Interner Bericht

Hardware / Software Codesign

Reiner W. Hartenstein,
Fachbereich für Informatik,
Universität Kaiserslautern

Nr. 246 / 94

Fachbereich Informatik

Universität Kaiserslautern · Postfach 3049 · D-67653 Kaiserslautern

Hardware / Software Codesign

Reiner W. Hartenstein,
Fachbereich für Informatik,
Universität Kaiserslautern

Nr. 246 / 94

Bau 12 / 455,
Postfach 3049
D - 67653 Kaiserslautern,
Germany

Telefon: (+49-631) 205 - 2606
Fax: (+49-631) 205 - 2640
e-mail: hartenst@rhrk.uni-kl.de

presented at FPL'93 -
3rd International Workshop
on
Field-Programmable Logic and Applications,
Oxford University, Oxford, U.K.
September 7-10, 1993

HARDWARE / SOFTWARE CO-DESIGN

R. Hartenstein

Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049, D-67653 Kaiserslautern, Germany
fax: (+49 631) 205-2640, e-mail: hartenst@rhrk.uni-kl.de

ABSTRACT

The paper gives some highlights on a new R&D area called Hardware/Software Co-Design. It tries to give a answers to several questions. What are the goals and unsolved problems? What are the hardware platforms? What are the relations between field-programmable hardware and this new area?

INTRODUCTION

A new research scene has been formed [6] [25] which has been indicated by recently held workshops on Hardware-Software Co-Design: Grassau, Bavaria (1991), Estes Park, Colorado (1992), Innsbruck, Tyrolia, Austria (1993), Cambridge, Mass (1993). With respect to topic areas covered these workshops substantially overlap with workshops on field-programmable media and related subjects such as FPL held 1991 and 1993 at Oxford, UK and 1992 in Vienna, FPGA held at Berkeley in 1992, as well as the workshop on novel machines using FPGAs held at Napa, CA in 1993. The question is, what are the goals of the Hardware-Software Co-Design group? What hardware platform is used by these people? How important are field-programmable media for these researchers? Are we facing a break-through?

Remedy for throughput bottlenecks in software

Let us illustrate such an application by the following scenario. A larger program is being implemented on workstation (host) equipped with an add-on universal accelerator board (UAB). Procedure X proves to be a throughput bottleneck, so that performance requirements

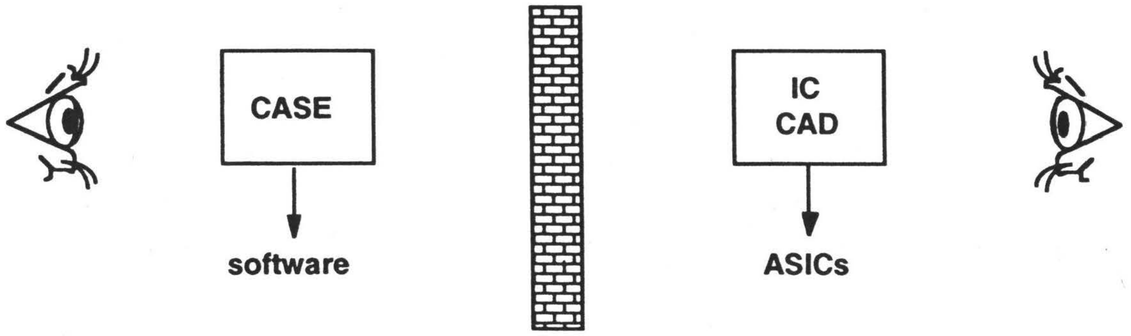

Figure 1 : Hardware and software have been more or less seen from different view points

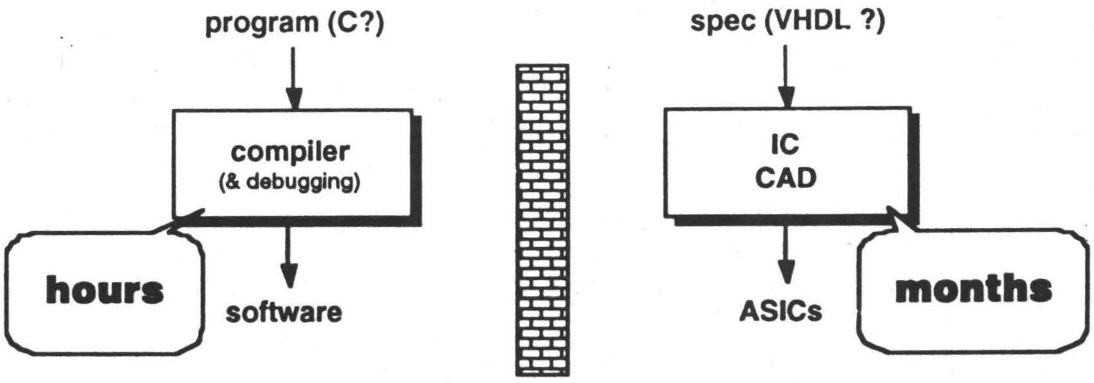

Figure 2 : A problem have been the extremely different turn-around times

cannot be met. The problem is solved by an accelerator for X compiled into the UAB. Also an hardware-software interface is compiled into the UAB and the host. Then the accelerated procedure X is running on the UAB being called by the program running on the host.

Technology Platforms and Tools: the Key

Hardware and software have been more or less seen from different view points (Figure 1). This makes hardware-software codesign very difficult. A major organizational hurdle has been the enormous difference of prototyping turn-around time between debugging a piece of software (e. g. an algorithm) and debugging hardware (Figure 2). The main reason is the delivery time in hardware prototype manufacturing (e. g. gate arrays), so that most designs cannot be tested earlier than several weeks after completion.

Meanwhile other hardware platforms are being commercialized. Several vendors offer field-programmable boards (FPCBs) with quasi standard host interfaces on board (and some application development software). Aptix Corporation, San José, CA, offers several FPCB™ types for ASIC prototyping with PC/AT bus interface, VME bus interface, 320 pin standard adapters, or, as a stand-alone programming module (SPM) (Stand-alone Programming Module). GO Giga Operations, Berkeley, CA, offers a dynamically reconfigurable accelerator board with ISA bus interface. National Semiconductor, Santa Clara, CA, offers a FPCB type board for ASIC prototypes with PC/AT bus interface. Quickturn Systems, Mountain View, CA, offers ASIC emulators with interfaces for Sun sparc, IBM RS6000, and hp9000/700.

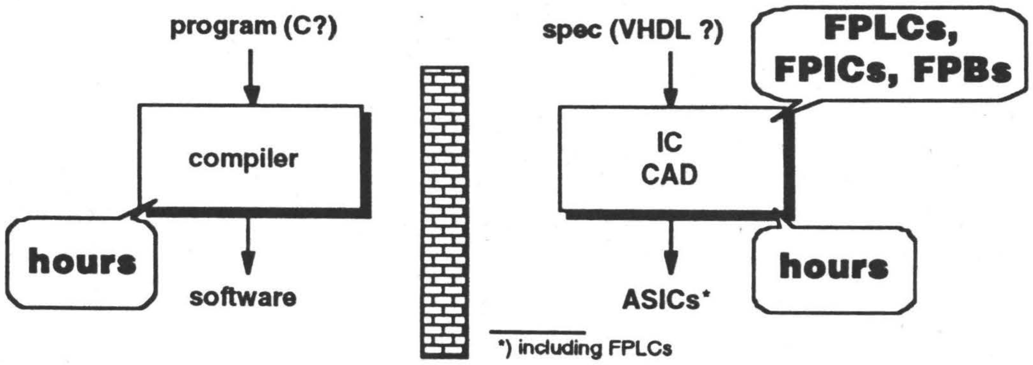

Figure 3 : Field-programmable technology brings turn-around times closer together

Experimental results indicate, that in using such hardware platforms, along with tools driven by recent progress in logic synthesis and formal methods, the turn-around time on the hardware side may be reduced to days or even only hours (Figure 3). This is a major motivation of the current research rush in hardware-software codesign. Figure 5 shows the design flow in using field-programmable media in more detail. A common view point for hardware design and software implementation has become desirable also with respect to design process organization (Figure 4). It is time for a marriage between CASE and CAD.

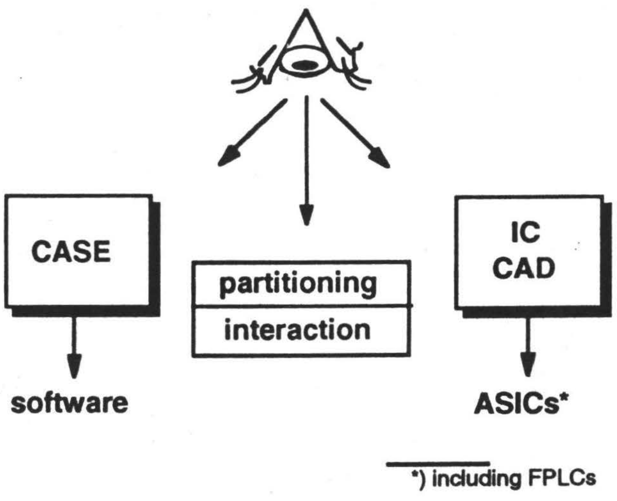

Figure 4 : Common view point now feasible

Research Goals and unsolved problems

Hardware-Software Co-Design as a new research area has a number of problems to be solved (Figure 6). What source language should be used? What abstraction level(s) should be used for the source? Are separate languages for hardware and software the right way, or, would it be better to have a common specification? What method should be used for partitioning and interface synthesis? Would automation be feasible? Should be the hardware / software interface be treated as a subsystem class of its own (Figure 7)?

Motivations which influence research goals are the increasing diversity and complexity of applications which employ embedded systems, as well as the decreasing design time available for many products. So it is necessary to develop software for a processor, the design of which has not yet been completed. It is desirable to have performance figures available before the prototype has been manufactured. For hardware / software trade-offs comparison of alternative design solutions would be useful. It should be possible to identify bottlenecks very early, and, to swap in place a module of reasonable granularity.

Development Systems built

A number of experimental hardware-software co-design environments have been implemented or proposed which could be classified as hardware-oriented (Figure 8 a), software-oriented (Figure 8 b), or high-level-(specification-)oriented (Figure 8 c), as well as based special paradigms, like on communicating finite state machines (FSMs, Figure 9 a), or, on a data-procedural (non-von-Neumann [16] [17]) machine paradigm (Figure 9 b). The Princeton solution [30] is a (CASE-like [8] [31]) high-level specification approach (Figure 8c) using a co-specification language which can be

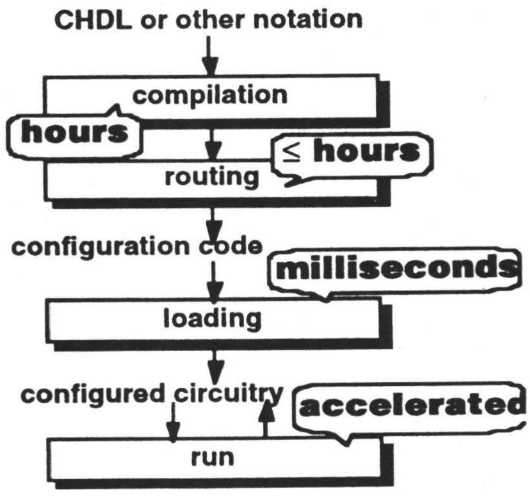

Figure 5 : Turn-around times

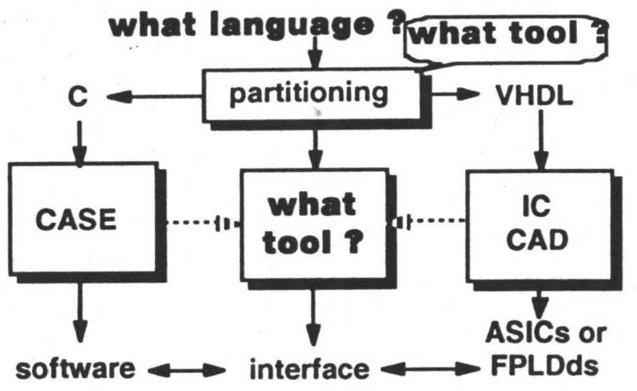

Figure 6 : Unsolved problems and research goals

translated into hardware (BESTMAC-C language) and software using C++ (Figure 7). The CODES system of Buchenrieder's system [7] is more a CASHE framework for concurrent engineering mainly using tools available commercially like Statemate [15] and others. The software-oriented system by Ernst and Henkel [12], based on a standard architecture, uses a source language C* (a superset of C). Gupta and De Micheli [13] [14] (Figure 10) move operations from software over to hardware, based on a standard architecture(Figure 11). Parkeret al.[24] use a mathematical method for automatic partitioning. The Berkeley approach [20] is based on communicating sequential machines (Figure 9 a). Despain [18] combines high-level synthesis (pipeline synthesis for a special mircoprocessor) with compiler techniques. Acosta [1] uses a dataflow specification as a source. Wenban et al. [29] use as a source a concurrent procedural language with typed communication channels, which are mapped into a set of protocols implemented in hardware. For more systems see also [10] [21].

FPL-based novel machines.

We may distinguish three fundamentally different approaches to hardware-software codesign: universal systems based on single or multiple von Neumann processor(s), embedded systems using von Neumann hardware together with customized hardware, and, custom computers, i. e. reconfigurable machines which may be adapted to application requirements. Even an entire workshop having attracted about 120 people (the FCCM'93, held at Napa, California in April 1993) has been dedicated to custom computers. Field-programmable media are an essential technology platform for such custom computers. This section deals with customs computers and the reasons, why such computers are often much more efficient than universal computers.

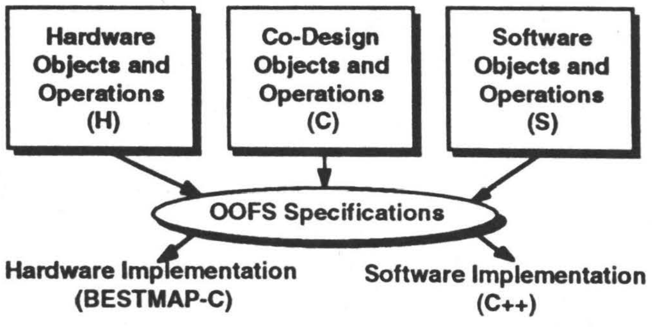
Hardware Objects and Operations (H)
Co-Design Objects and Operations (C)
Software Objects and Operations (S)
OOFS Specifications
Hardware Implementation (BESTMAP-C)
Software Implementation (C++)

Figure 7 : The Princeton Co-Design model

Von-Neumann-type *sequential programming* is based on sequential code, sequentially scanned by an instruction sequencer. The application-specific complexity is pushed into a large RAM. *Structural programming* however uses configuration code, loaded into hidden memory. field-programmable devices are configured to run like ASIC hardware. The underlying hardware is rapidly reconfigurable at any time and may be compacted and frozen by retargeting, so that later a real ASIC may be derived almost just by pushing a button.

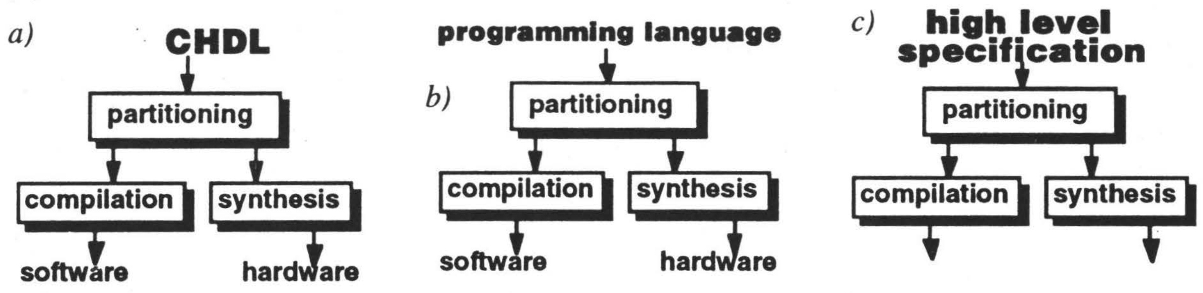

Figure 8 : Different approaches in research: a) hardware-, b) software-, c) high-level-oriented

FPLC-based processor architectures having been built recently are the SPACE machine, a network of CAL-based state machines [22], the DEC PAM, a smart memory approach [5], the Splash machine, a kind of field-programmable warp machine (similar to a programmable systolic array machine) [2] [3], and the MoM (Map-oriented machine), an Xputer architecture [4] [17] (based on a data-procedural machine paradigm [16]). By the way: CASHE stand for Computer-Aided Software-Hardware Engineering.
A paper by Wazlowski and others discusses compiler and architecture for a general-purpose metamorphic computing platform called PRISM-II [27] which improves the performance of many computation-intensive tasks by augmenting the functionality of the core processor with new instructions that match the characteristic of targeted applications. Also the Spyder system uses field-programmable gate arrays to implement a reconfigurable processor which achieves very high computation speed [19]

A project investigating the combination of a CM-2 machine and FPGAs from Xilinx [11] is a kind of hybrid between the universal systems solution and the embedded system approach using ASICs. A better solution would be to use instead of ASICs a kind of universal accelerator (based on a machine paradigm) being structurally programmable, is used. Why does this help? This problem will be subject of the next section

A data-procedural machine paradigm.

A major problem in hardware-software codesign is the requirement, that special skills needed for programming the interfaces between hardware and software. That is the reason, why this research area is dominated by people with a strong hardware background. We all know, that most software people use (with more or less awareness) a kind of virtual von Neumann

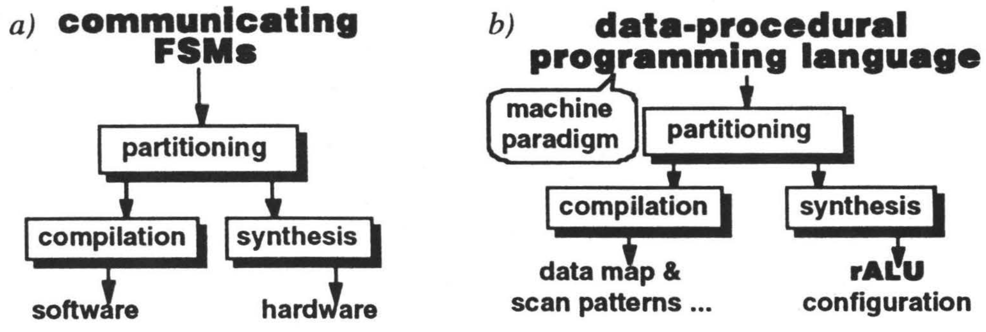

Figure 9 : Machine paradigm approaches: a) FSM-based, b) Xputer-based

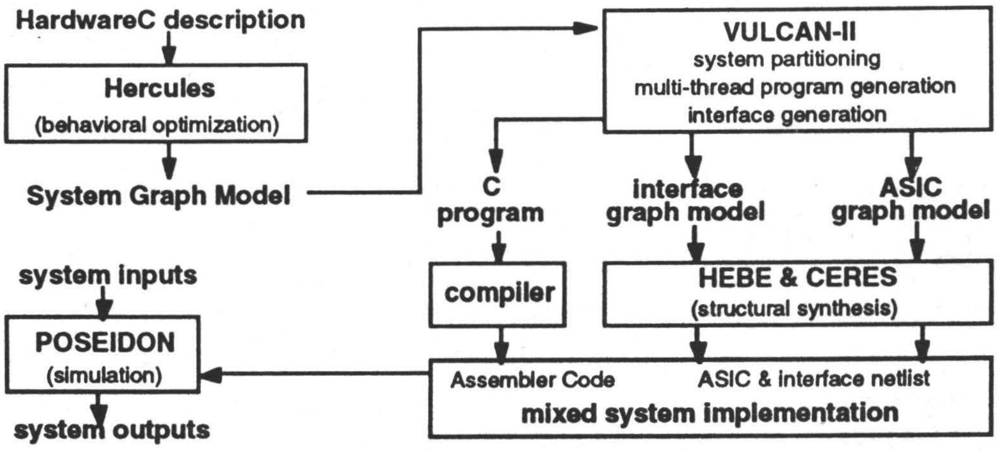

Figure 10 : The Stanford development system

machine as a model to express their concepts in an executable form. The problem is, how to reach the familiarity and universality of the von Neumann paradigm when using a custom computer.

A system based on a machine paradigm would be helpful to achieve such an acceptance by software people. The question is, how to achieve the structural programmability. A kind of soft ALU would be a way to achieve such a customizability. However, the application of a soft ALU on a von Neumann machine is not feasible because of very tight sequencer/ALU coupling. That's why we need another sequencing mechanism to allow highly flexible reconfigurable ALU (rALU). A solution to this problem is the use of a data counter instead of an instruction counter.

This leads to the procedural non-von-Neumann Xputer paradigm [16] with only very loose coupling between ALU and sequencing mechanism (Figure 13). The highly reconfigurable ALU of an Xputer (the rALU) allows very high fine granularity intra-ALU parallelism. Figure 14 illustrates a development system for Xputer applications. No interface has to be generated, since the hardware-software interface is a hardwired part of the machine platform.

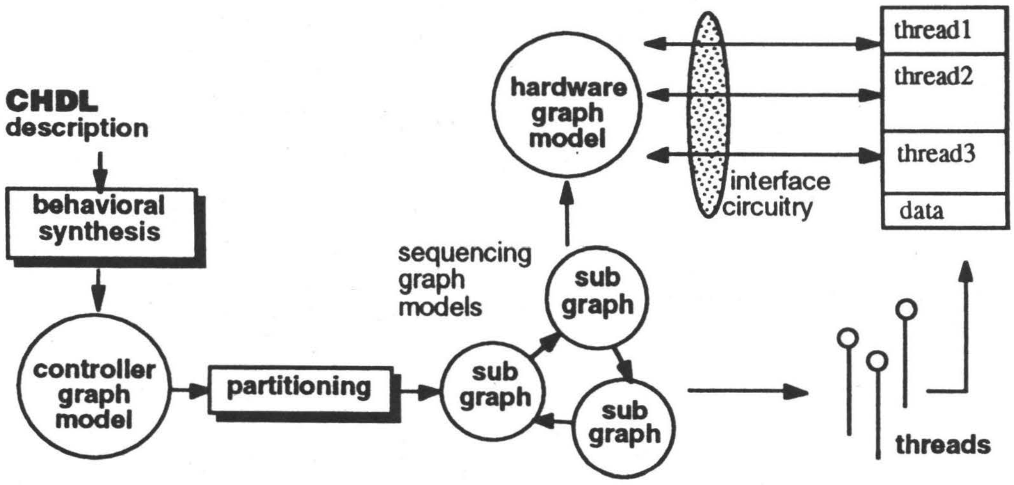

Figure 11 : The Stanford method: design flow

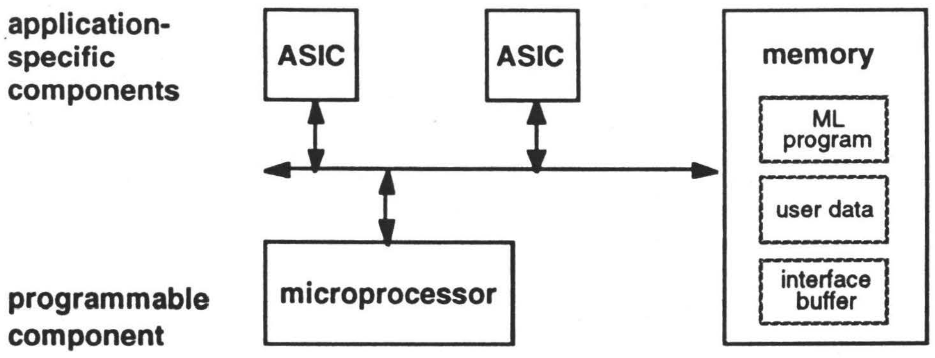

Figure 13 : The Stanford method: basic scheme of hardware platform

Figure 15 shows some acceleration factors obtained experimentally from a Xputer architecture. Figure 16 gives some overhead analysis results based on the examples listed in Figure 15. But there are more reasons for the high efficiency of data-procedural machines. The detailed explanation of all acceleration factors takes more time and space than available within this paper. The key to understand the high acceleration factors achieved is not to find out, why the Xputer is so good. The key issue rather is to find out, why the von Neumann machine is so inefficient.

Future directions

Currently the scene is confused between abstraction levels and too many languages, even too many language classes. The wide variety of approaches needs a shake-out, driven by improved understanding of the key issues. Also better environments are needed to test processor and compiler against common computational problems. Current methods of early performance estimation and cost measurement are too expensive and require too much effort to explore alternative solutions by moving modules across the hardware software borderline. We need much more area-efficient field-programmable technology and much better tools for their personalization as well as better compiler.

Conclusions

It has been shown, that synthesis combined with modified CASE methods using application-specific components as well as field-programmable components provide a promising extension of high-level synthesis approaches to realize complex system designs without corresponding increase in the magnitude of the synthesis task. CASHE is a promising approach to achieve

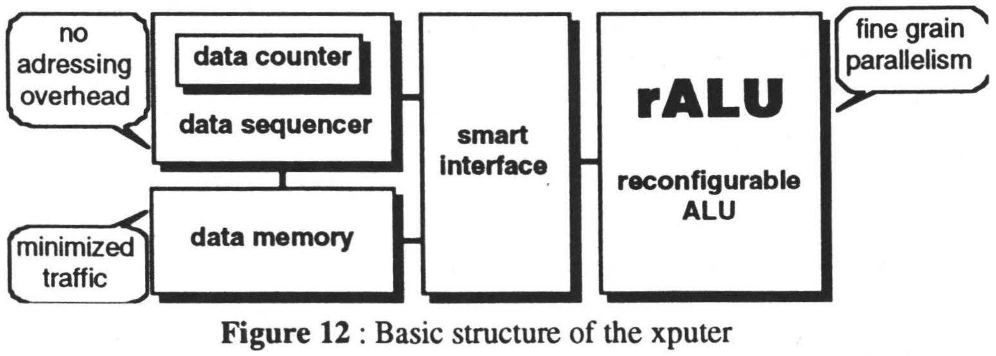

Figure 12 : Basic structure of the xputer

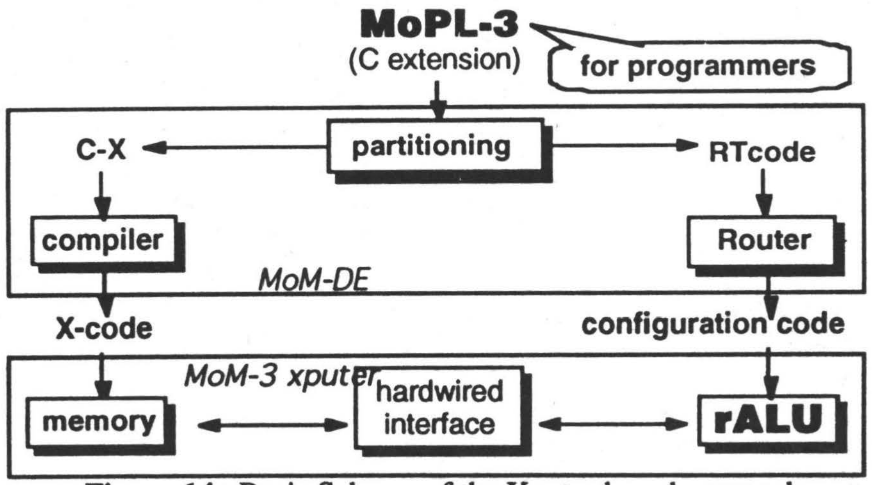

Figure 14 : Basic Scheme of the Xputer-based approach

more design automation, much better hardware/software trade-offs (Figure 17), and faster design turn-around. However, CASHE is in its infancy and needs much more basic research. Data-procedural machine principles have been introduced, where codesign is embedded into a machine paradigm such, that procedural source languages with high acceptance by programmers can be combined with the efficiency of highly flexible structural programming.

REFERENCES

[1] R. D. Acosta: Use of Dataflow Specifications for Software / Hardware Codesign; handout from ACM / IEEE 1st Int'l Workshop on Hardware-Software Co-Design, Estes Park, Colo, 1992

[2] J. M. **Arnold et** al.: Splash-2; ACM Symp. on Parallel Algorithms and Architectures, June 1992

[3] J. M. Arnold: The Splash software environment; Proc. IEEE Workshop for Custom Computing Machines (FCCM'93); Napa, CA, April 1993

[4] A. Ast, R. W. Hartenstein, H. Reinig, K. Schmidt, M. Weber: A new machine paradigm as a consequence of DSP evolution; in: (ed.: M. A. Bayoumi) VLSI Design Methodologies for DSP Architectures and Applications; Kluwer, 1993

[5] P. Bertin, D. Roncin, J. Vuillemin: Programmable Active Memories: a Performance Assessment; ACM Int'l Workshop on Field-Programmable Gate-Arrays; Berkeley, CA, Febr. 1992

Algorithm	68000 16 MHz millisec	MoM-2 10 MHz millisec	acceleration factor
CMOS design rule check	91,330.20	39.030	2340
digital filter	9126.40	29.440	310
Lee Routing:			
seek S	42.50	0.063	680
wavefront	70.00	0.38	186
backtracking	23.25	0.23	186

Figure 15 : Xputer acceleration factors obtained experimentally

Algorithm	data manipulation	addressing	control
CMOS design rule check	3 %	93 %	4 %
digital filter	14 %	58 %	28 %
Lee Routing:			
seek S	7 %	74 %	19 %
wavefront	3 %	92 %	9 %
backtracking	8 %	67 %	25 %

Figure 16 : Overhead analysis for Figure 15

[6] K. Buchenrieder: Codesign and Concurrent Engineering; Computer, Jan. 1993

[7] K. Buchenrieder, C. Veith: CODES: A Practical Concurrent Design Environment; handout from ACM / IEEE 1st Int'l Workshop on Hardware-Software Co-Design, Estes Park, Colo, 1992

[8] J. Cameron: JSP and JSD: The Jackson Approach to Software Development; IEEE-CS Press, 1989

[9] R. Camposano, W. Wolf (editors): High Level Synthesis; Kluwer, Boston, 1990

[10] T. A. Cook, E. A. Harcourt, T. K. Miller III, P. D. Franzon: Towards Unified Specification for Architecture and Compiler Design; handout from ACM / IEEE 1st Int'l Workshop on Hardware-Software Co-Design, Estes Park, Colo, 1992

[11] S. A. Cuccaro, C. F. Reese: The CM-2X: A Hybrid CM-2 / Xilinx Prototype; Proc. IEEE Workshop for Custom Computing Machines (FCCM'93); Napa, CA, April 1993

[12] R. Ernst, J. Henkel: Hardware-Software Co-Design of Embedded Controllers based on hardware Extraction; handout from ACM / IEEE 1st Int'l Workshop on Hardware-Software Co-Design, Estes Park, Colo, 1992

[13] R. Gupta, C. Coelho, D. De Micheli: Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components; Proc. DAC 1992, IEEE 1992

[14] R. Gupta, C. Coelho, D. De Micheli: Program Implemen-tation Schemes for Hard-ware-Software Systems; hand-out from ACM / IEEE 1st Int'l Workshop on Hard-ware-Software Co-Design, Estes Park, Colo, 1992

[15] D. Harel: Statecharts: a visual formalism for complex systems; Science and Computer Programming **8**,3, 1987

[16] R. W. Hartenstein, A. G. Hirschbiel, K. Lemmert, K. Schmidt, M. Weber: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; Int'l Conf. on Information Technology, Tokyo, Japan, Oct. 1990.

[17] R. W. Hartenstein, A. G. Hirschbiel, K. Schmidt, M. Weber: A Novel ASIC Design Approach based on a New Machine Paradigm; 16th ESSCIRC, Grenoble, Sept. 1, 1990.

[18] I. Huang, A. Despain: High-Level Synthesis of Pipelined Instruction Set Processors and Back End Compilers; Proc. DAC 1992, IEEE CS Press 1992

[19] C. Iseli, E. Sanchez: Spyder: A Reconfigurable VLIW Processor using FPGAs; Proc. IEEE Workshop for Custom Computing Machines; Napa, CA, April 1993

[20] A. Kavalade, E. A. Lee: Hardware / Software Co-Design Using Ptolemy - A Case Study; handout ACM / IEEE 1st Int'l Worksh. on Hardw./Software Co-Design, Estes Park, Colo, 1992

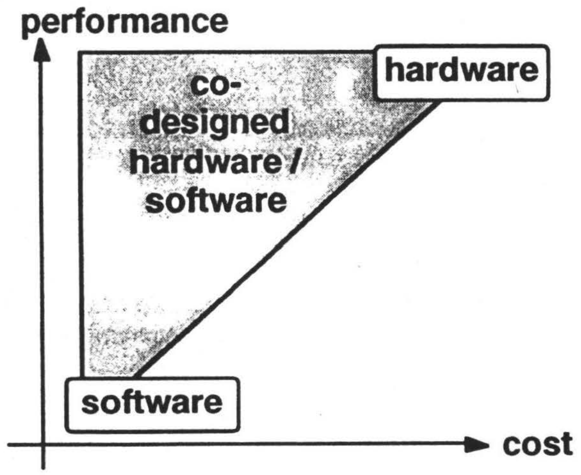

Figure 17 : Trade-off benefit from co-design

[21] S. Kumar, J. H. Aylor, B. W. Johnson, W. A. Wulf: A Framework for Hardware/Software Codesign; handout from ACM / IEEE 1st Int'l Workshop on Hardware-Software Co-Design, Estes Park, Colo, 1992

[22] G. Milne: Realising massively concurrent systems on the SPACE machine; Proc. IEEE Workshop for Custom Computing Machines (FCCM'93); Napa, CA, April 1993

[23] N. N. (Plessey): Quickgate (Product Overview); Plessey, Swindon, U.K., May, 1990.

[24] A. C. Parker et al.: Unified system construction (usc); in: [9]

[25] P. A. Subrahmanyam: Hardware-Software Codesign; Computer, Jan. 1993

[26] G. K. Wallace: The JPEG Still Picture Compression Standard; CACM **34**,4, Apr.1991.

[27] M. Wazlowksi et al.: PRISM-II Compiler and Architecture; Proc. IEEE Workshop for Custom Computing Machines; Napa, CA, April 1993

[28] M. Weber: An Application Development Method for Xputers; Ph. D. dissertation, Kaiserslautern University, 1990.

[29] A. S. Wenban, J. W. O'Leary, G. M. Brown: Codesign of Communication Protocols; handout from ACM / IEEE 1st Int'l Workshop on Hardware-Software Co-Design, Estes Park, Colo, 1992

[30] N. Woo, W. Wolf, A. Dunlop: Compilation of a single specification into hardware and software; handout from ACM / IEEE 1st Int'l Workshop on Hardware-Software Co-Design, Estes Park, Colo, 1992

[31] P. Zave, W. Schell: Salient features of an executable specification language and its environment; IEEE Trans. on Software Engineering, SE-**12**,2, Febr. 1986