Novel Architecture Design Methodologies and Synthesis Strategies for Future ASIC Design

F. Catthoor, B. Courtois, H. De Man, E. Deprettere,
P. Dewilde, M. Glesner, C. Goutis, J. Jess, J. Madsen,
K. O'Brien, O. Olesen, I. Park, L. Philipson, P. Quinton,
J. Vandewalle, J. v. Eijndhoven, N. Wehn, L. Svensson (ed)

October 11, 1990

Abstract

This paper presents an overview of the main objectives and current achievements of the architecture synthesis work within two Esprit projects: ASCIS ("Architecture Synthesis for Complex Integrated Systems") and NANA ("Novel Algorithms for New Architectures"). The partners in ASCIS are IMEC, Leuven, Belgium (main contractor); Technical University Darmstadt, Germany; Eindhoven University of Technology, the Netherlands; INPG/TIM3, Grenoble, France; Technical University of Denmark, Lyngby, Denmark; Patras University, Greece; and Lund University, Sweden (associate partner). The NANA partners are IMEC (main contractor); Delft University of Technology, the Netherlands; Katholieke Univ. Leuven, Belgium; LIP-IMAG, Lyon, France; and IRISA, Rennes, France. The main goal of this work is to contribute in solving one of the major bottlenecks that restrict the use of ASICs in industrial systems, namely, the lack of efficient design methodologies supported by CAD techniques which allow the creation of a cost-effective application-specific architecture for a given throughput or latency.

1 Introduction

State-of-the-art real-time signal and data processing applications typically involve a rapidly increasing arithmetic complexity, which in many cases is combined with a need for flexible and powerful decision-making operations [All85]. Furthermore, they are not only addressing word-oriented applications, but also employ advanced vector and matrix operations. Loops are abundant and sometimes non-parallelizable, which leads to computational bottlenecks. This large application domain ranges from audio and speech processing, user-end telecommunications, and automotive applications, all of which exhibit low to medium throughputs, up to image, video and radar processing which require much higher sample rates. In addition to such "hard" real-time applications, other data-intensive tasks which require fast execution also increase in importance. Some examples are real-time graphics and medical or other types of data acquisition.

All of these applications have in common that they are difficult to realize in an efficient way on general-purpose hardware with processors of the von-Neumann type [DSP88]. Such realizations result in considerable overheads in terms of memory requirement, I/O bandwidth, I/O interfaces, and controller cost. Furthermore, there is typically a waste of cycles on the general-purpose rigid datapaths. Similar arguments apply largely for MIMD

assemblies of general-purpose processors. In some cases, the design time for these implementation alternatives can be small, but for complex algorithms the mapping stage may require many iterations and time-consuming tuning. The major advantage is the flexibility to make last-minute changes. This is especially interesting during algorithm development, when the algorithm is subject to continuous refinement. However, when the system has reached the production stage we believe special-purpose or application-specific hardware is very heavily motivated in the above cases, especially if the application requires that the hardware have low power consumption or a small physical size, or if the production volume is reasonably high.

Unfortunately, the development cost of a complex ASIC can be very large, especially due to the long design time. State-of-the-art industrial CAD tools only begin to support part of the design path, i.e., from a structural specification down to layout [Gaj88]. The goal of the hardware synthesis work within ASCIS and NANA is to contribute design methodologies and CAD techniques to automate the path from behavior down to the structural specification of the system.

For automatic synthesis to be a viable alternative to hand design, especially in the consumer or domestic markets, a sufficiently large design efficiency has to be achieved. The results should approach the ones of manual full custom designs, in terms of throughput, physical size, power consumption, and packaging. In order to achieve this, we need [DeM90]:

- Efficient target architectural styles underlying the synthesis strategies. Regular arrays and application-specific microcoded processors are two examples of target styles.
- CAD tools which exploit the characteristics of the target style and the corresponding application domain. The tools will frequently be different for the different target styles. For instance, a scheduler which is efficient for one style may be unsuitable for another.

The work described in this paper specifically addresses the following topics, which will have a substantial impact on future ASIC design methodologies:

- Refined system specification formalisms and verification methods.
- Novel design methodologies and CAD techniques for regular-array processors.
- Novel design methodologies and CAD techniques for multiplexed-datapath architectures.
- Generic global optimization techniques to support synthesis approaches for NP-hard design tasks.
- Performance-driven controller synthesis which makes high throughput possible by overcoming part of the control bottleneck.

An overview of the project is given in figure 1.

The extent of each of these contributions will now be summarized in more detail. We believe a transfer of these new technologies to development-oriented projects including large systems companies will help to shape the future ASIC and systems industry in Europe.

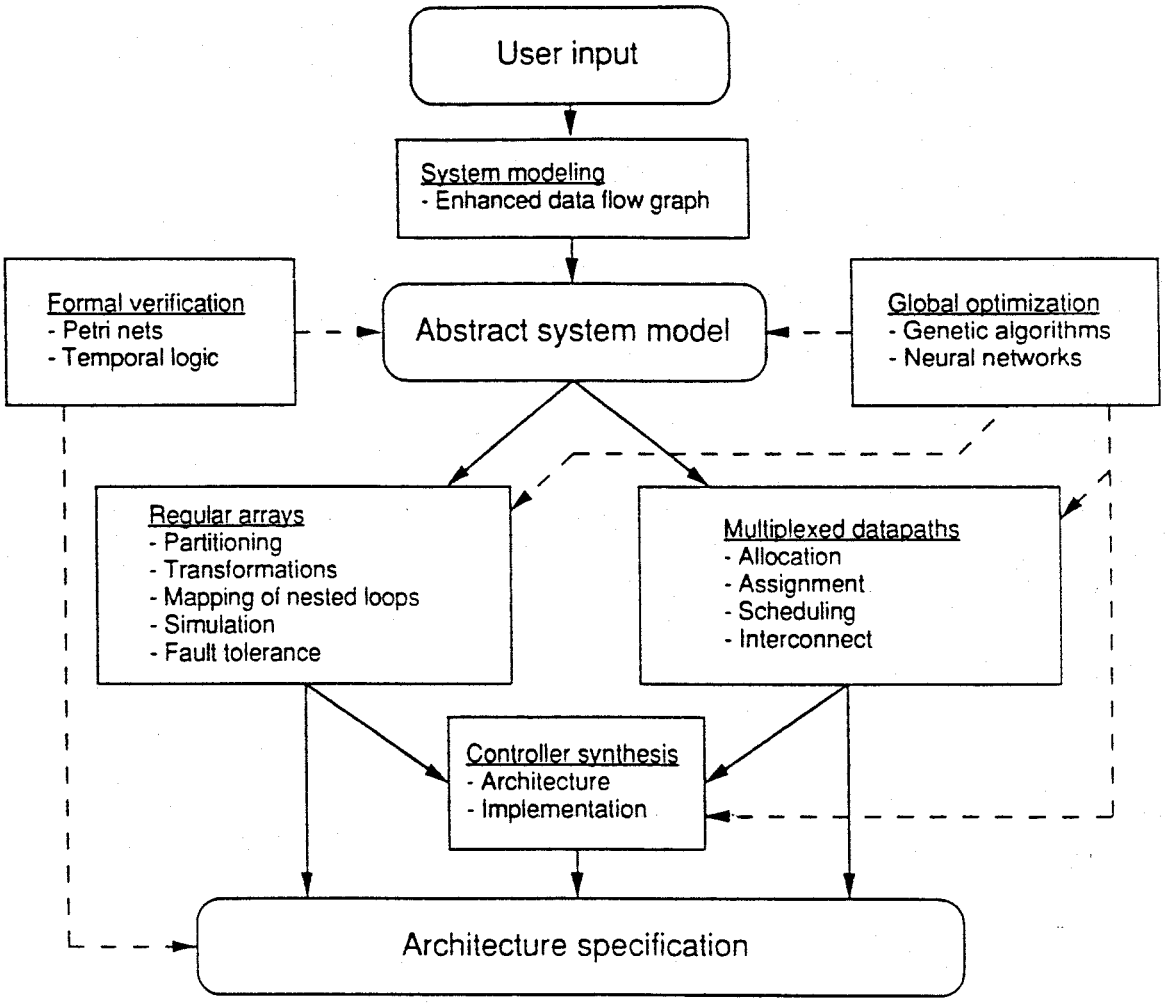

Figure 1: Overview of the architecture synthesis work within ASCIS and NANA. Solid arrows denote flow of design data. Dashed arrows denote application of tools and methodologies.

2 Specification and verification

The reduction of design time, and thus of design costs, is crucial for a more widespread use of ASIC technology. Leaving errors in an ASIC design to be detected in the first production samples is of course a disaster: it will cause large extra costs to locate the error and generate new production masks, combined with a delay of several months to review the design and produce new samples. Especially for small to medium volume ASIC systems, these initial costs and time are crucial, and such errors are hence unacceptable.

To verify the correctness of the designed system, two approaches are in widespread use:

- Verification of the network structure versus a set of "rules" defining a "correct" design style, such as use of clock signals, connections between outputs and so on.
- Simulation of the behavior of the design for a given set of input stimuli.

Although "rule-check" verification is very fast and useful, it does not check the functionality of the design. Simulation, on the other hand, has the following problems:

- Errors in the system may go undetected because they are not made "visible" with the given (finite) set of inputs.
- In practice, it is often hard to spot the errors in the simulation output: they are easily overlooked by the (human) designer who has to interpret the enormous amount of data.

For these reasons, *formal verification* is emerging as a hot research topic for the coming years. In formal verification, the goal is to mathematically prove that the system will behave well in all circumstances. There are several prerequisites to make this possible:

- A *mathematical model* of the system must be available, that allows formal reasoning about its behavior. The models used in simulation programs are not strong enough for this.
- Also, a partial or full *description of the desired behavior* of the system must be available, expressed in mathematical formalisms which fit the mathematical system model. This mathematical specification might be extracted from a more user-friendly input, so that the designer does not have to learn complex formalisms.
- A *proving system*, capable of verifying the required behavior versus the system model, must be developed.

The ASCIS work in formal verification is carried out in Eindhoven. A study of Propositional Temporal Logic has shown its feasibility to verify correctness of sequential logic circuits, CMOS transistor circuits, and finite state machine diagrams against each other [Jan89]. This work has been received with great interest in the worldwide research community, since verifications were done that previously seemed untractable.

In another study from Eindhoven, verification is performed directly on a data flow graph (DFG). This is a highly interesting perspective, since DFGs are widely used as the starting point for synthesis. Furthermore, DFGs are closer to the designers ideas than most other formalisms that allow formal verification. The commonly used DFG formats are not strong enough, but have to be amended if verification is to be possible. Interesting results on refinements of the semantics of DFGs, and subsequent formal verifications of them,

have already been obtained and published [Jon89]. The verifications are of a more global data-independent scope, allowing the verification of much larger systems at the cost of a reduced resolution: detailed, data-dependent functional behavior is not modeled. However, an important and strong verification tool can be based on this principle, which resembles the more traditional Petri net analysis. The resulting refined DFG format has also been adopted as a common specification format by all the ASCIS and NANA partners.

3 Synthesis of regular array architectures

As IC density increases, it becomes more and more attractive to implement special purpose architectures using highly regular parallel algorithms. Examples of such regular organizations are systolic or wavefront arrays, which can be defined as networks of elementary processors which are locally interconnected. The reasons to consider such architectures include:

- *High performance* due to the systematic use of parallelism and pipelining.
- *Decreased design time* obtained by implementing only a few elementary cells which are then replicated many times.
- *Design modularity,* i.e., the possibility of using the same set of cells for the various values of some size parameter of the algorithms.
- *Simplification of test and fault-tolerance issues,* as the complexity of the analysis is broken by the regularity of the design.

Research on formal methods for the design of regular architectures has been going on for about ten years. Briefly, these methods consist of transforming an iterative specification of the desired algorithm into the specification of an architecture, by mapping the initial iteration space onto a new "space-time" space, composed of the processor number space and the execution time. The basic concepts of these methods are now widely accepted and rely upon the notion of the *dependence graph,* which may be seen as a single-assignment representation of the algorithm.

Several problems must be solved for tools based on these formal methods to become widely used in practice. These include choosing a suitable specification language, deciding on strategies for the transformations of the specification, and partitioning an algorithm onto several processor arrays if required. A practical system has to offer simulation capabilities for the array at all levels of transformation. Fault-tolerance issues must also be addressed. The contributions of the ASCIS and NANA partners cover all these areas.

The task of the "front end" of a regular array synthesis system is to build the dependence graph from a more user-friendly specification of the algorithm. Several specification languages are used within the projects. In Delft and in Patras, the algorithms are specified with FORTRAN-style DO loops; at IRISA and at IMEC, functional languages (ALPHA and SILAGE) are used.

The algorithm transformation strategies are receiving much interest. The Patras partner is looking at time-optimal sequence of operations and data transfers [Bir90], and also at techniques for the decomposition of the index space resulting in the minimization of the communication requirements and the execution time [Kyr90]. Work at IMEC is focused in particular on the reindexing transformations needed for efficient mapping of real-time image processing applications, which are never fully regular, onto a modular but not fully

systolic and repetitive ASIC array [VSw90]. The IRISA work concentrates on interactively guided, high-level transformations of the signal dependence graph to make it more uniform. Also, the computability problem of parameterized and conditional dependence graphs has been tackled [QuDo89, BQR89].

The very important problem of partitioning is addressed within the NANA project. The method successfully used in Delft [Bu90] is to first derive a full-size array, then partition this array into a set of serially executed subsets, and, finally, apply processor clustering techniques in order to obtain efficient arrays from inefficient ones.

The regular-array effort in Darmstadt is concentrated on the design of an interactive system for processor array design, including an event-driven simulator which would permit the simulation of the array at all the levels of transformation. Furthermore, novel methods for the introduction of fault tolerance at an early stage in the design have been introduced.

The processor array design methods are tried out on several algorithms, ranging from so-called "kernel algorithms" (matrix computation, filters in one and two dimensions, dynamic programming, SVD decomposition, etc.) to real-life applications such as image analysis and radiosity algorithms.

4 Synthesis of multiplexed datapath architectures

A second important architectural style is characterized by cooperating, application-specific datapaths steered by a hierarchically decomposed controller. This style is tuned to applications requiring a moderate sample rate (10 kHz – 10 MHz). Many applications at these rates require a combination of computation-intensive arithmetic and complex decision-making operations. For these applications, the array style (as described in section 3) is unsuitable.

This architecture style spans different degrees of complexity, both for the datapaths and for the controller. An algorithm with frequent use of nested loops and conditions requires a complex controller. A large amount of computation to be performed in a short sample period calls for complex datapaths, where the operations can be performed in parallel. An algorithm with both these properties consequently leads to a very complex overall processor architecture.

The datapaths may be composed of an arbitrary mixture of primitive abstract building blocks (ABBs), such as registers, adders, shifters, etc. The communication between the cooperating datapaths is supported with dedicated or shared busses, either parallel or serial. The applications may also require a complex (microcoded or hardwired) control unit, which is hierarchically organized or partitioned in order to increase the performance or save area. Both lowly and highly time-multiplexed and parallelized architecture realizations are being studied, as the clock and sample rates almost never match in practical applications. The effects of the presence of nested loops and conditions and of the introduction of advanced pipelining techniques are also being investigated.

Several of the ASCIS partners are working on different aspects of mapping high-level behavioral specifications onto processors of this target style. At INPG/TIM3, the problem of mapping control-dominated algorithms onto highly multiplexed datapaths is investigated. The INPG ABB library will not only contain standard building blocks such as ALUs, but also complex devices such as cache memories or I/O processors. For these devices, the number of clock cycles required for an operation will be unknown at compile time. This increases the complexity of the corresponding controllers [Jerr89]. An algorithm has been developed that will allocate general ABBs to operations in a datapath in such a way as to minimize the overall area of the datapath [POJC91]. This involves trading off areas of

ABBs against the interconnect area the datapath would require. In addition, interactive optimization of the datapath is being facilitated, thereby allowing the user to choose specific ABBs for certain operations and thus guide the system to his desired result.

Both highly and lowly multiplexed datapaths are being investigated at IMEC. One of the most important tasks to be solved for real-time dataflow-dominated algorithms is the memory organization at a high level. A number of test vehicles from the real-time signal processing domain have been selected and manually investigated [Geu90]. The formalization of the collected experience is used in the definition of a general methodology for memory handling. The emphasis lies on the crucial problems of in-place detection, data organization, and address generation [Vba90]. Also, a number of rule-based high-level transformations of the signal flow graph may improve the efficiency of the final implementation. Next, the high-level description has to be broken up into clusters of operation nodes which are tightly coupled. These clusters may then be grouped into sets which are mapped onto a shared custom-built datapath. Here, a combination of algorithmic and rule-based techniques will be required.

In Lyngby, methods are developed for analysis of and hardware synthesis for iterative constructs, with emphasis on handling general (that is, even unbounded) loop structures. In contrast to the situation in DSP synthesis, where the *worst-case* response time has to meet the sample-time criterion, the *average* throughput is the important performance parameter in this target domain, which includes support processors for interactive computer graphics. So far, a VHDL modeling scheme for the data flow graph defined in Eindhoven has been developed [MaBr90]. The DFG serves as input for analysis of loop constructs found in several graphics algorithms, the aim being to classify loops according to parameters such as complexity and trip count.

Note that for all these applications, the presence of a potentially very complex hierarchical control unit increases the complexity of the architecture synthesis. The optimal organization of the controller has to be determined, and then each of the parts has to be realized efficiently. This important part of the design process is also addressed by ASCIS, as described in section 6. Another important item are the scheduling and assignment tasks. These require global optimization techniques, as described in section 5.

5 Generic global optimization techniques

Many problems which must be solved during the mapping process from high-level descriptions to VLSI architectures belong to the class of NP-hard problems. Although good heuristics exist for some specific problems, there is general demand for generic global optimization techniques to efficiently support the synthesis process. Simulated annealing is a good example of such a technique, which has proven to be very useful. However, the complexity of the cost function that is evaluated for every potential step in the solution space and the resulting large computation time restricts the use of this technique in efficient interactive synthesis environments.

In recent years some new optimization methods have been published, promising general applicability with less computational complexity than simulated annealing. Two such new techniques are investigated in Darmstadt and in Eindhoven:

- Genetic algorithms (search algorithms based on the mechanics of natural selection and natural genetics).
- Optimization by means of neural networks.

Both these techniques are characterized by their massive parallelism. Within the ASCIS project, genetic algorithms are used to solve the allocation and scheduling problem in the datapath synthesis. Neural nets are applied to the partitioning problem which is one of the key problems in VLSI synthesis. First implementations of the developed techniques promise quite good results compared to classical approaches [Weh90, PhJo90].

6 Performance-driven controller synthesis

Designing high performance circuits requires not only high performance datapaths but also fast and efficient control units. Building high performance control units requires an appropriate control architecture as well as an efficient implementation (layout). This part of the ASCIS research is carried out in Lund and in Lyngby.

In Lund, work has been performed on optimization of the architectures of control units. One particular architectural feature that has been investigated is the subroutine mechanism. Subroutines enable the reuse of hardware resources; this could significantly reduce the area of the control unit. An algorithm has been developed that automatically transforms a control unit to exploit subroutines in an efficient way [Ran89]. Comparison to handmade designs has shown that the algorithm in many cases produces as good results as an experienced designer. Some resulting controller designs have also been used as test cases for the verification work in Eindhoven (section 2).

The Lyngby work within the area of controller synthesis has been concentrated on the implementation-level optimizations. Multilevel logic implementation of controllers has been investigated [AMMP89], the focus being on performance-driven mapping onto compiled functional cells [Mad89]. A technology mapper has been implemented, which maps the optimized logic onto logic cells created by a cell compiler. The relationship between gate sizing and decomposition has been investigated [Pall90] in order to perform an area-efficient mapping that satisfies the timing constraints imposed on the mapping by the controller synthesis.

7 Conclusion

Already in the first year of the ASCIS and NANA projects, several major contributions to the area of architecture synthesis have been produced. These include: a new abstract system model that allows formal verification of important properties of the algorithm; efficient multi-level simulation of regular arrays and the introduction of fault tolerance in such architectures; several new contributions to methodologies for mapping algorithms both to regular arrays and to multiplexed-datapath architectures; promising applications of novel global optimization methods to NP-hard design problems; and optimization of controllers both at the architecture level and at the implementation level.

To a large extent, the success of the projects so far has been due to the excellent cooperation between the partners, both in the context of formal links between the different tasks and by informal contacts at the frequent meetings and workshops. The cross-fertilization between ASCIS and NANA has also proved especially valuable.

Acknowledgement

We would like to express our appreciation to the many people who have contributed to the progress of the project. In particular, we want to thank P. Andersson, A. Benaini, M. Birbas, J. P. Brage, G. De Jong, F. Franssen, W. Geurts, M. Held, G. Janssen, A. A. Jerraya, E. Kyriakis-Bitzaros, Ch. Mauras, H. Pallisgaard, W. Philipsen, P. Poechmueller, K. Ranerup, Y. Saouter, D. Soudris, A. Stas, A. van der Hoeven, M. van Swaaij, and I. Verbauwhede.

References

[All85] J. Allen, "Computer Architecture for Digital Signal Processing," *Proc. IEEE*, Vol. 73, No. 5, pp. 854–873, May 1985.

[AMMP89] A. C. Andersen, J. Madsen, R. Madsen and H. Pallisgaard, "Automatic Synthesis of Multilevel Combinational Logic." *ESSCIRC-89*, pp. 109–112, Vienna, 1989.

[Bir90] M. K. Birbas, D. J. Soudris, and C. E. Goutis, "A new method for mapping iterative algorithms on regular arrays," *Bilkent International Conference on New Trends in Communication, Control and Signal Processing,* Bilkent, July 1990.

[BQR89] A. Benaini, P. Quinton, Y. Robert, Y. Saouter, and B. Tourancheau, *Synthesis of a new systolic architecture for the algebraic path problem,* Research Report INRIA no 1094 (1989), to appear in *Science of Computer Programming.*

[Bu90] Jichun Bu, *Systematic design of regular VLSI processor arrays,* Ph.D. thesis, May 1990, Delft Univ. of Technology.

[DeM90] H. De Man, F. Catthoor, G. Goossens, J. Van Meerbergen, J. Rabaey, J. Huisken, "Architecture-driven synthesis techniques for mapping digital signal processing algorithms into silicon," *Proc. IEEE,* Vol. 78, No. 2, pp. 58–78, Feb 1990.

[DSP88] B. C. Cole et al., "The embedded processor breaks out of its niche," *Electronics*, pp. 61–87, March 1988.

[Gaj88] D. Gajski (ed.), *Silicon Compilation*, Addison-Wesley, 1988.

[Geu90] W. Geurts, J. Steensma, S. Note, F. Catthoor, and H. De Man, "Design of an efficient architecture for the arithmetic coding algorithm," *Proc. 4th IEEE DSP workshop,* New Paltz NY, Sep 1990.

[Jan89] G. L. J. M. Janssen, "Hardware verification using Temporal Logic: A Practical View," *Proc. of the IMEC-IFIP WG10.2 WG 10.5 International Workshop on Applied Formal Methods for Correct VLSI Design,* L. Claesen (ed.), pp. 291–300, Houthalen, November 13–16, 1989.

[Jerr89] A. A. Jerraya, *Contribution A La Compilation De Silicium Et Au Compilateur SYCO,* these d'etat, INPG, 1989.

[Jon89] G. G. de Jong, "Verification of data Flow Graphs using Temporal Logic," *Proc. of the IMEC-IFIP WG10.2 WG 10.5 International Workshop on Applied Formal Methods for Correct VLSI Design*, L. Claesen (ed.), pp. 301–310, Houthalen, Nov 13–16, 1989.

[Kyr90] E. D. Kyriakis-Bitzaros and C. E. Goutis, "An efficient decomposition technique for mapping uniform recurrencies into regular array processors," *Int. Workshop on Algorithms and Parallel VLSI Architectures* (proceedings to be published in 1991 by Elsevier, Amsterdam), Pont-a-Mousson, June 1990.

[MaBr90] Jan Madsen and Jens P. Brage, "Flow Graph Modelling Using VHDL Bus Resolution Functions," *Proceedings of the First European Conference on VHDL Methods*, IMT, Marseille, Sept. 5–7, 1990.

[Mad89] Jan Madsen, "A new approach to optimal cell synthesis," *Proc. of the IEEE ICCAD*, Santa Clara, Nov. 5–9, 1989.

[Pall90] Henrik Pallisgaard, "A technology mapper for CMOS functional cell generators," submitted to the 28th ACM/IEEE Design Automation Conference 1991, MicroElectronics Center, Technical University of Denmark.

[PhJo90] W. J. M. Philipsen and G. G. de Jong, "Refinement of Petri Nets: The Neural Net Approach," *Proc. of the Int. Neural Network Conf.*, Vol. 1, pp. 266–269, Kluwer Academic Publishers, Paris, July 9–13, 1990.

[POJC91] I. Park, K. O'Brien, A. A. Jerraya, and B. Courtois, "A new algorithm for the optimal allocation of functional units and connections onto multiplexed data paths," submitted to EDAC 1991.

[QuDo89] P. Quinton and V. Van Dongen, "The mapping of linear recurrence equations on regular arrays," *The Journal of VLSI Signal Processing*, 1:95–113, 1989.

[Ran89] K. Ranerup and L. Philipson, "Optimization of finite state machines using subroutines," Technical Report, Dept. of Computer Engineering, Lund University, 1989 (to be published).

[Vba90] I. Verbauwhede, F. Catthoor, J. Vandewalle, H. De Man, "High-level memory management for real-time signal processing of algebraic algorithms on application-specific micro-coded processors," *Int. Workshop on Algorithms and Parallel VLSI Architectures* (proceedings to be published beginning of 1991 by Elsevier, Amsterdam), Pont-a-Mousson, June 1990.

[VSw90] M. van Swaaij, J. Rosseel, F. Catthoor, H. De Man, "High-level synthesis of ASIC regular arrays for real-time signal processing systems," *Int. Workshop on Algorithms and Parallel VLSI Architectures* (proceedings to be published in 1991 by Elsevier, Amsterdam), Pont-a-Mousson, June 1990.

[Weh90] N. Wehn, M. Held, M. Glesner, "A novel scheduling/allocation approach for datapath synthesis based on genetic paradigms," In *IFIP Working Conference on Logic and Architecture Synthesis*, Paris 1990.