware			3100	41

Tabel. Hardware /software co-simulaties van een ATM Network Interface Card bij Daveau et al. (referentie 3).

reerde ontwerpoplossingen werden uitgedrukt in aantallen klokcyclussen per ATM-cel en aantallen Mb/s waarbij 5000 cycles/cell gelijkgesteld werd met 25 Mb/s. Die aanname komt overeen met gebruik van Pentium processoren. Tabel 1 toont de resultaten van de simulaties.

Ondanks de tekortkomingen van hardware/software-engineering bij de huidige stand van zaken biedt het principe, waarvan hier slechte een enkele onderzoekstoepassing werd geschetst, evidente voordelen. Ontwerpen van zowel hardware als software in één ontwikkel-lifecycle en met een samenhangende toolset heeft zeker de toekomst en voor ingebedde systemen ligt de toekomst meestal dichterbij dan verwacht. ■

Referenties:

1. Vranken, H.P.E.: *„Design for Test & Debug in Hardware/Software Systems"*, Proefschrift TU Eindhoven. (1998).
2. Jerraya, A.A. et al.: *„Multilanguage Specification for System Design and Codesign"*. TIMA Laboratory, 46 Avenue Félix Viallet, 38031 Grenoble (Frankrijk). Te downloaden van: http://verdon.imag.fr.
3. Daveau, J.M. et al.: *„Hardware/Software Co-Design of an ATM Network Interface Card: a Case Study"*. TIMA Laboratory, 46 Avenue Félix Viallet, 38031 Grenoble (Frankrijk). Te downloaden van: http://verdon.imag.fr.

Verder informatie:

ITU-TZ.100 Functional Specification and Description Language, Recommendation Z.100 - Z.104, Maart 1993. Staunstrup, J., Wolf, W.: *„Hardware/Software Co-Design: Principles and Practice"*. Kluwer Academic Publishers. *(1997)*