Memory Addressing Organization for Stream-based Reconfigurable Computing

(invited paper)

Michael Herz,
Agilent Technologies,
D-71034 Böblingen, Germany
herz@computer.org

Reiner Hartenstein,
Dept. of Computer Science (Informatik)
University of Kaiserslautern,
D-67653 Kaiserslautern, Germany
reiner@hartenstein.de

Miguel Miranda, Erik Brockmeyer, IMEC,
Francky Catthoor, IMEC, and,
part-time professor at K.U.Leuven
B-3001 Leuven-Heverlee, Belgium,
{miranda}@imec.be

Abstract. *The paper gives a survey on methods to cope with the memory bandwidth problem, especially in the context of data-stream-based computing systems, like reconfigurable (and hardwired) datapath arrays, introduced in [1] and elsewhere.*

1. Introduction

Being more area-efficient than reconfigurable logic platforms like FPGAs [2], coarse grain Reconfigurable Arrays (cgRA) [1] are preferred hardware platforms for Reconfigurable Computing (RC) applications like multi media and next generation cellular wireless [2] [3] [4]. In a similar way as already known from systolic arrays, such RAs are running in a data-transport-triggered execution mode, with many data streams flowing into and out of the RA. The more data stream traffic is meeting the RA the more the efficiency and performance depends on the RA / memory communication architecture. This paper deals with the requirements which are much different from traditional parallel computing systems.

2. Smart Address Generators

Latency of data accesses often determines the computation throughput (a study e. g. in [5]). Address generators have the potential to reduce computation time significantly. In a grid-based design rule check [6] implemented on the MoM-1 (see section "The GAG") a speed-up of more than 2000 has been achieved, compared to a VAX-11/750, where a dedicated address generator contributed a factor of about 10 by avoiding memory cycles otherwise needed for address computation overhead. This section gives an overview on address generators and focuses on: simultaneous address generation and data manipulation, reduction of the time needed for address generation, and, reducing or avoiding memory cycles by software to hardware migration in address computation.

The Structured Memory Access (SMA) Machine. [5] is a von Neumann machine reducing address calculation overhead. by special hardware to generate data structure references. SMA consists of a computational processor (CP) and an independent memory access processor (MAP). For computations, the CP buffers an entire instruction block from where it may loop over one or more blocks of instructions.

MAP supports absolute accesses always accessing the same data and accesses relative to some index value. Index register functions use a hardware stack which tracks the active indices of inner loops, and all data structure references are made by using a subset of the index values. Tables in the SMA processor store the base address of each data structure and other information needed for an entire address from indices. These tables are loaded once at the beginning of a program execution. MAP has an internal Operand-Instruction Buffer (OIB) and immediately generates operand addresses and places them on the read or write queue of the outstanding memory requests. Index instructions are especially supported by sophisticated hardware features.

The GAG (generic address generator. The MoM-1 (Map-oriented Machine 1) was originally called PISA machine (pixel-oriented system for image analysis) [7] [8] [9], an image processing machine with 2-D memory organization, primarily designed to implement a pattern matching approach to grid-based design rule check (DRC [10] [11]), which achieved a speed-up of more than 2000 compared to a VAX-11/750. This acceleration stems from avoiding address calculation overhead (90% of the VAX CPU time) and fully parallelized pattern matching by a dynamically reconfigurable PLA (DPLA, designed. at Kaiserslautern and manufactured via the German multi university E.I.S. project).

The MoM-1 address generator called MCU (move control unit) is an application specific generic address generator (later called GAG) is configured before execution time and needs no memory cycles at runtime. It generates independent x- and y-addresses to move a 3-by-3 word "scan window" with a limited set of operations needed for a video scan pattern over a 2-D pixel file. A local 2-D shift register file avoids multiple reading of the same date due to overlapping scan window locations.

Application-specific Address Generator (ASAG). Grant, Denyer and Finlay [12] use a synthesis method for application specific address generators (ASAG), dedicated for applications, where the sequence of storage and retrieval of particular blocks of data is strongly patterned. The needed address patterns are generated by a dedicated counter, or via circuit transformations (bit shuffling and combinatorial logic operations) applied to a counter output. Details of the synthesis method and an example can be found in [12].

GAG of MoM-2. The Map-oriented Machine 2 [13] uses a 2-level address generator [14] based on the extremely flexible slider method [10], supporting vari-size scan windows (MoM-2 supports only up to 4-by-4 size scan windows) and their navigation through 2-D memory space. The data sequencer's address generator is configured by parameters and needs no memory cycles at run time. It consists of a Task Manger, a JumpGenerator [13], and a Single Step Control Unit (SSCU), which operate in a pipelined fashion.

The JumpGenerator which computes x- and y-address of the scan window base location [13] on the basis of a few buffered parameters, where each 1-D address generator holds three identical steppers (fig. 2 a). Each address stepper is programmed by an initial base value (delivered by the base stepper: e. g. B_0) which is iteratively modified by a step width (e. g. ΔA) until limit value (delivered by the limit stepper: e. g.

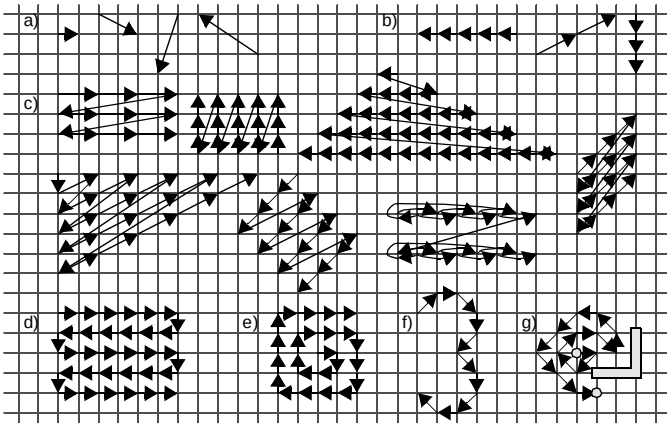


Fig. 1: MoM-3 Scan Pattern Examples: (a) Single Steps, (b) Linear Scans, (c) Video Scans, (d) Zigzag Scan, (e) Spiral Scan, (f) Curve Following (Data Dependent), and (g) Lee Routing (Data Dependent).

L0) is reached (see scan path example fig. 2 b). The slider model for a wide variety of sophisticated scan path patterns (a few examples in fig. 1) is implemented by a nested loop, with sliding base and limit values (outer loop) initializing repetitive address stepper calls generating (inner loop) a sequence of different linear scan paths (illustrated by example in fig. 3). For complex access pattern the control of the Task Manager is required. For instance an access pattern like the JPEG zig-zag scan requires frequent exchange of the parameters.

The SSCU controls the updates of a scan window cache on the basis of its "handle" base address provided by the JumpGenerator. The updates of the Scan Cache between two steps may affect the complete cache or may be optimized with a 2-D shift register structure to avoid multiple reading of the same data words, e. g. in case of overlapping window locations.

AGU. [15] is an application specific address generation unit (AGU) for video signal processor (VSP), a special DSP, which implements a 2-level address generation with window based memory access. However, compared to MoM-2 it is less flexible due to missing full slider method. AGU runs independent from execution unit (EU). Three AGUs running in parallel calculate the address for external image memory. The AGU provides seventeen addressing modes: a 2-dimensional raster scan mode, a block scan mode is suited for spatial filtering, eight variations of a neighborhood search mode, a 2-dimensional indirect access mode to address external image memory, a FFT mode (1-D and 2-D), and, an affine transformation mode are available.

The GAG of the MoM-3. with Handle Position Generator (HPG) and Memory Address Generator (MAG).uses an improved simplified version [16] of the same address generation method as the MoM-2, but it supports multiple (up to 7) access patterns at the same time. The data sequencer of the MoM-3 consists of an Instruction Sequencer (IS, [17])

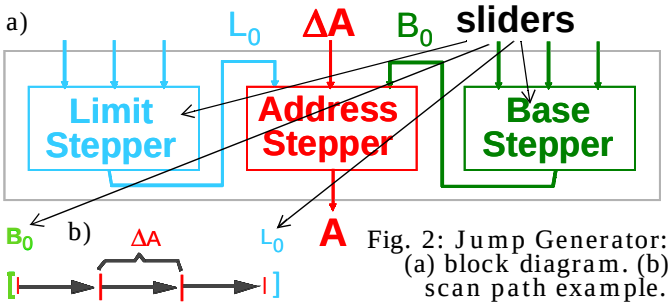
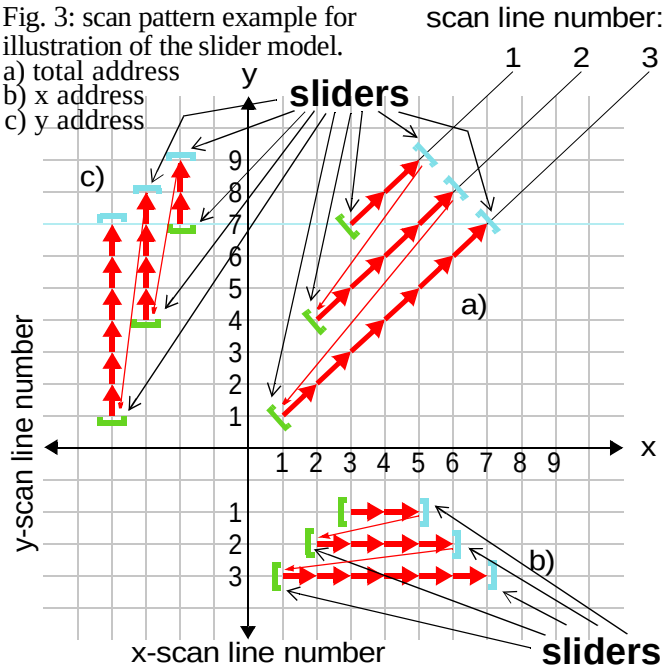


Fig. 2: Jump Generator: (a) block diagram. (b) scan path example.

Fig. 3: scan pattern example for illustration of the slider model.
a) total address
b) x address
c) y address



and up to 7 Generic Address Generators (GAG, [18] [19]). Figure 5 shows an implementation example of synthesizable GAG memory communication fabrics supporting 4 read data streams and 4 write data streams, mapped onto the same reconfigurable KressArray, together with the application.

The 2-D memory organization of the MoM architecture, together with the 2D scan windows feature supports memory parallelism by multiple memory banks (fig. 4 c), as well as storage scheme optimization by scan path pattern transformations like shearing, stretching, rotation and others, and, their combinations (rotation example shown in fig. 4 a-b).

The Texas Instruments TMS320C54x DSP. is a fixed-point DSPs in the TMS320 family [20]. It provides hardware support to avoid too much address computation overhead including two dedicated address generators, eight auxiliary registers and two auxiliary register arithmetic units.

It combines an advanced Harvard architecture (with one program memory bus, three data memory buses, and four address buses), with a CPU with a powerful set of arithmetic, logic, and single cycle bit-manipulation operations and with application specific hardware logic, on-chip memory, on-chip peripherals, and a highly specialized instruction set, a 40-bit ALU, including a 40-bit barrel shifter and two independent 40-bit accumulators. Separate program and data memory allow a high degree of parallelism, like, for example, three reads and one write in a single cycle. Additionally a 17-bit x 17-bit parallel multiplier is coupled to a 40-bit dedicated adder to perform a non pipelined single-cycle multiply/accumulate (MAC) operation. Furthermore the CPU includes a compare, select, store unit (CSSU) to perform add/compare selection operations as needed in the Viterbi operator. An exponent encoder is integrated to compute the exponent of a 40-bit accumulator value in a single cycle.

The internal RAMs can be configured as data memory or as program/data memory. The DARAM can be accessed twice per machine cycle, the CPU can read and write to a single block in the same cycle. The TMS320C54x has an independent data-address generation logic (DAGEN, [20]) offering seven basic data addressing modes: immediate addressing, absolute addressing, accumulator addressing,

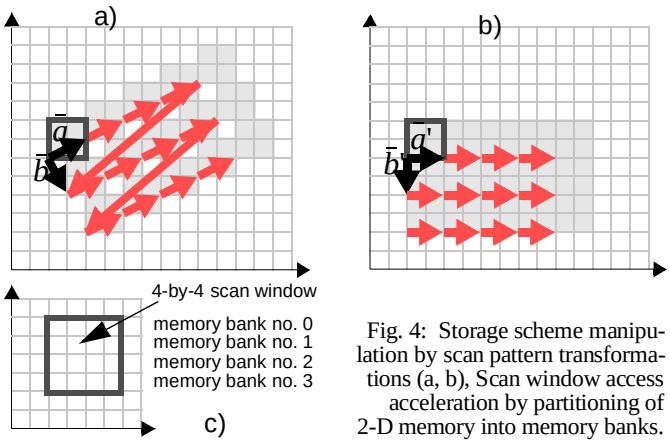


Fig. 4: Storage scheme manipulation by scan pattern transformations (a, b), Scan window access acceleration by partitioning of 2-D memory into memory banks.

direct addressing, indirect addressing, memory-mapped register addressing, stack addressing.

Adopt. is a high-level address optimization and hardware realization environment [21] to minimize the cycle, area and power overhead typical to Data-Transfer Intensive (DTI) applications often making use of embedded distributed memories to cope with the increasing bandwidth requirements exploring different system-level optimizing alternatives suitable for custom processors in partitioned architectures. Adopt includes target architecture selection, address expression splitting/clustering and global algebraic optimizations, amongst others.

The customized MMU architecture (cMMU) uses many small application-specific Address Calculation Units (ACUs)

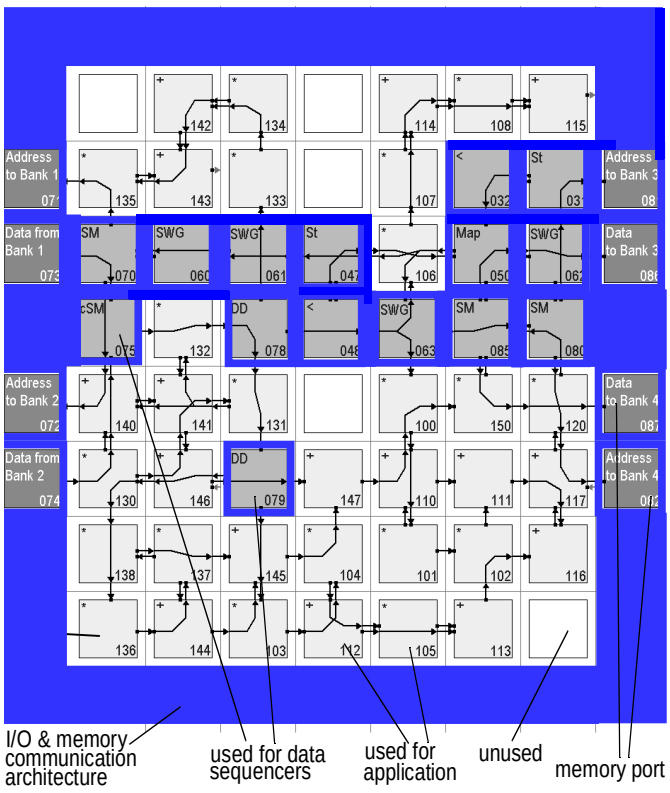


Fig. 5: Mapping application (linear filter) and memory communication architecture (dark background) onto the same KressArray, including the address ports and the data ports to 4 different memory banks (5 of 8 memory port connects are routed through application DPUs).

controlled by a hierarchical (master/slave, global/local) hardwired controller for mapping individual array index reference onto dedicated hardware [21]. An alternative to a fully programmable memory management unit (MMU [22]) architecture is a dedicated architecture. To alleviate the interconnection overhead in terms of delay, power and area associated with routing between underlying (hierarchical) memory architecture and the MMU, a distributed address unit architecture is then selected.

For indexed addressing, a fixed storage ordering is selected when multidimensional signals are presented. Typically this order is first optimized for size reduction by so called in-place mapping [23]. Then, an address expression (AE) for every manifest index expression of a signal located in a linearly addressed RAM is generated. In case of nested loops, the AEs are indexed by common loop iterators. This creates opportunities for cost minimization by using advanced algebraic transformations. In Adopt this is done by applying aggressive transformations on a global scale and over word-level arithmetic [24].

Two clearly differing target architecture styles for the address calculation unit can be identified: incremental address generation unit (iAGU) and custom address calculation unit (cACU). In the iAGU the Address Sequences (ASs) are generated and realized as a counter modified by a two- or multi-level logic filter, under control by the global/master cMMU controller. That is based on the original approach of [25] but extended for more general filters. The least area-and power-expensive solution can be to map the AS onto an iAGU rather than a cACU. This decision depends heavily on regularity properties of the target AS [26]. In the cACU case, the AEs are realized as Application-Specific Units (ASUs) with custom arithmetic building blocks selected from a library being a subset of the Cathedral-3 methodology for lowly multiplexed custom data-path synthesis [21]. This is the first style proposed as an alternative to the iAGU style for the synthesis of address generators [21] [26]. Many other ASs with irregular access patterns are very area hungry, depending on sequence length. Here the AE can be much more efficiently implemented on cACU - a trade-off for power and area since a regular access pattern implemented on a counter has potentially much less toggle activity than with a cACU.

For the ACU style it is better to first identify similar AE clusters (instead of scheduling first, followed by function, mux and register allocation) with the goal to obtain optimal cACU partitioning of the in ASU [27]. After cACU partitioning, further regularity improvement between the shared clusters [28] aiming to decrease the multiplexing overhead is possible by applying local scope algebraic transformations. After that it is mapped by a commercial behavioral synthesis tool.

For iAGU synthesis also AE multiplexing possibilities are provided (at sequence level by expanding the AE at compile time). This way an interleaved AS version of the multiplexed AEs can be obtained. Heuristic measures based on first and second order differences between pairs of AS values are sufficient [29]. The result is a partitioned iAGU architecture in terms of counters and look-up tables. After counter sequence sharing decisions, the characteristics of the counter (i.e., counter modulo value, reset and increment state values, etc.) must be defined still for an optimal synthesis of the iAGU unit. Analysis of the (pseudo) periodicity properties of the target AS [21] [26] is exploited. Also, the optimal assignment of counter to address port-bit, detected during the architecture exploration phase is finally incorporated [26]. Once the exact definition of the different iAGUs has been obtained, RT-level merging possibilities amongst the resulting iAGUs can still be

explored. Again, similarities across shared ASs are used to decide on the best candidates for the merging of counters and/or look-up tables [29]. However, as opposed to the sharing step, the search space is now restricted to ASs with common state transitions (same loop scope) to reuse the counters. Performing the final bit-level optimization of the merged look-up tables is left to subsequent logic synthesis..

Intersil HSP45240 Address Sequencer. The Intersil HSP45240 [30] [31] is a commercial stand-alone address generator device providing specialized addressing for functions like FFTs, 1-D and 2-D filtering, matrix and image operations, and supporting block oriented addressing of large data sets up to 24-bits at clock speeds up to 50 MHz.

The AS is a 24-bit programmable address generator which consists of 4 functional blocks: the start circuitry (SC), the sequence generator (SG), the crosspoint switch (CP), and the processor interface (PI). The crosspoint switch allows for addressing schemes like "bit-reverse" addressing for FFT's and others. The SG is block oriented permitting that the desired address sequence is subdivided into a user-defined number of one or more address blocks with user defined individual numbers of addresses.

HSP45240 architecture and operation is quite sophisticated (for details see [32]). One of three modes of operation may be selected: One-Shot Mode without Restart (see [31]). One-Shot Mode with Restart: Continuous Mode:.

Conclusions. In this section modern and earlier address generators have been surveyed. It can be summarized that hardware support for address generation may be quite effective in featuring parallelism of sequencing operations, and / or reducing the number of memory cycles needed for address generation, and also in reduced configuration code size, and thus fast reconfiguration. In the following the memory bandwidth aspects of the presented address generators will be discussed. Smart address generators may also reduce the number of memory cycles by eliminating multiple accesses to the same data:

3. Customized memory organization and access scheduling issues.

Also address generators having been surveyed in previous section are an important methodological ingredient for memory organization and access scheduling issues. Data Transfer and Storage Exploration (DTSE) [33] is a systematic methodology to optimize data dominated applications for memory size, energy consumption and bus load reduction. The step discussed in this paper involves the design of a (partly) custom memory organization. We believe that this can be directly matched to the opportunities that modern FPGAs like the Xilinx Virtex are providing. The final memory organization should be as power-efficient as possible, but on the other hand its performance must (just) meet the real-time constraints and the memory size should fit in the available memory space.

The design of a custom memory organization consists of two main tasks: introducing and exploiting parallelism, and designing the memory architecture. In the DTSE methodology, this results in two steps. First, the Storage Cycle Budget Distribution (SCBD) step trades off the costly memory bandwidth (i.e. the amount of parallelism of the memory architecture) with the speed that can be achieved with it [34]. This results in a set of constraints on the memory architecture in terms of required parallelism. Second, the Memory Allocation and Assignment (MAA) step constructs a low-cost memory architecture which satisfies these constraints. This detailed memory architecture allows to derive quite accurate data on transfer and storage related costs [35].

The goal of inserting the SCBD step before the actual Memory Allocation and Assignment is to explore different solutions in the speed versus cost (energy or area) trade-off space. This results in a set of so-called Pareto trade-off points: each point represents a solution for which it is not possible to find an alternative which is both faster and less costly. So for each of these points, it is possible to execute the algorithm in more/less time, but at the cost of a less/more expensive (energy-consuming) memory architecture. This freedom can be used in various ways to improve the quality of the design [36]. For instance, when combining several components into a system, the global time and power budget has to be distributed over these. Or even for a single component, the energy versus cycles trade-off can be used to optimize the power supply voltage selection and operating frequency.

Several other approaches have looked at custom memory architectures for a given execution order [37][38][39]. Only [40] takes into account the constraints (access conflict graph related) due to available memory bandwidth and the concurrent accesses imposed by the scheduling. It is directly based on the output of the SCBD step [34].

Many other steps related to background memory management are important for reconfigurable architectures. The reconfigurable data-procedural Xputer [10] [11] machine architecture MoM (Map-oriented machine [41] [42] [43] [44]) supports optimized address sequences and optimum storage schemes [45] [46]. A broad overview of memory organization related issues can be found in [47].

4. Conclusions

The paper has shown that the requirements for memory communication architectures of data-stream-driven reconfigurable computing systems are different from those for traditional parallel computing systems, and, has given a survey on architectural resources and optimization methods to cope with the memory bandwidth problem, especially for (data-)stream-based computing systems, like reconfigurable (and hardwired) datapath arrays.

5. Literature

1. R. Hartenstein: Trends in Reconfigurable Logic and Reconfigurable Computing; ICECS 2002, Sept 15 - 18, Dubrovnik, Coratia, 2002
2. B. Plunkett: Computational Efficiency: Adaptive Computing vs. ASICs; ICECS 2002, Dubrovnik, Croatia, Sept 15-18, 2002
3. J. Becker: Configurable Systems-on-Chip: Commercial and Academic Approaches; ICECS 2002, Dubrovnik, Croatia, Sept 15-18, 2002
4. Paul Master, Bob Plunkett: The Adaptable Computing Machine Concept; SoC 2001 - Enabling Technologies for System-on-Chip Development; Nov. 19-20, 2001, Tampere, Finland
5. A. Pleszkun, E. Davidson: Structured Memory Access Architecture; Proceedings of IEEE International Conference on Parallel Processing, pp. 461-471, 1983.
6. R. Hartenstein, A. Hirschbiel, M. Weber: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90 - International Conference memorating the 30th Anniversary of the Computer Society of Japan, Tokyo, Japan, 1990.
7. R. Hartenstein, R. Hauck, A. Hirschbiel, W. Nebel, M. Weber: PISA, a CAD Package and Special Hardware for Pixel-oriented Layout Analysis; Internal Report 111/84, Department of Computer Science, University of Kaiserslautern, Germany, 1984.
8. R. Hartenstein, R. Hauck, A. Hirschbiel, W. Nebel, M. Weber: PISA, a CAD Package and Special Hardware for Pixel-oriented Layout Analysis; Proceedings International Conference on Computer Aided Design (ICCAD), Santa Clara, CA, USA, 1981.
9. A. Hirschbiel: PISA-Maschine: eine spezielle Hardware für Pixel-orientierte Bildverarbeitung; Diploma-Thesis, Department of Computer Science, University of Kaiserslautern, Germany, 1985.
10. R. Hartenstein, A. Hirschbiel, M. Weber: A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware; InfoJapan'90; International Conference memorating the 30th Anniversary of the Computer Society of Japan, Tokyo, Japan, 1990.

11. 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]
12. D. Grant, P. Denyer, I. Finlay: Synthesis of Address Generators; Proceedings of IEEE International Conference on Computer-Aided Design (ICCAD), pp 116-119, 1989.
13. A. Hirschbiel: A Novel Processor Architecture Based on Auto Data Sequencing and Low Level Parallelism; Ph. d. Thesis, Department of Computer Science, University of Kaiserslautern, Germany, 1991.
14. A. Schaffer: Hardware-Realisierung des JumpGenerators für die MoM; Diploma Thesis, Computer Science Dept., Univ. Kaiserslautern, Oct. 1990.
15. K. Kitagaki, T. Oto, T. Demura, Y. Araki, T. Takada: A New Address Generation Unit Architecture for Video Signal Processing; Proceedings of SPIE International Conference on Visual Communications and Image Processing'91: Image Processing, Part Two of Two Parts, pp.891-900, Boston, MA, USA, Nov. 11-13, 1991
16. H. Reinig: A Scalable Architecture for Custom Computing; Ph. d. Thesis, Department for Computer Science, Univ. Kaiserslautern, 1999.
17. P. Zipf: Entwurf und Implementierung eins Instruction Sequencers für die MoM-3 als Xilinx FPGA; Diploma Thesis, Department of Computer Science, University of Kaiserslautern, Germany, Aug. 1994.
18. M. Weber: Entwurf eines Generischen Adreßgenerators; Diploma Thesis, Department of Computer Science, Univ. Kaiserslautern, March 1993.
19. A. Schwender: Implementierung und Test des Generischen Adreßgenerators der MoM-3; Diploma Thesis, Department of Computer Science, University of Kaiserslautern, Dec. 1995.
20. N.N.: TMS320C54x DSP - CPU and Peripherals; Reference Set, Volume 1, Texas Instruments, USA, 1997.
21. M. Miranda, F. Catthoor, M. Janssen, H. De Man: High-level Address Optimization and Synthesis Techniques for Data-Transfer Intensive Applications; IEEE Trans. on VLSI Systems, no.4, vol.6, Dec. 1998.
22. H. Baehring: Mikrorechner-Systeme - Mikroprozessoren, Speicher, Peripherie; Springer Verlag, Berlin, Germany, 1994.
23. E. De Greef, F. Catthoor, H. De Man: Array Placement for Storage Size Reduction in Embedded Multimedia Systems; Proceedings of 11th International Conference on Application-specific systems, Architectures and Processors, ASAP 97, Zurich, Switzerland, July 14-16, 1997.
24. J. M. Janssen, F. Catthoor, H. De Man: A specification invariant technique for operation cost minimization in flow-graphs; Proc. 7th ACM/IEEE Int'l Symp. on High-level Synthesis, pp. 146-157, 1994.
25. 20. D.Grant, P.Denyer: Address generation for array access based on modulus n counters; Proc. 2nd ACM/IEEE Europ. Design Automation Conf., Amsterdam, The Netherlands, pp.118-122, Feb. 1991.
26. M. Miranda, M. Kaspar, F. Catthoor, H.De Man: Architectural Exploration And Optimization For Counter Based Hardware Address Generation; Proceedings of the 8th ACM/IEEE European Design & Test Conference, pp. 293-298, 1997.
27. S. Note, W. Geurts, F. Catthoor, H. De Man: Cathedral-III: Architecture driven high-level synthesis for high throughput DSP applications; Proceedings of the 28th ACM/IEEE Design Automation Conf., pp. 597- 602, 1991.
28. J. M. Janssen, F. Catthoor, H. De Man: A Specification Invariant Technique for Regularity Improvement Between Flow-graph Clusters; Proceedings of the 7th ACM/IEEE European Design & Test Conf., pp. 138-143, 1996.
29. M. Miranda, F. Catthoor, H. De Man: Address equation multiplexing for real time signal processing applications; VLSI Signal Processing, VII, pp. 188-197, New York, 1994. J. Rabaey, P. Chau and J. Eldon (eds.). IEEE Press.
30. N.N.: HSP45240 Address Sequencer: Data sheet, Intersil Corp., USA, Sep.1997.
31. N.N.: HSP45240/833 Address Sequencer: Data sheet, Intersil, Feb.1998.
32. N. N.: Timing Relationships For HSP45240; Application Note, Intersil Corp., USA, Mar.1998.
33. F. Catthoor, S. Wuytack, E. DeGreef, F. Balasa, L. Nachtergaele, A. Vandecappelle: Custom Memory Management Methodology, Exploration of memory organization for embedded multimedia system design; Kluwer Academic Publ.ishers, Boston, MA, 1998.
34. S. Wuytack, F. Catthoor, G. De Jong, H. De Man: Minimizing the Required Memory Bandwidth in VLSI System Realizations; IEEE Trans. on VLSI Systems, Vol.7, No. 4, pp.433-441, Dec. 1999.
35. A. Vandecappelle, M. Miranda, E. Brockmeyer F. Catthoor, D. Verkest: Global Multimedia System Design Exploration using Accurate Memory Organization Feedback; Proc. 36th ACM/IEEE Design Automation Conf., June 1999.
36. E. Brockmeyer, A. Vandecappelle, and F. Catthoor: Systematic cycle budget versus system power trade-off: a new perspective on system exploration of real-time data-dominated applications; In Proc. IEEE Int. Symp. on Low Power Electronics and Design, pages 137--142, Rapallo, Italy, Aug. 2000.
37. L. Benini, A. Macii, M. Poncino: A recursive algorithm for low-power memory partitioning; Proc. IEEE Int'l. Symp. on Low Power Design, Rapallo, Italy, pp.78-83, Aug. 2000.
38. L. Ramachandran, D. Gajski, V. Chaiyakul: An algorithm for array variable clustering; Proc. 5th ACM/IEEE Europ. Design and Test Conf., Paris, France, pp.262-266, Feb. 1994.
39. P. Lippens, J. van Meerbergen, W. Verhaegh, A.van der Werf: Allocation of multiport memories for hierarchical data streams; Proc. IEEE Int'l Conf. on Computer Aided Design, Santa Clara CA, Nov. 1993.
40. W-T. Shiue, S. Tadas, C. Chakrabarti: Low power multi-module, multi-port memory design for embedded systems; Proc. IEEE Wsh. on Signal Processing Systems (SIPS), Lafayette LA, IEEE Press, pp.529-538, Oct. 2000.
41. R. Hartenstein, A. Hirschbiel, M. Weber: MOM - Map Oriented Machine; in: E. Chiricozzi, A. D'Amico: Parallel Processing and Applications, North-Holland, 1988
42. R. Hartenstein, A. Hirschbiel, M.Weber: MoM - a partly custom-design architecture compared to standard hardware; Proc. CompEuro 89, IEEE Press 1989
43. R. Hartenstein, A. Hirschbiel, K. Schmidt, M. Weber: A Novel ASIC Design Approach based on a New Machine Paradigm; Proc. ESSCIRC`90, Grenoble, France
44. R. Hartenstein, M. Riedmüller, K. Schmidt, M. Weber: A Novel Asic Design Approach Based on a New Machine Paradigm; IEEE J. SSC, July 1991
45. M. Herz: High Performance Memory Communication Architectures for Coarse-Grained Reconfigurable Computing Architectures; Ph. D. Dissertation, Universitaet Kaiserslautern, January 2001
46. M. Herz, et al.: A Novel Sequencer Hardware for Application Specific Computing; Proc. ASAP'97, Zurich, Switzerland, July 14-16, 1997.
47. P. Panda, F. Catthoor, N. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A. Vandecappelle, P. G. Kjeldsberg: Data and Memory Optimizations for Embedded Systems; ACM Trans. on Design Automation for Embedded Systems (TODAES), Vol.6, No.2, pp.142-206, April 2001.