Reiner W. Hartenstein, Juergen Becker, Michael Herz, Ulrich Nageldinger: An Innovative Platform for Embedded System Design ; Proceedings of the Workshop Zielarchitekturen Eingebetteter Systeme, ZES'97, in conjunction with the Fachtagung Architekturen von Rechensystemen ARCS'97, Rostock, Germany, September 11, 1997

An Innovative Reconfigurable Platform for Embedded System Design

Reiner W. Hartenstein, Jürgen Becker, Michael Herz,
Ulrich Nageldinger

University of Kaiserslautern
Erwin-Schrödinger-Straße, D-67663 Kaiserslautern, Germany
Fax: ++49 631 205 2640, email: abakus@informatik.uni-kl.de
URL: http://xputers.informatik.uni-kl.de

Abstract

This paper presents a new general purpose reconfigurable technology platform for embedded system design called Kress ALU Array III (KrAA-III), which is a novel field programmable ALU array (FPAA). FPAAs can be structurally programmed to resemble a complex operator used by an application, providing high computation power by massive pipelining. While embedded accelerator design with traditional FPGAs is very similar to sophisticated ASIC-design due to the bit-level of FPGAs, the KrAA-III features 32 bit wide datapaths, enabling rapid design using high-level languages. In this paper, the underlying concepts and the hardware of the KrAA-III are presented.

1. Introduction

Meanwhile field-programmable devices have reached a market volume of almost four Billion US Dollars (worldwide). Hardware has become soft and structural programming is gaining momentum towards traditional procedural programming. Also some field-programmable boards are commercialized (e. g. [Cha94],[CST95]). An important application are embedded reconfigurable accelerators (e.g. [KG93],[HB96a]). Reconfigurable architectures start to compete with procedurally programmable processors. Why did all the MPC and HPC scenes ignore this platform of efficient parallelism all the time?

Why are “Real Time Systems“, “Multimedia Hardware“, “DSP“ und “Custom Computing Machines“ strongly disjoint scenes? They all have the same goal: high performance for a reasonable price. Also a new IEEE Technical Committee called “Engineering of Computer-Based Systems (ECBS)“, tries to tackle the chaos [HB96b]. But what is missing is an interdisciplinary model as powerful as von Neumann, which helps to link these still isolated scenes.

Emanating from the technology of *field-programmable logic* (FPL) the new paradigm of *structural programming* has evolved. So we now have two programming paradigms: program-

ming in time and programming in space. The R&D area of Custom Computing Machines (CCMs), such as FPGA-based CCMs (F-CCMs), merges both programming paradigms to an integrated methodology to speed up-performance. We have obtained a dual software-only implementation: procedural software running on the host, together with structural software running on the reconfigurable accelerators.

A closer look onto F-CCMs shows that former field programmable devices (FPGAs) were originally designed to implement control and glue logic, because they use only narrow datapaths (mostly one bit). Wider register or memory resources on chip are rarely implemented by these devices. The logic is mostly dedicated to implement boolean formulas and not arithmetic functions. In contrast to these fine-grained FPGAs we face different requirements for implementing a computing element. Although some FPGA manufacturers developed more feasible devices (e.g. XC6k [Xil96], FLEX10k [Alt95], AT6000 [Atm94]) they meet such requirements only insufficiently. A moderately wide datapath (e.g. 16, 32 bits) is necessary. The functional units have to implement also more comprehensive arithmetic functions. Very short configuration times are required. Reconfigurations should also be possible during runtime. Further context switches are necessary to change efficiently between several tasks during runtime.

Because of these requirements such a computing element goes more in the direction of processing elements (PEs) like used in systolic array implementation, in contrast to FPGAs, which are similar to an ASIC approach. Therefore we obtain the new class of Field Programmable ALU Arrays (FPAAs) which can be seen between FPGAs and PEs [HB96d] [HB96e]. Examples of FPAAs are the Programmable Arithmetic Devices For DIgital Signal Processing (PADDI, [YR92] [YR93]) or the reconfigurable Datapath Architecture (rDPA, [Kre96]).

This paper presents a novel architecture of FPAAs called Kress ALU Array III (KrAA-III). It is shown, how the KrAA-III can be used to build a novel architecture called Xputer, which serves as a universal platform for embedded accelerators. This architecture provides acceleration speed-up due to parallel processing as well as the capability to be programmed by high-level languages due to the coarse grained structure of the KrAA-III. The paper is structured as follows: In the next section, the concepts and hardware of the KrAA-III is introduced. In section 3, the programming of the KrAA-III is demonstrated. Next, the Xputer structure is introduced. Finally in section 4, an example for an embedded system using the KrAA-III is shown.

2. The Architecture of the Novel Kress ALU Array

To support highly computation-intensive applications, structurally programmable platforms are needed providing word level parallelism instead of the bit level parallelism like FPGAs. It has shown that on applications with large data elements, fine-grained devices pay too much area for interconnect than coarser-grained devices (for an extensive study on area utilization refer to [DeH96]). For area efficiency this new platform should be suitable for full custom design, like known from ASAPs (Application Specific Array Processors) operating in SIMD (Single Instruction Multiple Data) mode. But ASAPs are not structurally programmable and support only problems with completely regular data dependencies (Figure 1).

Therefore FPAAs are as dense as ASAPs but each ’PE’ is programmed individually. Every array element of the KrAA-III consists of a reconfigurable datapath unit (rDPU). (Kress called his first approach reconfigurable datapath array, rDPA, [Kre96].) In contrast to the original approach, two full-duplex local interconnects between the rDPUs are available (figure 2). Furthermore the global interconnect between all rDPUs is constructed hierarchically and enables parallel data transfers. For speed-up of task switches a context switch mechanism is developed, that allows reconfiguration of unused layers during runtime. Additionally, some features have been adopted from the original rDPA [Kre96]:

	characteristics or feature	systolic or wavefront array	Kress Array
1	massive pipelining: parallel, opposite directions, crossing	yes	
2	circuit layout	structured VLSI design (full custom) capable, wiring by abutment, i. e. highly area-efficient	
3	interconnect media	hardwired	programmable at 3 levels
4	total interconnect pattern	regular mesh only	no restrictions: individual globally, locally & in PE
5	ensemble of PE functions	uniform only	locally individual
6	pipeline shape	linear only	free form: meandering, feed back, forks, joins & others
7	data stream synthesis	projection (linear only)	optimizing scheduler
8	array synthesis, (or, (re-) configuration, respectively)	projection (linear only)	simulated annealing optimizer
9	applications executable	problems with regular data dependencies only	any — no restrictions
10	generate running data streams	no systematics published	smart interface available

Figure 1. Kress Array (III) - hyper generalization of the systolic array: it combines the area-efficiency and very high throughput of the systolic array with the universality of the so-called "von Neumann" machine.

- transparent scalability:
  Several KrAA-III devices can be connected to form an array of arbitrary size.
- dynamic partial reconfiguration:
  Parts of the array can be reconfigured, while others stay configured. This can also be done during runtime.
- 32-bit datapath width
- the rDPU can serve simultaneously as routing and arithmetic operator
- pipe-like asynchronous inter-rDPU communication
- smart interface [HHS91] for data scheduling (data streams entering and leaving the FPAA):
  All source data, needed for an application, is typically read via a global databus and results are written after computations. This is to reach the inner rDPUs. To improve the data scheduling internal registers of the rDPUs can be allocated to store data.

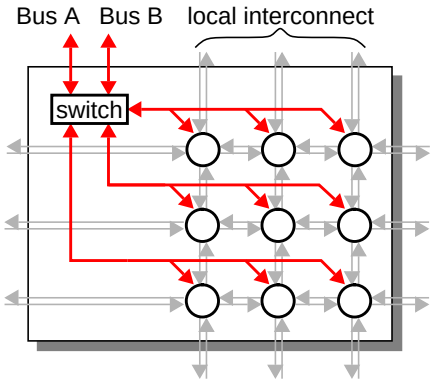

Figure 2. KrAA-III chip structure illustration with hierarchic global routing scheme

The Novel Hierarchical Global Routing Network. Basically data can be fed via all local and global busses. Within an application specific solution data may be fed into the array via the local connections at the border of the array. If the KrAA-III is used within a universal system there are different mappings possible and source data may be needed within the array at rDPUs not reachable via the local interconnect. For universal purposes data is only routed via the global databus. Local connections are used to pass intermediate results between rDPUs. To avoid that the global bus is getting a bottleneck a novel hierarchical global routing network is introduced.

Figure 2 shows the structure of the global bus network. The inner global busses connected to the rDPUs are developed once to save valuable design space. As illustrated, they are further connected to a switch, which can either isolate inner busses or connect them to any other bus. If inner busses are isolated, rDPUs connected to this bus can communicate independently from the remaining array. This is also the case if two inner busses are connected to each other. Furthermore there are two parallel busses to the outside of the KrAA-III chip, which can be connected to each inner bus. This enables parallel data transfers from outside the array, where more hierarchy levels and more parallel busses are possible.

As a result of the global routing network structure pictured in figure 2 a maximum of three independent global data transfers are possible inside one KrAA-III chip. Furthermore two parallel data transfers on the global bus from outside can be performed concurrently.

The Improved Local Routing Scheme. In addition to the hierarchical global routing network the KrAA-III has local routing capabilities between nearest neighbors. In contrast to the rDPA, which has only one unidirectional connection between two rDPUs, the KrAA-III has two 32 bit full-duplex connections (figure 3f). This connections are reconfigurable, i.e. the direction is programmable (figure 3c,d). To extend local routing capabilities the rDPu can serve as routing element (figure 3b) also during performing an arithmetic operation (figure 3a). Local routing saves a lot of global bus cycles and is independent from global bus routing.

As all datapaths are 32-bit, the local connections over chip-boundaries are serialized in order to reduce the number of chip-pins.

The New Structure of the Reconfigurable Datapath Unit. The original DPU designed by Kress [Kre96] has improved to an rDPU with higher functionality (figure 4), convering all arithmetic operations of the high-level language C. Additionally, routing operations can be performed in parallel.

The configuration memory consists of four independent layers. Each layer holds a complete configuration data set for the rDPU. Reconfigurations can be performed very fast by a context switch mechanism. That means that all rDPUs of an array change the actual configuration memory layers. Therefore also the register file for data in the rDPU has to be implemented with four layers. Otherwise all data had to be stored before a context switch is performed. To gain a further speed up out of the independent configuration layers the configuration control and the channels for configuration data are independent from each other. Thus reconfigurations of the three idle layers can be performed in parallel to the calculations on the active layer. Therefore the configuration time is no longer a penalty for the accelerator. If configurations and calculations have to be performed sequentially the configuration time had to be added to the runtime and therefore some applications would not benefit from such an accelerator. In the case of the KrAA-III a related hardware/software co-design framework (e.g. CoDe-X, [Bec97][HB97a]) has to regard only reconfiguration times that are longer than the execution

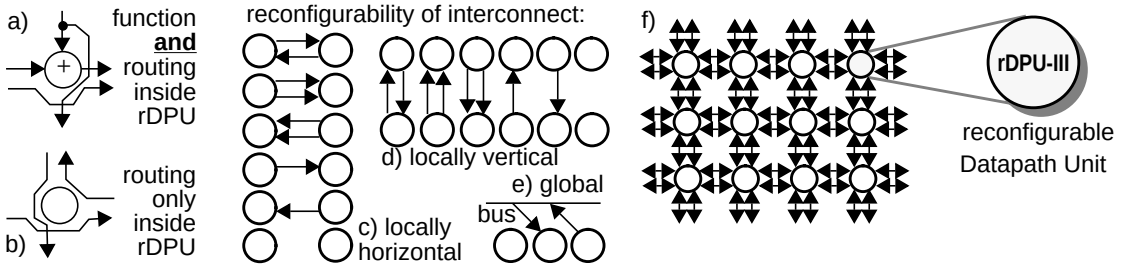

Figure 3. KrAA-III local routability.

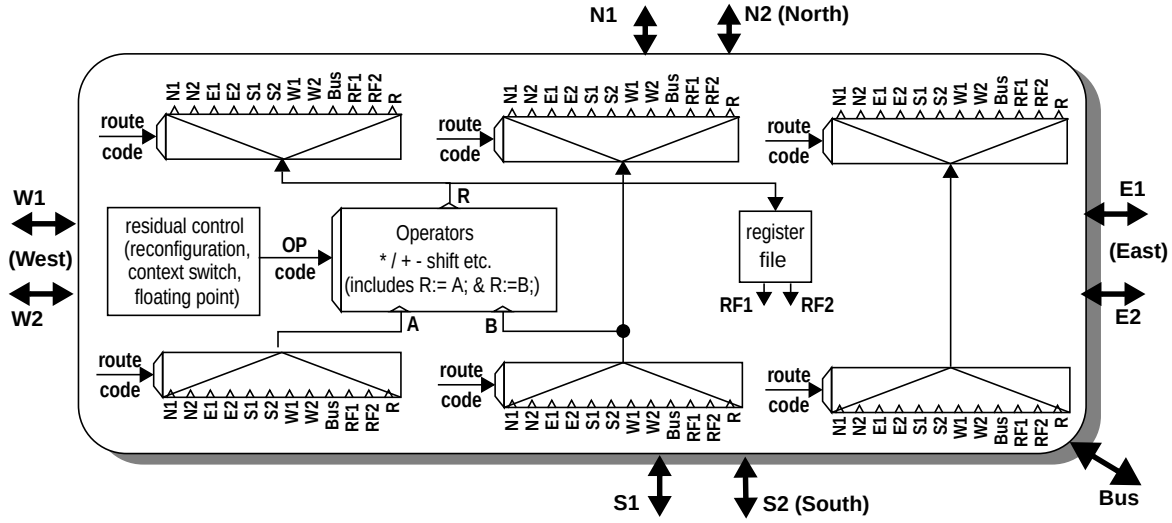

Figure 4. The rDPU-III architecture.

times of predecessor tasks. Furthermore tasks executed several times (e.g. nested loops or recurrent functions) can stay in configuration layers permanently.

3. The Programming of the Kress ALU Array

In this section, the programming of the KrAA-III is demonstrated. In contrast to the synthesis of dedicated hardware, this approach offers a software only accelerator implementation [Bec97]. That means no longer a hardware designer is needed for implementation. This is performed in using the DPSS (Datapath Synthesis System, [Ha95a], [Ha95b]) as a synthesis tool for the KrAA-III, which is integrated transparently into the CoDe-X programming framework [Bec97]. The DPSS benefits from the course grained structure of the KrAA-III. While a synthesis system for a FPGA struggles to implement high-level arithmetic operations on a bit-level FPGA structure, the DPSS can map complete arithmetic expressions described in the C subset ALE-X (Arithmetic and Logic Expression Language for Xputers, [Ha95a]) directly onto the rDPUs.

As a computation example, we assume an image processing application, performing an edge detection algorithm, which is implemented by two dimensional FIR (Finite-Impulse-Response, [GW77]) filters with appropriate parameters (figure 5). The basic structure of a 3 by 3 FIR Filter is shown in figure 6a. The edge detection is performed by applying first a 3 by 3

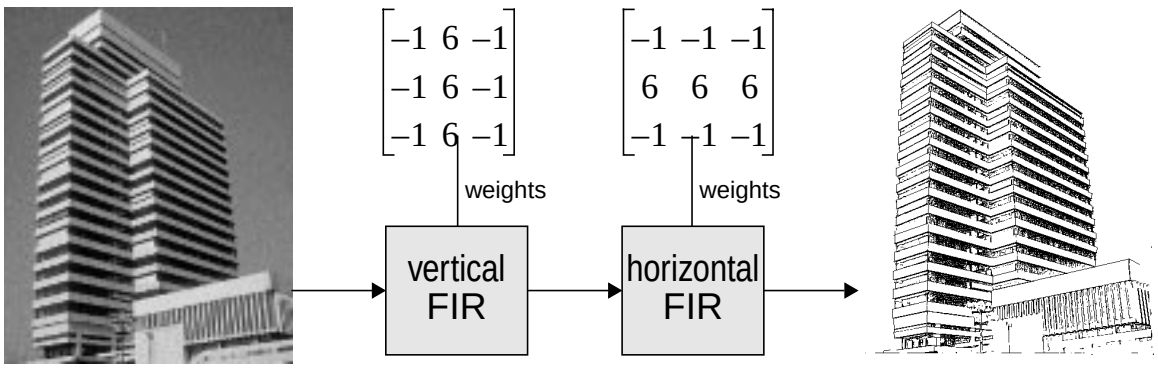

Figure 5. Edge detection example with appropiate weights

a)

w_{00}	w_{10}	w_{20}
w_{01}	w_{11}	w_{21}
w_{02}	w_{12}	w_{22}

$$\begin{aligned}\bar{w}_{11new} = {} & w_{22}k_{22} + w_{12}k_{12} + w_{02}k_{02} \\ & + w_{21}k_{21} + w_{11}k_{11} + w_{01}k_{01} \\ & + w_{20}k_{20} + w_{10}k_{10} + w_{00}k_{00}\end{aligned}$$

b)

```
wnew[x][y] = w[x+1][y+1] * k22
           + w[x]  [y+1] * k12
           + w[x-1][y+1] * k02
           + w[x+1][y]   * k21
           + w[x]  [y]   * k11
           + w[x-1][y]   * k01
           + w[x+1][y-1] * k20
           + w[x]  [y-1] * k10
           + w[x-1][y-1] * k00;
```

Figure 6. Equation of 3 by 3 FIR filter (a) and description in C subset ALE-X (b)

FIR filter which detects the vertical edges and then a second 3 by 3 FIR filter to detect the horizontal edges. The two FIR distinguish by the selection of the weights. For each pixel of the image the FIR filter calculates a new value as the weighted sum of all pixel-values in a 3 by 3 area with the current pixel as center. In figure 6a, the w_{xy} are the pixels while the k_{xy} denote the according weights. To perform the desired effect of edge detection, the weights k_{xy} have to be chosen in an adequate way (for example weights see figure 5).

To implement the filter using the KrAA-III, the C subset ALE-X ([Ha95a]) description can be used (figure 6b), allowing a rapid design of the accelerator‘s data manipulation part. The description is processed by the DPSS, which maps the application onto the KrAA-III. The DPSS generates structural code for the KrAA-III, where placement and routing is carried out by a fast Simulated Annealing algorithm [Kre96]. To organize the data streams to and from the KrAA-III, Data Scheduling [Ha95a] is used. Resource scheduling is not necessary, as this has already been done in the placement step of the DPSS. The resulting KrAA-III structure of the filter example is shown in figure 7. The calculation of the new value of the center pixel is executed in a mixture between a pipeline and a tree structure.

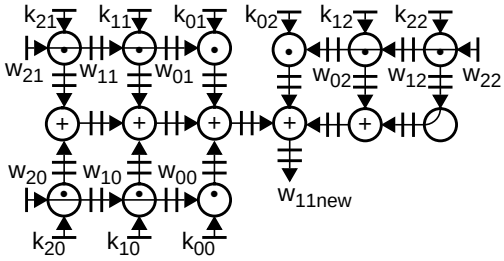

Figure 7. 3x3 FIR filter mapped into KrAA-III

Figure 7 shows only one possible mapping of the FIR filter example. The sum of the products is performed by several two- and three-input adders. The weights are configured into registers nearby the multiplication operators. The multiplications are performed by multiplication- and multiplication-with-data-routing-operators (see figure 3a). This operator performs a multiplication with the input data and simultaneously routes the data input to a specified data output. Furthermore one routing operator (figure 3b) is used in the example. Alternatively this routing operator could be replaced by a global bus routing cycle, but an only routing rDPU is already free for an arithmetic operation.

4. A Real World Computing Application: Automotive Collision Detection

This section introduces an embedded accelerator example for real-time automotive collision detection (published in [HB97b]). The system uses a camera with a high-dynamic-range CMOS image sensors (HDRC, [GHS95]) to scan the street in front of a car. The view from the camera is then used for steering to avoid collisions with other objects on the street. The accelerator is used for image preprocessing, namely edge detection in the raw image, supporting the task of object detection. The FIR filter implementation in section 3 configured into the KrAA-III is sufficient. The preprocessed image is further processed by a host computer, which controls the steering decisions. The structure overview of a hardware to perform such a collision

detection can be seen in figure 8. The way operations are performed is as follows. A data sequencer computes the data dependent address sequences (e.g. to follow the street borders) for the HDRC and the data memory. On the sequenced data the KrAA-III performs the edge detection function.

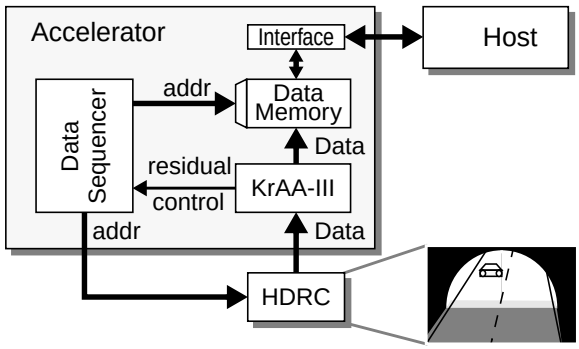

Figure 8. Automotive collision detection using the KrAA-III integrated into an embedded accelerator

5. Conclusions

The novel KrAA-III ALU array architecture has been presented. Several new mechanisms (improved rDPU, context switch mechanism and hierarchically routing structure) for speed up of configurable computing are introduced. Especially the context switch mechanism recommends the KrAA-III architecture, because of the ability to hold tasks needed several times in a configuration layer. Furthermore reconfigurations of unused configuration layers can be done in parallel to calculations of the active layer. It has been shown, that applications for the KrAA-III can be synthesized out of high-level specifications due to its coarse-grained structure. The synthesis tool DPSS, integrated into the CoDe-X programming framework, is capable of mapping algorithms into effective structures on the KrAA-III. The use of the KrAA-III and the Xputer paradigm has been demonstrated by an FIR filter implementation. Based thereon, the embedded system example of automotive collision detection is an application area. In contrast to the design of a dedicated hardware, the KrAA-III approach provides more flexibility and universality because of reconfigurability, as well as a faster implementation because of programmability with C.

6. References

[Bec97] J.Becker: A Partitioning Compiler for Computers with Xputer-based Accelerators; Ph. D. Thesis, Kaiserslautern University, 1997.

[HB97b] R. Hartenstein, J. Becker, M. Herz, U. Nageldinger: An Embedded Accelerator for Real World Computing; accepted for publication in Proceedings of IFIP International Conference on Very Large Scale Integration, VLSI‘97, to be held in Gramado, Brazil, August 26-29, 1997.

[HB97a] R. Hartenstein, J. Becker: A Two-level Co-Design Framework for data-driven Xputer-based Accelerators; published in Proc. of 30th Annual Hawaii Int’l Conf. on System Sciences (HICSS-30), Jan. 7-10, Wailea, Maui, Hawaii, 1997.

[HB96e] R. Hartenstein, J. Becker, M. Herz, U. Nageldinger: A General Approach in System Design Integrating Reconfigurable Accelerators; Proc. IEEE Int’l Conf. on Innovative Systems in Silicon; Austin, TX, Oct. 1996.

[HB96d] R. W. Hartenstein, J. Becker, et al.: Custom Computing Machines vs. Hardware/Software Co-Design: from a globalized point of view; 6th International Workshop On Field Programmable Logic And Applications, FPL’96, Darmstadt, Germany, September 23-25, 1996, Lecture Notes in Computer Science, Springer Verlag, 1996.

[DeH96] Andre DeHon: Reconfigurable Architectures for General-Purpose Computing; Technical Report 1586, MIT Artificial Intelligence Laboratory, September, 1996.

[HB96c] Reiner W. Hartenstein, Jürgen Becker, et al.: An Embedded Accelerator for Real Time Image Processing; 8th EUROMICRO Workshop on Real Time Systems, L’Aquila, Italy, June 1996.

[HB96b] R. Hartenstein, J. Becker, et al.: Two-Level Hardware/Software Partitioning Using CoDe-X; Int'l IEEE Symp. on Engineering of Computer Based Systems (ECBS), Friedrichshafen, Germany, March 1996

[HB96a] R. Hartenstein, J. Becker, et al.: High-Performance Computing Using a Reconfigurable Accelerator; CPE Journal, Special Issue of Concurrency: Practice and Experience, John Wiley & Sons Ltd., 1996

[Xil96] N.N.: The Programmable Logic Data Book; Xilinx, 1996.

[Kre96] R. Kress: A fast reconfigurable ALU for Xputers; Ph. D. Thesis, Kaiserslautern University, 1996.

[Ha95b] R. W. Hartenstein, et al.: A Scalable, Parallel, and Reconfigurable Datapath Architecture; Sixth International Symposium on IC Technology, Systems & Applications, ISIC'95, Singapore, Sept. 6-8, 1995.

[Ha95a] R. W. Hartenstein, et al.: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; Asia and South Pacific Design Automation Conference, ASP-DAC'95, Chiba, Japan, Aug. 29 - Sept. 1, 1995.

[GHS95] H.G. Graf, B. Höfflinger, U. Seger, A. Siggelkow: Elektronisch sehen; Elektronik, Vol. 3/95, Franzis-Verlag, 1995.

[Alt95] N.N.:FLEX10k Embedded Programmable Logic Family, Data Sheet; Altera Corp., July 1995.

[CST95] S. Casselman, J. Schewel, M. Thornburg: H.O.T. (Hardware Object Technology) Programming Tutorial; Release 1, Virtual Computer Corporation, January 1995

[Atm94] N.N.: Configurable Logic Design and Application Book 1994•1995; Atmel Corporation, 1994.

[ABH94] A. Ast, J. Becker, R. W. Hartenstein, et al.: Data-procedural Languages for FPL-based Machines; FPL'94, Prague, Sept. 7-10, 1994, Lecture Notes in Computer Science, Springer, 1994.

[Cha94] P. Chan: A Field-Programmable Prototyping Board: XC4000 BORG User Guide; UCSRC-CRL-94-18, Apr 1994

[KG93] A. Koch, U. Golze (Technical University Braunschweig, Germany): A Universal Co-Processor for Workstations; in: W. R. Moore, W. Luk (eds.): More FPGAs; Abbington EE&CS Books, Oxford, UK 1993 (selection from Proc. Int'l Symposium on field-programmable Logic and Applications, Oxford, UK, Sept. 1993)

[YR93] Alfred K.W. Yeung, Jan M. Rabaey: A Reconfigurable Data-Driven Multiprocessor Architecture for Rapid Prototyping of High Throughput DSP Algorithms; HICSS-26, Vol. 1, IEEE Computer Society Press, 1993.

[YR92] Alfred K.W. Yeung, Jan M. Rabaey: A Data-Driven Architecture for Rapid Prototyping of High Throughput DSP Algorithms; IEEE VLSI Signal Processing Workshop, Oct. 1992.

[HHS91] 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, p. 181-198, (Elsevier Scientific).

[GW77] R.C. Gonzalez, P. Wintz: Digital Image Processing; Addison-Wesley Publishing Company, USA, 1977.