Reconfigurable Computing:
the Roadmap to a New Business Model - and its Impact on SoC Design

(invited embedded tutorial)
Reiner Hartenstein, University of Kaiserslautern, Germany
http://www.fpl.uni-kl.de hartenst@rhrk.uni-kl.de

Abstract. *Making gate arrays obsolete, FPGAs are successfully proceeding from niche to mainstream. But after a decade of research on Reconfigurable Computing a new breed of reconfigurable platforms is an emerging competitor to FPGAs: coarse grained reconfigurable platforms with drastically improved efficiency.*

Like microprocessor usage, also programming reconfigurable platforms is RAM-based, but by structural programming instead of procedural programming. But so far reconfigurable platforms do not yet repeat the RAM-based success story of the software industry. Because of lacking awareness of this paradigm switch there is not yet a configware industry. A new business model is needed, as well as a fundamentally new product design flow approach.

This embedded tutorial surveys a decade of R&D on Reconfigurable Computing and related CAD. The paper illustrates, that results are available for commercialization: to cope with the current SoC design crisis by a transisiton from hardware/software co-design to platform-based SoC design by configware/software co-compilation. The paper shows a roadmap to the success story of a coming configware industry.

1. Introduction

Reconfigurable platforms are heading from niche to mainstream [1], bridging the flexibility gap between ASICs and microprocessors. It's time to revisit R&D results to derive a roadmap to SoC design and emerging new business model.

2. Coarse-Grained Reconfigurable Architectures

In contrast to using FPGA use (fine grain reconfigurable) the area of *Reconfigurable Computing* stresses the use of coarse grain reconfigurable arrays (RAs) with pathwidths greater than 1 bit, because fine-grained architectures are massively less efficient, due to a huge reconfigurability overhead and poor routability [2] [3]. Since computational datapaths have regular structure potential, full custom designs of *reconfigurable datapath units* (rDPUs) are drastically more area-efficient. Coarse-grained architectures provide operator level CFBs, and very area-efficient datapath routing switches. A major benefit is massive reduction of configuration memory and configuration time, and drastic complexity reduction of the P&R (placement and routing) problem. Several architectures will be briefly outlined (for more details see [4]). Some of them introduce *multi-granular* solutions, where more coarse grain can be achieved by bundling of resources, like 4 ALUs of 4 bits to obtain a 16 bit ALU.

2.1 Primarily Mesh-Based Architectures

Mesh-based architectures arrange their PEs mainly as a rectangular 2-D array with horizontal and vertical connections which supports rich communication resources for efficient parallelism. and encourages nearest neighbour (NN) links between adjacent PEs (NN or 4NN: links to 4 sides {east, west, north, south}, or, 8NN: NN-links to 8 sides {east, north-east, north, north-west, west, south-west, south, south-east} like w. *CHESS array*: [5]). Typically, longer lines are added with different lengths for connections over distances larger than 1. *DP-FPGA (Datapath FPGA)* [6] has been introduced to implement regularly structured datapaths. It is a FPGA-like mixed fine and coarse grained architecture with 1 and 4 bit paths. Its fabric includes 3 component types: control logic, the datapath, and memory. The datapath block consists of 4 bit-slices: each bit-slice with a lookup table, a carry chain and a 4 bit register. DP-FPGA provides separate routing resources for data (horizontal, 4 bits wide) and control signals (vertical, single bit). A third resource is the shift block to support single-bit or multi bit shifts and irregularities.

The KressArray is primarily a mesh of rDPUs physically connected through wiring by abutment: no extra routing areas needed. In 1995 it has been published [7] as "rDPA" (reconfigurable DataPath Array). "KressArray" has been coined later. The KressArray is a super-systolic array (generalization of the systolic array) which is achieved by *DPSS* (see section 3.2). Its interconnect fabric distinguishes 3 physical levels: multiple unidirectional and/or bidirectional NN links (fig. 1), full length or segmented column and/or row backbuses, a single global bus reaching all rDPUs (also for configuration). Each rDPU can serve for routing only, as an operator, or, an operator with extra routing paths. All connect levels are layouted over the cell, so that wiring by abutment capability is not affected. A first 32 bit KressArray included an additional control unit for the *MoM-3* [8] *Xputer* [9] [10] [11] [12]. Its rDPUs support all C language operators. With the new Xplorer environment [13] rDPUs also support any other operator repertoires including branching, and loops. I/O data streams from and to the array can be transferred by global bus, array edge ports, or ports of other rDPUs (addressed individually by address generator). Supported by the DPSS application development tool and a platform architecture space explorer (PSE) environment the basic principles of the KressArray define an entire family of KressArrays covering a wide but generic variety of interconnect resources and functional resources. A later PSE version (see section 4.2), supports the rapid creation of RA and rPDU architectures optimized for a particular application domain, and rapid mapping of applications onto any RA of the family.

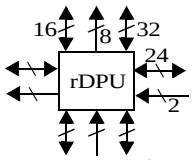

Fig. 1: KressArray NN ports examples.

Colt [14] combines concepts from FPGAs and data flow computing [15]. It's a 16 bit pipenet [16] with mesh-connected IFUs (Interconnected Functional Units), a crossbar switch, an integer multiplier, and six data ports, and relies highly on runtime reconfiguration using wormhole routing. Each IFU features an ALU, a barrel shifter to support multiplication and floating point. *MATRIX* [17] is a multi-granular array of 8-bit BFUs (Basic Functional Units) with procedurally programmable microprocessor core including ALU, multiplier, 256 word data and instruction memory and a controller which can generate local control signals from ALU output by a pattern matcher, a reduction network, or, half a NOR PLA. The routing fabric

provides 3 levels of 8-bit buses: 8 nearest neighbour (8NN) and 4 second-nearest neighbour connections, bypass connections of length 4, and global lines. For more details also see [4]. The Garp architecture [18] resembles an FPGA and comes with a MIPS-II-like host and, for acceleration of specific loops or subroutines, a 32 by 24 RA of LUT-based 2 bit PEs. Basic unit of its primarily mesh-based architecture is a row of 32 PEs. The host has instruction set extensions to configure and control the RA. Garp has a sophisticated routing architecture. *RAW (Reconfigurable Architecture Workstation)* [19] provides a 4 by 4 array RISC multi processor architecture of NN-connected 32-bit modified MIPS R2000 microprocessor tiles with ALU, 6-stage pipeline, floating point unit, controller, register file of 32 general purpose and 16 floating point registers, program counter, local cached data memory and instruction memory. *REMARC (Reconfigurable Multimedia Array Coprocessor)* [20], a reconfigurable accelerator, tightly coupled to a MIPS-II RISC processor, consists of an 8 by 8 array of 16 bit "nanoprocessors" with memory, and a global control unit. It uses NN connections and 32 bit horizontal and vertical buses which also allow some broadcast to processors, also to support SIMD operations.

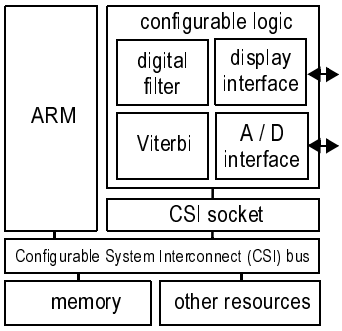

Fig. 2: Triscend cSoC example.

The hexagonal *CHESS array* [5] features a chessboard-like floorplan with interleaved rows of alternating ALU / switchbox sequence and includes embedded RAM areas. Switchboxes can be converted to 16 word by 4 bit RAMs. The interconnect fabrics has segmented 4 bit and 16 bit buses of different length. An ALU data output may feed the configuration input of another ALU, so that its functionality can be changed on a cycle-per-cycle basis at runtime without uploading. Partial configuration by uploading is not possible. *The DReAM Array (Dynamically Reconfigurable Architecture for Mobile Systems [21])* for next generation wireless communication is an array of RPUs. Each RPU consists of: 2 dynamically reconfigurable 8-bit Reconfigurable Arithmetic Processing (RAP) units, 2 barrel shifters, a controller, two 16 by 8-bit dual port RAMs (used as LUT or FIFO), and, a Communication Protocol Controller. The RPU array fabric uses NN ports and global buses segmentable by switching boxes.

Chameleon Systems' CS2000 family multi-protocol multi-application reconfigurable platform RCP (reconfigurable communication processor) [22] aims at initial markets in communication infrastructure and is intended to cope with the chaotic world of evolving standards, protocols and algorithms with application areas as 2nd and 3rd generation wireless basestations, fixed point wireless local loop (WLL), smart antennas, voice over IP (VoIP), very high speed digital subscriber loop (DSL), and, for instance, supports 50 channels of CDMA2000. CS2000 chips have a 32 bit RISC core, connected to a RA of 6, 9, or 12 reconfigurable tiles, with 7 32-bit rDPUs (each including an 8 word instruction memory), 4 local memory blocks of 128 x 32 bits, 2 16x24-bit multipliers. The *MECA family* of DSPs, optimized for VoIP, by compressing voice into ATM or IP packets etc., aims at next generation VoIP and VoATM. Compared to conventional DSPs- a speed-up factor of 10 is reported. *CALISTO (Configurable ALgorithm-adaptive Instruction Set TOpology)* is an adaptive instruction set architecture for internet protocols (IP) and ATM packet-based networks with flexibility for Any-Service-Any-Port (ASAP) to deliver voice and data simultaneously over a unified data network. It is a communications processor for carrier-class voice gateways, soft switches, and remote access concentrators/remote access servers (RAC/RAS), aiming at use like echo cancellation, voice/fax/data modems, packetization, cellification, delay equalization. The multi-context (2 extra configuration memories) *FIPSOC (Field-programmable System-on-Chip)* ASIC emulator (rapid prototyping), has an 8051 controller, a 8x12, 8x16, or 16x16 RA of 4 bit "digital macro cells" (DMC) and a RAA (reconfigurable analog array). with configurable analog blocks (CAB) usable as differential amplifiers, comparators, converters etc.

2.2 Linear-Array-based Architectures

Some RAs are based on one or several linear arrays, mainly aiming at mapping pipelines onto it. *RaPiD (Reconfigurable Pipelined Datapath)* [23] aims at deep pipelines for highly regular, computation-intensive tasks. It is a 1-D RA. featuring 15 DPUs of 8 bit with integer multiplier (32 bit output), 3 integer ALUs, 6 general-purpose datapath registers and 3 local 32 word memories, all 16 bits wide. ALUs can be chained. RaPiD includes an I/O stream generator with address generators and FIFOs. RaPiD's sophisticated routing and configuration interconnect fabric cannot be detailed here.

PipeRench [25], an accelerator for pipelined applications, provides several reconfigurable pipeline stages ("stripes") and relies on fast partial dynamic pipeline reconfiguration and run time scheduling of configuration streams and data streams. PipeRench allows the configuration of a pipeline stage each cycle, while concurrently executing all other stages. The sophisticated fabric consists of (horizontal) stripes of interconnect and PEs. A stripe provides 32 ALUs with 4 bits each. The whole fabric has 28 stripes.

Paradigm	Platform	Programming source
"von Neumann"	*Hardware*	*Software*
Xputer	coarse grain *Flexware*	high level *Configware*
RL (FPGA etc.)	fine grain *Flexware*	netlist level *Configware*

Fig. 3: About terminology.

2.3 Architectures using Crossbars

PADDI (Programmable Arithmetic Device for DSP) stands for architectures for rapid prototyping of computation-intensive DSP data paths, featuring sophisticated fabrics using a central reduced crossbar (difficult to rout) and a 2 level hierarchy of segmentable buses. *PADDI-1* [26] [27] consists of clusters of 8 arithmetic execution units (EXUs), 16 bits wide, including 8 word SRAM (which may be con-catenated for 32 bits). *PADDI-2* [28] features a data-driven execution mechanism and has 48 EXUs, 16 bits wide. The *Pleiades* Architecture [29] is a kind of generalized low power "PADDI-3" with microprocessor and heterogeneous RA of EXUs, which allows to mix fine and coarse grained EXUs, and, have memories in place of EXUs.

2.4 Future Reconfigurable Architectures

A universal RA obviously is an illusion. The way to go is toward ASPPs (application-specific programmable products) like sufficiently flexible RAs, optimized for a particular application domain like e. g. wireless communication, image processing or multimedia etc. There is a need for tools supporting such dedicated RA architecture development. But architectures have an immense impact on implementability of good mapping tools. "Clever"fabrics are too sophisticated to find good tools. Best are simple generic fabrics architecture principles, or, a mapping tool which generically creates by itself the architectures it can manage easily [13], or, a combination of both approaches like platform space exploration (s. section 4.2).

3. Programming Coarse Grain RAs

Programming frameworks for RAs are highly dependent on structure and granularity, and differ by language level. For MATRIX, PADDI-2 and REMARC it's assembler level. Some support the designer by a graphical tool for manual P&R. Others feature automatic design flow from HDL or

high-level programming language. Environments differ by the approach used for technology mapping, placement, routing. Using only a simple script for technology mapping [30] DP-FPGA [6] is not considered. Technology mapping is mostly simpler for coarse grain than for FPGAs. Approaches are: direct mapping, where the operators are mapped straight forward onto PEs, with one PE for one operator, or, using an additional library of functions not directly implementable by one PE, or, more sophisticated tree matching also capable to merge several operators into one PE by a modified FPGA tool kit. An exception is the RAW compiler doing partitioning instead of technology mapping, since RAW has RISC cores as PEs accepting blocks from program input.

For operator placement, the architecture has an impact. An approach often used for FPGAs synthesis is placement by simulated annealing or a genetic algorithm. Garp uses a tree matching algorithm instead, where placement is done together with technology mapping. The use of greedy algorithms is feasible only for linear arrays (PipeRench), or with a high level communication network (RAW). PADDI is an exception by using a scheduling algorithm for resource allocation.

Routing also features quite different approaches. In two cases, the routing is not done in an extra phase but integrated into the placement and done on the fly. One approach (KressArray) uses a simple algorithm restricted to connects with neighbours and targets with at most the distance of one. The other (RaPiD) employs the pathfinder algorithm [30], which has been developed for FPGA routing. Greedy routing would be not satisfying. General exceptions to the routing approaches is the RAW architecture, which features only one high-level communication resource, so no selection of routing resources is needed, and the PADDI architecture, which features a crossbar switch having the same effect. Greedy

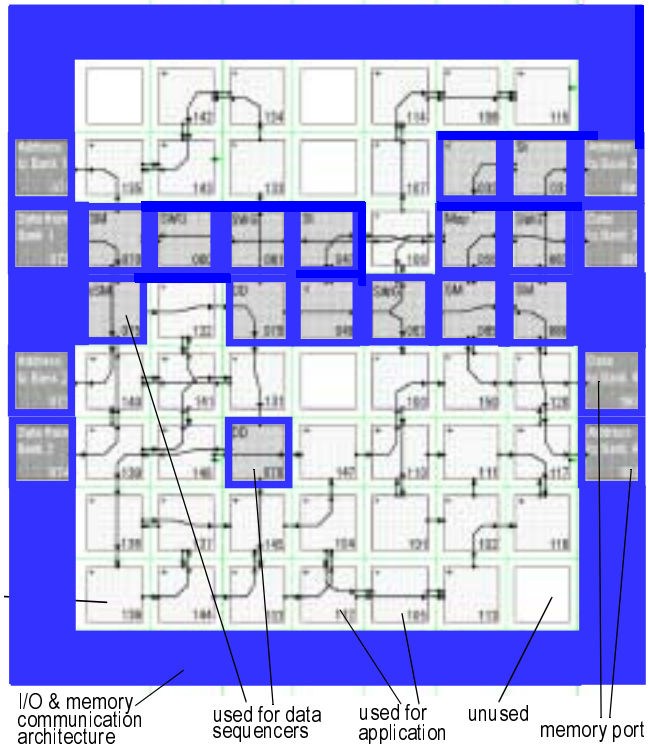

Fig. 4: Mapping application (linear filter) and memory communication architecture (dark background) onto the same KressArray, including the address ports and the data ports to 4 different memory banks (5 of 8 memory port connects are routed through application DPUs).

mapping	Kress DPSS	CHESS	RaPiD	Colt
placement	simulated annealing	simulated annealing		genetic algorithm
routing		Pathfinder		greedy algorithm

Fig. 5: FPGA-Style Mapping for coarse grain reconfigurable arrays.

routing algorithms are only used for 1-D RAs, or architectures capable to cure routing congestion by other mechanisms, like Colt with wormhole run-time reconfiguration.

3.1 Assembler Programming

Assembler level code for coarse grain architectures can be compared to configuration code for FPGAs. In the case of systems comprising a microprocessor / RA symbiosis, only the reconfigurable part is considered for classification. Programming is done mainly at a kind of assembler level for PADDI-2, MATRIX, and, RAs of REMARC. For Programming PADDI-2 [28], a tool box has been developed which includes software libraries, a graphical interface for signal flow graphs, routing tools, simulation tools, compilation tools and tools for board access and board debugging. Major parts of this process are done manually. The input specifies assembly code for each function in the signal flow graph. The programmer manually partitions the signal flow graph with a graphical tool, which also aids in manual placement and routing. As an alternative to manual placement and routing, an automated tool is provided, which guarantees to find a mapping, if one exists, by exhaustive methods which need much computation time. For Programming MATRIX [17] an assembly level macro language has been developed. Some work on P&R pointed out the original MATRIX's weak points [31]. REMARC tools [20] allow concurrent C programming of RISC processor and RA using a GCC compiler also generating RISC instruction code to invoke REMARC code.

3.2 Frameworks with FPGA-Style Mapping

Like known from mapping onto FPGAs CHESS, Colt, KressArray, and RaPiD (see fig. 5) use simulated annealing or other genetics for placement, and two use pathfinder for routing [30]. The KressArray DPSS (Datapath Synthesis System) accepts a C-like language source. Compilation for RaPiD works similar, but relies on relatively complex algorithms. Colt tools use a structural description of the dataflow. CHESS has been programmed from a hardware description language (JHDL) source. P&R quality has a massive impact on application performance. But, due to the low number of PEs, P&R is much less complex than for FPGAs and computational requirements are drastically reduced. Tools for Colt [14] accept a dataflow description (below C level) for placement by a genetic algorithm and routing by a greedy algorithm (routing congestion is cured at run-time by wormhole reconfiguration). Data stream headers hold configuration data for routing and the functionality of all PEs encountered.

Programming RaPiD [23] uses RaPiD-C, a C-like language with extensions (like synchronization mechanisms and conditionals for loops) to explicitly specify parallelism, data movement and partitioning, Outer loops are transformed into sequential code for address generators, inner loops into structural code for the RA. The netlist is mapped onto RaPiD by pipelining, retiming, and P&R by simulated annealing, with routing (by pathfinder [30]) done on the fly to measure placement quality [32]. To Program the CHESS array [5] a compiler [33] was implemented accepting JHDL [34] sources and generating CHESS netlists. Placement is done by simulated annealing and routing by Pathfinder [30].

3.3 Other mapping approaches

Greedy algorithms are poor in mapping to FPGAs. But, although Garp is mesh-based, mapping treats it like a linear array which allows mapping in one step by a simple greedy routing algorithm. RAW features only one communication resource, removing the wire selection problem from routing. Instead, the compiler schedules time multiplexed NN connections. CPU cores inside RAW PEs simplify mapping by loading entire source code blocks. PipeRench resembling a linear array and interconnect fabrics restrictions keep placement simple for a greedy algorithm. PADDI uses a standard P&R approach.

Garp tools[18] use a SUIF-based C compiler [35] to generate code for MIPS host with embedded RA configuration code to accelerate (only non-nested) loops. It also generates interfacing instructions for the host, and a DFG (data flow graph). The proprietary Gamma tool [36] maps the DFG onto Garp using a tree covering algorithm Configuration code is generated (incl. routing [38]), assembled into binary form, and, linked with the hosts C object code. For more details also see [39]. RAW tools [40] [41] include a SUIF-based C compiler and a run-time system managing dynamic mechanisms like branch prediction, data caching [42], speculative execution, dynamic code scheduling. For details see [4]. The RAW project aims more at parallel processing rather than reconfigurable computing and failed in finding a good automatic mapping algorithm [43].

PipeRench tools [25] [44] use the DIL single-assignment language (SAL) for design entry and as an intermediate form. First, the compiler inlines all modules, unrolls loops and generates a straight-line SA program (SAP). After optimizations and breaking the SAP into pieces fitting on one stripe, a greedy P&R algorithm is run which tries to add nodes to stripes. Once placed, a node is routed and never moved again. P&R is fast by crossbar switch usage, coarse granularity, and, restriction to unidirectional pipelines. CADDI [45], assembler and simulator, has been implemented for PADDI. First a silage [46] specification is compiled into a CDFG (control /data flow graph), used for estimations of critical path, minimum and maximum bounds for hardware for a given time allocation, minimum bounds of execution time, and for transformations like pipelining, retiming, algebraic transformations, loop unrolling and operation chaining. The assignment phase maps operations to EXUs by a rejectionless antivoter algorithm [46]. For more details also see [4].

For KressArrays the DPSS (DataPath Synthesis System) [7] generates configuration code for KressArrays from ALE-X high-level language sources [7] [50] supporting datapaths with assignments, local variables and loops. After classical optimizations it generates an expression tree. Next processing steps include a front end, logic optimization, technology mapping creating a netlist, simultaneous P&R by simulated annealing, and I/O scheduling (incl. loop folding, memory cycle optimization, register file usage). The result is the application's KressArray mapping and array I/O schedule. Finally configuration binaries are assembled. Routing is restricted to direct NN connect and rout-through of length 1. Other connect is routed to buses or segmented buses. DPSS has also been part of the MoM-3 Xputer compiler accepting and partitioning a subset of C subset into sequential MoM code and structural KressArray code. The more general CoDe-X approach [51] uses this MoM compiler as part of a partitioning co-compiler accepting a C language superset and partitioning the application onto the host and one or several Xputer-based accelerators.

machine category		Computer ("v.Neumann")	Xputer [10] (no transputer!)
machine paradigm		procedural sequencing: deterministic	(no *dataflow* [15]) data **stream**(s)
driven by:		control flow	
RA support		no	yes
engine principles		instruction sequencing	data sequencing
state register		program counter	(multiple) data counter(s)
communication path set-up		at run time	at load time
data path	resource	single ALU	array of ALUs
	operation	sequential	parallel

Fig. 6: Machine Paradigms.

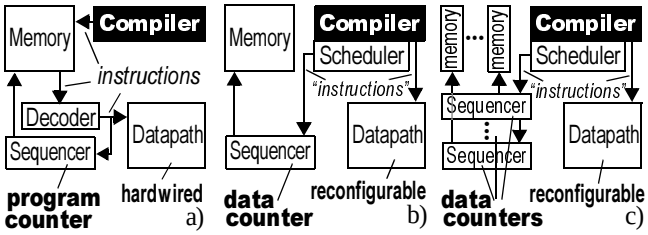

Fig. 7: Machine paradigms: a) v.Neumann, b) Xputer, c) parallel Xputer

3.4 Run-time Mapping

The VIRTEX FPGA family from Xilinx, the RAs being part of the CS2000 series systems from Chameleon and others are run-time reconfigurable. Programming a host/RA combination is a kind of H/S Co-design. However using such devices changes many of the basic assumptions in the HW/SW co-design process: host / RL interaction is dynamic and needs a kind of tiny operating system like eBIOS, also to organize RL reconfiguration under host control. A typical goal is mimization of reconfiguration latency (especially important in communication processors), to hide configuration loading latency, and, list scheduling of eBIOS calls (also see § "CoDe-X ...." in section 4.1).

4. Compilation Techniques

"von Neumann" and the classical compiler are obsolete (fig. 11 a). Today, host/accelerator(s) symbiosis is dominant (fig. 11 b) and most of the platforms summarized above make use of it. Newer commercial platforms include all on a single chip, like Altera's EXCALIBUR combining a core processor (ARM, or MIPS), embedded memory and RL. Sequential code is downloaded to the host's RAM. But accelerators are still implemented by CAD, a C compiler is only an isolated tool, and, *software / configware partitioning is still done manually* [36] [44] [52] [53] [53] [55], so that massive hardware expertise is needed to implement accelerators.

4.1 Co-Compilation

Using RAs as accelerators again changes this scenario: now implementations onto both, host and RA(s) are RAM-based, which allows turn-around times of minutes for the entire system, instead of months needed for hardwired accelerators. This means a

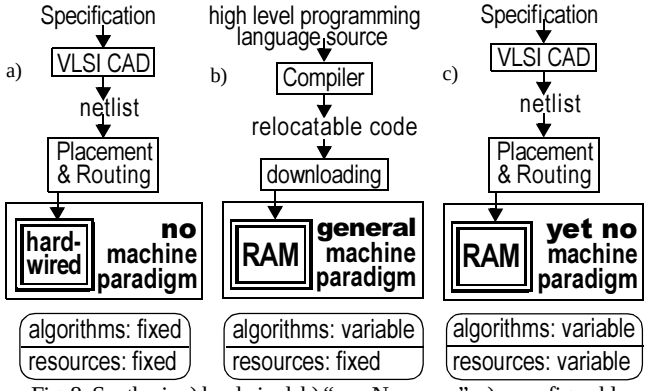

Fig. 8: Synthesis a) hardwired, b) "von Neumann", c) reconfigurable.

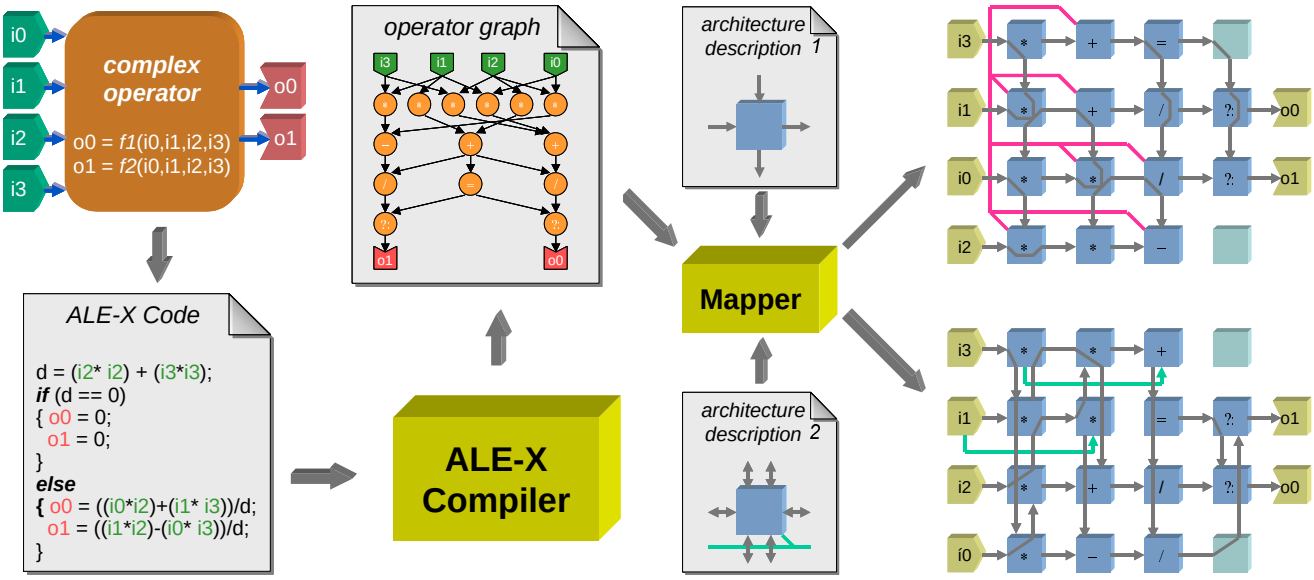

Fig. 9: Simplified example to illustrate platform space exploration (finding an optimized KressArray by KressArray Xplorer [43]).

change of market structure by migration of accelerator implementation from IC vendor to customer, demanding automatic compilation from high level programming language sources onto both, host and RA: *co-compilation* including automatic software / configware partitioning. (fig. 11 c). Since compilers are based on a machine paradigm and "v. Neumann" does not support soft datapaths (because "instruction fetch" is not done at run time: fig. 10) we need a new paradigm (Xputer [55]) for the RA side, where the program counter (fig. 7 a) is replaced by a data counter (*data sequencer* [56]: fig. 7 b). Figure 6 compares both paradigms. With multiple data sequencers (fig. 7 c) a single Xputer may even handle several parallel data streams (example in fig. 4).

CoDe-X is the first such co-compilation environment having been implemented ([51] fig. 12), which partitions mainly by identifying loops suitable for parallelizing transformation [3] [51] [54] into code downloadable to the MoM accelerator Xputer. The *Xputer* Machine Paradigm for soft hardware (fig. 6) [9] [10] [11] [12]. is the counterpart of the von Neumann paradigm. Instead of a "control flow" sublanguage a "data stream" sublanguage like MoPL [57] recursively defines *data goto, data jumps, data loops*, *nested data loops,* and *parallel data loops*. Later on Chameleon Systems reports for CS2000 a co-compilation [22] tool box *C~SIDE*, combining compiler optimization, multithreading to hide configuration loading latency, and, list scheduling to find a 'best' schedule. Whether automatic partitioning is used is undisclosed. C~SIDE includes a GNU C compiler for the RISC host, a HDL synthesizer for the reconfigurable fabric, a simulator, a C-style debugger, a verifier, and eBIOS (eConfigurable Basic I/O Services), a kind of operating system to interfaces the RISC processor with the reconfigurable fabric. C~SIDE also supports run-time reconfiguration (also see [47] [48]). The *IDE (Integrated Development Environment)* tool box for CALISTO with C-compiler, debugger, simulator, "Evaluation Module" (EVM), and Real-time operating system (RTOS) supports "Any Service Any Port" (ASAP) configurations for up to 240 channels of carrier class G.711 VoIP (voice over IP).

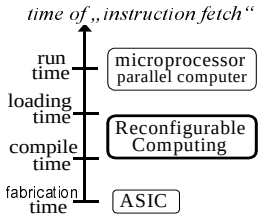

Fig. 10: "Instruction Fetch".

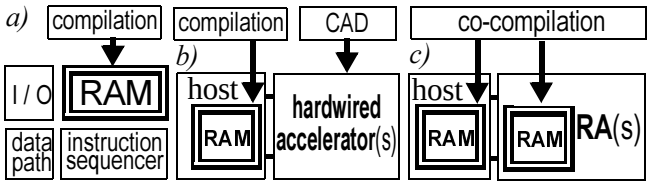

Fig. 11: Computing Platforms: a) "v. Neumann", b) current, c) emerging.

4.2 Design Space Explorers (DSEs)

Some development environments aim beyond compilation. DSEs (survey: [4]) and use interactive or automatic guidance systems or design assistants giving advice during the design flow to select one of many alternative solutions to meet design goals. We may distinguish Design Space Explorers (DSEs) to optimize a *design* or Platform Space Explorers (PSEs) to optimize a *programmable platform*. *Interactive DSEs* are *DPE (Design Planning Environment)* [59] with effect predictors and proposal generators, template-based *Clio* [60] (both for VLSI) and *DIA (Datapath-Intensive ASICs)* [61], targeting semi-custom ASIC behavioural level, generate a schematic, a data flow graph, or a layout from area, throughput, power, e.a. constraints specs.

A PSE serves to find an optimum RA or processor array (PA) platform for an application domain by optimizing array size, path width, processor's MIPS, number of ALUs and branch units, local SRAM size, data and instruction cache sizes, local bandwidth etc. from requirements like chip area, total computation, memory size, buffer size etc. Software or configware programming is finally not part of exploration, but may serve platform evaluation. All three being non-interactive, the *„DSE" [42] for RAW* [19] featuring an analytical model, *ICOS* (Intelligent Concurrent Object-oriented Synthesis) [62] featuring object-oriented fuzzy techniques, and *"DSE for Multimedia Processors"* [67] (DSEMMP) aim at automatic synthesis of a multiprocessor platform from system descriptions, performance constraints, and a cost bound and generate an architecture.

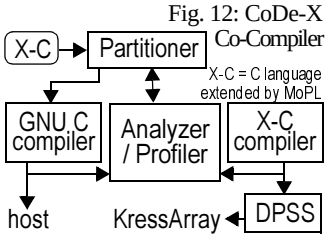

Fig. 12: CoDe-X

4.3 Data Transfer and Storage Exploration

Currently memory bandwidth and power dissipation are the most urgent optimization problems in DSE and PSE use as well as in mapping applications onto platforms. Due to rapidly spreading usage of portable systems recent research focuses on low power embedded processors as well as on low power RAs. The processor / memory communication bandwidth gap, which spans up to 2 orders of magnitude (see fig. 13), where new memory architectures like RAMbus or DDRAM and others bring only slight alleviation, can be even wider in data-intensive RA use, where caches do not help (fig. 10).

The more recently published Data Transfer and Storage Transformations (DTST) ([63] - [69]) offer a methodology for memory and communication power savings, and, loop transformations [51] [70] [54] etc. for power savings [71] [72] and speed-up - by working on data smaller local memory ([73] - [75]) instead of distant larger memory. Such DTSTs are a platform issue capable to extend the power of PSEs. A general architecture supporting such data locality strategies has been implemented already a decade earlier: the smart memory interface of the MoM reconfigurable architectures ([76] - [80] et al.), based on the generic address generator (GAG) general sequencer concept ([12] - [82] et al.), at that time also used for a flexible and storage scheme optimization methodology [63] for concurrent multiple memory banks (for illustration see fig. 4). It has been shown ([63] and earlier), that by using a 2-dimensional memory organization this methodology provides a rich supply of generic DTST transforms as well as their excellent visualization.

Local optimization usually leads to performance-degrading runtime solutions of access conflicts with estimated cost overhead of 10 - 100% (in power) for hardware and around 35% (in clock rate) for software [73] [74]. Also for global exploration the use of conflict-directed ordering (CDO) [75] as an extension of force-directed scheduling (FDS) [83] has been proposed [84]. Instead of a signal access flow graph (SAFG) [75] a multi-dimensional conflict graph (MD-CG) is used for a generalized CDO (G-CDO) algorithm for data transfer and storage exploration (DTSE) system [85] [86].

The KressArray Xplorer also yields solutions to the memory bandwidth problem [63] and low power problems by supporting mixed rDPU types, so that both, data sequencers and rDPUs dedicated to the application can be mapped onto the same KressArray what is illustrated by the example in fig. 4. These Xplorer capabilities provide a straight-forward approach to support architectural implementations of the Xputer soft machine paradigm.

4.4 Compiler / PSE symbiosis

Since to map an application onto a coarse grain RA may take only minutes, retargettable mappers or compilers may be also used for platform exploration. By profiling the results of the same application or benchmark on different platforms may be compared. Such a compiler / PSE symbiosis like in *Xplorer* provides direct verification and yields more realistic and more precise results than explorers using abstract models for estimation and gives better support for manual tuning.

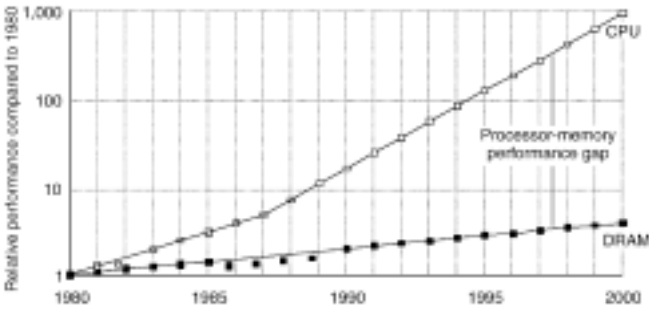

Fig. 13: Processor / memory performance gap (from Dave Patterson [87]).

The KressArray Xplorer, an interactive PSE framework [13] [43] has been implemented around a modified DPSS mapper [7]. This universal design space exploration environment supports both, optimum architecture selection (e. g. domain-specific) and application development onto it and includes several tools: *architecture editor* (to edit communication resources and annealing parameters), *mapping editor* (to change I/O port type, freeze locations of edge port, cell or cell group etc.), *instruction mapper* to edit and define the operator repertoire, *architecture suggestion generator* [88], *HDL generator* for cell simulation, *retargettable cell layout generator* (planned, similar to [89]), *power estimator* (planned [90], using methods from [92]). A cycle through an exploration loop usually takes only minutes, so that a number of alternative architectures may be checked in a reasonable time. By mapping the application onto it verification is provided directly. The Xplorer also supports optimization solutions to the memory bandwidth problem and the power dissipation problem (see section 4.3).

SoC System on a Chip
cSoC configurable SoC
Soap Chip System on a programmable Chip
ASPP application-specific programmable product

Fig. 14: Acronyms.

4.5 Parallel Computing vs. Reconfigurable

RISC core IP cells are available so small, that 32 (soon 64 or more) of them would fit onto a single chip to form a massively parallel computing system. But this is not a general remedy for the parallel computing crisis [93], indicated by rapidly shrinking supercomputing conferences. For many application areas process level parallelism yields only poor speed-up improvement per processor added. Amdahls law explains just one of several reasons of inefficient resource utilization. A dominating problem is the instruction-driven late binding of communication paths (fig. 10), which often leads to massive communication switching overhead at run-time. R&D in the past has largely ignored, that the so-called “von Neumann” paradigm is not a communication paradigm. However, some methods from parallel computing and parallelizing compiler R&D scenes may be adapted to be used for lower level parallelism on RA platforms (compare § “Co-Compilation“).

5. A New Business Model (Conclusions)

Deep submicron allows SoC implementation - not just subsystems, and the silicon IP business reduces entry barriers for newcomers and turns infrastructures of existing players into liability [94] [95]. Already in the early days of reconfigurability the business model has changed several times with the programming model. The PAL (1st wave) with write-once RAM has supported customization *after manufacture*. The FPGA (2nd wave) supports multiple reconfiguration *during* development The cSoC (3rd wave) permits multiple reconfiguration *after* development. We may distinguish (also see fig. 14) following classes of cSoC chip: high density FPL from catalogue (Soap Chip), configurable System on a Chip (cSoC), and, special SoC with FPL IP core (no acronym).

But so far we have not yet learnt the lessons taught by the history of silicon application synthesis, which distinguishes three phases [49] [96]: hardware design (fig. 8 a), microcontroller usage (fig. b), and FPL / RA usage (fig. c). The first shift has switched the

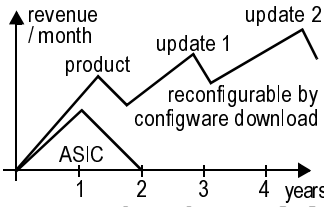

Fig. 15: accelerator longevity [94].

company	architecture	business model	market
Adaptive Silicon	not disclosed	sell cores	embedded DSP
Chameleon Systems	32 bit array	sell chips	networking
Malleable	not disclosed	bought by PCM Sierra	voice over IP
MorphICs	not disclosed	sell solutions	cellular wireless
PACT	not disclosed	sell cores	DSP & networking
Silicon Spice	not disclosed	bought by Broadcom	networking
Systolix	bit-serial systolic	sell cores	signal conditioning
Triscend	programmable SoC	sell chips	embedded systems

Fig. 16: Start-ups offering embedded reconfigurable array solutions [94].

business model from structural synthesis by net-list-based CAD (fixed algorithms, no machine paradigm) to RAM-based procedural synthesis by compilation, based on a machine paradigm, which drastically reduces the design space by guidance - the secret of success of the software industry. Note: RAM-based means flexibility and fast turn-around and shifts product definition from hardware vendor to customer's site. But the 3rd phase (resources have become variable), RAM-based structural accelerator synthesis (fig. 11 b) still uses phase 1 methods (CAD). It is time to switch to real compilation techniques, based on a soft machine paradigm. But the R&D scenes still ignore, that we now have a dichotomy of RAM-based programming: procedural versus structural, integrating two worlds of computing.

Exploding design cost and shrinking product life cycles of ASICs create a demand on RA usage for product longevity. Performance is only one part of the story. The time has come fully exploit their flexibility to support turn-around times of minutes instead of months for real time in-system debugging, profiling, verification, tuning, field-maintenance, and field-upgrades. A new "soft machine" paradigm and language framework is available for novel compilation techniques to cope with the new market structures transferring synthesis from vendor to customer.

Nevertheless, reconfigurable platforms and their applications are heading from niche to main-stream, bridging the gap between ASICs and micro-processors (fig. 16). Many system-level integrated future products without reconfigurability will not be competitive. Better architectures by RA usage, rather than technology progress, will be the key to keep up the current innovation speed beyond the limits of silicon. It is time to revisit past decade R&D results to derive commercial solutions: at least one promising approach is available. It's time to overcome the design crisis by switching to compilation techniques. It is time for you to get involved. Theory and backgrounds are ready for creation of a dichotomy of computing science for curricular innovations urgently needed.

6. Literature

1. R. Hartenstein, H. Grünbacher (Editors): The Roadmap to Reconfigurable computing - Proc. FPL2000, Aug. 27-30, 2000; LNCS, Springer-Verlag 2000
2. A. DeHon: Reconfigurable Architectures for General Purpose Computing; report no. AITR 1586, MIT AI Lab, 1996
3. R. Hartenstein: The Microprocessor is no more General Purpose (invited paper), Proc. ISIS'97, Austin, Texas, USA, Oct. 8-10, 1997.
4. R. Hartenstein (embedded tutorial): A Decade of Research on Reconfigurable Architectures - a Visionary Retrospective; DATE 2001, Munich, March 2001
5. A. Marshall et al.: A Reconfigurable Arithmetic Array for Multimedia Applications; Proc. ACM/SIGDA FPGA'99, Monterey, Feb. 21-23, 1999
6. D. Cherepacha and D. Lewis: A Datapath Oriented Architecture for FPGAs; Proc. FPGA'94, Monterey, CA, USA, February 1994.
7. R. Kress et al.: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; ASP-DAC'95, Chiba, Japan, Aug. 29 - Sept. 1, 1995
8. H. Reinig: A Scalable Architecture for Custom Computing; Ph.D. Thesis, Univ. of Kaiserslautern, Germany, July 1999.
9. R. Hartenstein, A. Hirschbiel, M. Weber: MoM - a partly custom-design architecture compared to standard hardware; IEEE CompEuro 1989
10. R. Hartenstein et al.: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90, 30th Anniversary of the Computer Society of Japan, Tokyo, Japan, 1990.
11. R. Hartenstein et. al.: A Novel ASIC Design Approach Based on a New Machine Paradigm; IEEE J.SSC, Volume 26, No. 7, July 1991.
12. R. Hartenstein, A. Hirschbiel, K. Schmidt, M. Weber: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High-Performance-HW; Future Generation Computer Systems 7, 91/92, North Holland - invited reprint of [10]
13. U. Nageldinger et al.: KressArray Xplorer: A New CAD Environment to Optimize Reconfigurable Datapath Array Architectures; ASP-DAC, Yokohama, Japan, Jan. 25-28, 2000.
14. R. A. Bittner et al.: Colt: An Experiment in Wormhole Run-time Reconfiguration; SPIE Photonics East '96, Boston, MA, USA, Nov. 1996.
15. D. Gajski et al.: A second opinion on dataflow machines; Computer, Feb '82
16. K. Hwang: Advanced Computer Architecture; McGraw-Hill, 1993.
17. E. Mirsky, A. DeHon: MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources; Proc. IEEE FCCM'96, Napa, CA, USA, April 17-19, 1996.
18. J. Hauser and J. Wawrzynek: Garp: A MIPS Processor with a Reconfigurable Coprocessor; Proc. IEEE FCCM'97, Napa, April 16-18, 1997.
19. E. Waingold et al.: Baring it all to Software: RAW Machines; IEEE Computer, September 1997, pp. 86-93.
20. T. Miyamori and K. Olukotun: REMARC: Reconfigurable Multimedia Array Coprocessor; Proc. ACM/SIGDA FPGA'98, Monterey, Feb. 1998.
21. J. Becker et al.: Architecture and Application of a Dynamically Reconfigurable Hardware Array for Future Mobile Communication Systems; Proc. FCCM'00, Napa, CA, USA, April 17-19, 2000.
22. X.Tang, et al.: A Compiler Directed Approach to Hiding Configuration Loading Latency in Chameleon Reconfigurable Chips; in [1]
23. C. Ebeling et al.: „RaPiD: Reconfigurable Pipelined Datapath", in [24]
24. M. Glesner, R. Hartenstein (Editors): Proc. FPL'96, Darmstadt, Germany, Sept. 23-25, 1996, LNCS 1142, Springer Verlag 1996
25. S. C. Goldstein et al.: PipeRench: A Coprocessor for Streaming Multimedia Acceleration; Proc. ISCA'99, Atlanta, May 2-4, 1999
26. D. Chen and J. Rabaey: PADDI: Programmable arithmetic devices for digital signal processing; VLSI Signal Processing IV, IEEE Press 1990.
27. D. C. Chen, J. M. Rabaey: A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic-Specific High-Speed DSP Data Paths; IEEE J. Solid-State Circuits, Vol. 27, No. 12, Dec. 1992.
28. A. K. W. Yeung, J.M. Rabaey: A Reconfigurable Data-driven Multiprocessor Architecture for Rapid Prototyping of High Throughput DSP Algorithms; Proc. HICSS-26, Kauai, Hawaii, Jan. 1993.
29. J. Rabaey: Reconfigurable Computing: The Solution to Low Power Programmable DSP; Proc. ICASSP'97 Munich, Germany, April 1997.
30. D. Lewis: Personal Communication, April 2000.C. Ebeling et al.: Placement and Routing Tools for the Tryptich FPGA; IEEE Trans on VLSI Systems 3, No. 4, December 1995.
31. A. DeHon: Personal Communication, February 2000.
32. C. Ebeling: Personal Communication, March 2000.
33. A. Marshall: Personal Communication; February 2000.
34. B. Hutchings, B. Nelson: Using General-Purpose Programming Languages for FPGA Design; Proc. DAC 2000, Los Angeles, June 2000
35. M. W. Hall et al.: Maximizing Multiprocessor Performance with the SUIF Compiler; IEEE Computer, Dec. 1996
36. T. J. Callahan and J. Wawrzynek: Instruction-Level Parallelism for Reconfigurable Computing; in [37] pp. 248-257.
37. R. Hartenstein, A. Keevallik (Editors): Proc. FPL'98, Tallinn, Estonia, Aug. 31- Sept. 3, 1998, LNCS, Springer Verlag, 1998
38. J. Hauser: Personal Communication, March 2000.
39. R. Hartenstein (invited embedded tutorial): Coarse Grain Reconfigurable Architectures; ASP-DAC'01, Yokohama, Japan, Jan 30 - Feb. 2, 2001
40. R. Barua et al.: Maps: A Compiler-Managed Memory System for RAW Machines; Proc. ISCA'99, Atlanta, USA, June, 1999.
41. W. Lee et al.: Space-Time Scheduling of Instruction-Level Parallelism on a RAW Machine; Proc. ASPLOS'98, San Jose, Oct. 4-7, 1998.
42. C. A. Moritz et al.: Hot Pages: Software Caching for RAW Microprocessors; MIT, LCS-TM-599, Cambridge, MA, Aug. 1999.
43. U. Nageldinger: Design-Space Exploration for Coarse Grained Reconfigurable Architectures; Dissertation, Universitaet Kaiserslautern, June 2001
44. M. Budiu and S. C. Goldstein: Fast Compilation for Pipelined Reconfigurable Fabrics; Proc. FPGA'99, Monterey, Feb. 1999, pp. 135-143.
45. D. Chen at al.: An Integrated System for Rapid Prototyping of High Performance Data Paths; Proc. ASAP'92, Los Alamitos, Aug. 4-7, 1992
46. P. H. Hilfinger: A High-Level Language and Silicon Compiler for Digital Signal Processing; Proc. 1985 IEEE CICC., Portland, May 20-23, 1985.M. Potkonjak, J. Rabaey: A Scheduling and Resource Allocation Algorithm for Hierarchical Signal Flow Graphs; Proc. DAC'89, Las Vegas, June 1989
47. J. Noguera, R. Badia: A HW/SW Partitioning Algorithm for Dynamically Reconfigurable Architectures; Proc. DATE 2001

48. J. Noguera, R. Badia: Run-time HW/SW Codesign for Discrete Event Systems using Dynamically Reconfigurable Architectures; Proc. ISSS 2000 (Int'l Symp. System Synthesis), Madrid, Spain, Sept. 20 - 22, 2000
49. N. Tredennick: Technology and Business: Forces Driving Microprocessor Evolution; Proc. IEEE 83, 12 (Dec. 1995)
50. T. Molketin: Analyse, Transformation und Verteilung arithmetischer und logischer Ausdruecke; Diplomarbeit, Univ. Kaiserslautern, 1995
51. J. Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators; Ph. D. dissertation, Kaiserslautern University, 1997.
52. M. Weinhardt, W. Luk: Pipeline Vectorization for Reconfigurable Systems; Proc. IEEE FCCM, April 1999
53. M. Gokhale, J. Stone: NAPA C: Compiling for a hybrid RISC / FPGA architecture; Proc. IEEE FCCM April 1998
54. K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers; Ph.D. Thesis, University of Kaiserslautern 1994
55. J. Becker et al.: A General Approach in System Design Integrating Reconfigurable Accelerators; Proc. IEEE ISIS'96; Austin, TX, Oct. 9-11, 1996
56. M. Herz, et al.: A Novel Sequencer Hardware for Application Specific Computing; Proc. ASAP'97, Zurich, Switzerland, July 14-16, 1997
57. A. Ast, J. Becker, R. Hartenstein, R. Kress, H. Reinig, K. Schmidt: Data-procedural Languages for FPL-based Machines; in [58]
58. R. Hartenstein, M. Servit (Editors): Proc. FPL'94, Prague, Czech Republic, Sept. 7-10, 1994, LNCS, Springer Verlag, 1994
59. D. Knapp & al.: The ADAM Design Planning Engine, IEEE Trans CAD, July 1991
60. J. Lopez et al.: Design Assistance for CAD Frameworks; Proc. EURO-DAC'62, Hamburg, Germany, Sept. 7-10, 1992
61. L. Guerra et al.: A Methodology for Guided Behavioral Level Optimization; Proc. DAC'98, San Francisco, June 15-19, 1998
62. P.-A. Hsiung et al.: PSM: An Object-oriented Synthesis Approach to Multiprocessor Design; IEEE Trans VLSI Systems 4/1, March 1999
63. M. Herz: High Performance Memory Communication Architectures for Coarse-Grained Reconfigurable Computing Architectures; Ph. D. Dissertation, Universitaet Kaiserslautern, January 2001
64. F. Cathoor et al.: Custom Memory Management Methodology; Kluwer, 1998
65. N. D. Zervas et al.: Data-Reuse Exploration for Low-Power Realization of Multimedia Applications on Embedded Cores; PATMOS'99, Kos Island, Greece, Oct 1999
66. D. Soudris et al.: Data-Reuse and Parallel Embedded Architectures for Low-Power Rreal-Time Multimedia Applications; PATMOS'2000, see: [90]
67. J. Kin et al.: Power Efficient Media Processor Design Space Exploration; Proc. DAC'99, New Orleans, June 21-25, 1999
68. S. Kougia et al.: Analytical Exploration of Power Efficient Data-Reuse Transformations on Multimedia Applications; ICASSP'2001, Salt Lake City, May 2001
69. S. Wuytak et al.: Formalized Methodology for Data Reuse Exploration for Low Power Hierarchical Memory Mappings; IEEE Trans. on VLSI Systems, Dec. 1998
70. D. Kulkarni, M. Stumm: Loop and Data Transformations: A tutorial; CSRI-337, Computer Systems Research Institute, University of Toronto, June 1993
71. V. Tiwari et al.: Power Analysis of Embedded Software: A First Step Towards Software Power Minimization; IEEE Trans. on VLSI Systems, Dec. 1994
72. G. Sinevriotis et al.: Power Analysis of the ARM 7 Embedded Microprocessor; PATMOS'99, Kos Island, Greece, Oct 1999
73. A. Vandecapelle et al.: Global Multimedia Design Exploration using Accurate Memory organization Feedback; Proc. DAC 1999
74. T. Omnès et al: Dynamic Algorithms for Minimizing Memory Bandwidth in High throughput Telecom and Multimedia; in: Techniques de Parallelization Automatique, TSI, Éditions Hermès, 1999
75. S. Wuytack et al: Minimizing the required Memory Bandwidth in VLSI System Realizations; IEEE Trans. VLSI Systems, Dec. 1999
76. R.W. Hartenstein, A.G. Hirschbiel, M. Weber: A Flexible Architecture for Image Processing; Microprocessing and Microprogramming, vol 21, pp 65-72, 1987
77. R. Hartenstein, A. Hirschbiel, M. Weber: MOM - Map Oriented Machine; in: E. Chiricozzi, A. D'Amico: Parallel Processing and Applications, North-Holland, 1988
78. R.W. Hartenstein, A.G. Hirschbiel, M.Weber: MoM - a partly custom-design architecture compared to standard hardware; Proc. Compeuro 89, IEEE Press 1989
79. R. Hartenstein, A. Hirschbiel, K. Schmidt, M. Weber: A Novel ASIC Design Approach based on a New Machine Paradigm; Proc. ESSCIRC`90, Grenoble, France
80. R. W. Hartenstein, M. Riedmuller, K. Schmitt, M. Weber: A Novel Asic Design Approach Based on a New Machine Paradigm; IEEE J. SSC, July 1991
81. R. Hartenstein, A. Hirschbiel, M. Riedmüller, K. Schmidt, M.Weber: A High Performance Machine Paradigm Based on Auto-Sequencing Data Memory; HICSS-24, Hawaii Int. Conference on System Sciences, Koloa Hawaii, 1991
82. Reiner W. Hartenstein, Helmut Reinig: Novel Sequencer Hardware for High-Speed Signal Processing; Workshop on Design Methodologies for Microelectronics, Smolenice Castle, Slovakia, September 1995
83. P. Paulin et al.: Algorithms for High-Level Synthesis; IEEE Design & Test, Dec'89
84. F. Cathoor et al: Interactive Co-design of High Throughput Embedded Multimedia; DAC 2000
85. L. Nachtergaele et al.: Optimization of Memory Organization and Partitioning for Decreased Size and Power in Video and Image Processing Systems; Proc. IEEE Workshop on Memory Technology, Aug 1995
86. F. Cathoor et al.: Custom Memory Management Methodology - Exploration of Memory Organization f. Embedded Multimedia Systems; Kluwer 1998
87. D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, K. Yelick: A Case for Intelligent RAM; IEEE Micro, Mar. / Apr. 1997.
88. R. Hartenstein, M. Herz, T. Hoffmann, U. Nageldinger: Generation of Design Suggestions for Coarse-Grain Reconfigurable Architectures; in [1]
89. V. Moshnyaga, H. Yasuura: A Data-Path Modules Design from Algorithmic Representations; IFIP WG 10.5 Worksh. on Synthesis, Generation and Portability of Library Blocks for ASIC Design, Grenoble, France, Mar 1992
90. U. Nageldinger et al.: Design-Space Exploration of Low Power Coarse Grained Reconfigurable Datapath Array Architectures; in [91]
91. D. Soudris, P. Pirsch, E. Barke (Editors): Proc. PATMOS 2000; Göttingen, Germany Sept. 13 - 15, 2000; LNCS, Springer Verlag, 2000
92. L. Kruse et al.: Lower Bounds on the Power Consumption in Scheduled Data Flow Graphs with Resource Constraints; Proc. DATE, Mrch 2000.
93. R. Hartenstein (invited paper): High-Performance Computing: über Szenen und Krisen; GI/ITG Workshop on Custom Computing, Dagstuhl, June 1996
94. T. Kean (invited keynote): It's FPL, Jim - but not as we know it! Market Opportunities for the new Commercial Architectures; in [1]
95. R. Hartenstein (invited keynote): Reconfigurable Computing: a New Business Model - and its Impact on SoC Design; DSD'2001 Warzaw, Poland, Sept 4 - 6, 2001
96. T. Makimoto: The Rising Wave of Field-Programmability; in: [1]