29

High-Level Synthesis: A Critical Assessment*

Norbert Wehn
Siemens AG
Semiconductor Division
D-81549 München

Darmstadt University of Technology
Institute of Microelectronic Systems
D-64283 Darmstadt

Abstract

High-level synthesis is a very active research area in VLSI design automation upon which a lot of effort has been spent during the past. However, the high-level synthesis methodology has not yet received the same level of acceptance in industry as logic and RT synthesis. The purpose of this paper is not to give a tutorial[1], but rather to discuss some reasons for this lack of acceptance with respect to commercial issues, to analyze what requirements a high-level synthesis tool needs to fulfill to enable a similar boost in a designer's productivity as logic and RT synthesis tools have and finally to give an outlook on emerging challenges for high-level synthesis tools in the future.

1 INTRODUCTION

In the past ten years we have seen dramatic changes in the field of design automation (see [NEWTON 92] and the Special Universary Issue Electronic Design, Penton Publication [PENTON 92]). In less than a decade designers of complex circuits have come to rely on automatic or semiautomatic computer-aided design systems for physical design, logic synthesis and register-transfer synthesis. The transition from transistor-level entry to gate-level schematic entry enabled an order-of-magnitude improvement in designer productivity. Similarly, the transition from schematic entry to hardware description (HDL) based synthesis has again enabled another order-of-magnitude improvement in designer productivity.

In order to be competitive in today's marketplace (in terms of time-to-market and design-to-cost) CAD tools dominate system and chip design methodologies throughout

*This paper is a revised version of a paper from the IFIP Workshop on Logic and Architecture Synthesis 1993, Grenoble, France.

[1] For a tutorial introduction to high-level synthesis, the interested reader is referred to [MCFARLAND ET AL. 90] or one of the numerous books published on the topic. Some of them have in turn tutorial character [WALKER AND CAMPOSANO 91, MICHEL ET AL. 92, GAJSKI ET AL. 92], others are devoted to application specific synthesis [VANHOOF ET AL. 93, CATTHOOR AND SVENSSON 93] and dedicated synthesis approaches [THOMAS ET AL. 90, CAMPOSANO AND WOLF 91, KU AND DE MICHELI 92, GEBOTYS ET AL. 92].

G. Saucier et al. (eds.), *Logic and Architecture Synthesis*
© IFIP International Federation for Information Processing 1995

the industry. Although the CAD market represents only a few percent ($1.3 billion at the beginning of the '90s) of the total electronic world market, CAD is essential for the electronics industry. The CAD industry is moving very fast. Start-up companies can grow very quickly, and they can disappear very quickly as well. Companies offering CAD synthesis support are credited with an an annual 45% growth rate. SYNOPSYS, one of the most successful synthesis companies, is even credited with a 50% growth rate [COURTOIS 93]. The reason is that the driving system technology is shifting from the computer industry to telecommunication and consumer electronics. Typical examples are digital television, digital video disc, ISDN, CD-player, digital compact cassette, digital mobile phone, mechatronics etc. These products have a short life cycle, but an extremely high degree of system complexity. As a consequence, the designer productivity has to be increased dramatically (by a factor of 15 by the year 2000). This productivity boost can only be achieved by massive use of advanced CAD support beyond the physical design level [DEMAN 92]. There is no doubt that the introduction of logic and RT-synthesis methodologies in industrial design environments was extremely successful and provided a large productivity increase.

2 HIGH-LEVEL SYNTHESIS AS THE NEXT PRODUCTIVITY BOOST?

2.1 High-Level Synthesis within the ASIC Design Flow

High-Level synthesis (there are many synonyms for high-level synthesis in the literature: architectural synthesis, behavioral synthesis etc.) has long been considered to be the successor to logic- and RT-synthesis [IEEE 90]. In the past ten years a lot of effort has been spent in the development of high-level synthesis techniques and there are still many ongoing research projects, both in academia and in industry.

The advantages of shifting to higher levels of abstraction are obvious [MCFARLAND ET AL. 90]: shorter design cycle, fewer design errors, exploration of the design space, the integrated circuit technology becomes available to more engineers ("meet-in-the-middle" approach), to name but a few of them. However, designers are still sceptical of the ability of high-level synthesis to deliver either higher productivity or the required circuit quality. Figure 1 reflects this situation.

The reasons for this situation are manifold. Synthesis on the layout level is primarily based on formal foundations in graph theory and graph algorithms [LENGAUER 90]. Logic synthesis originates from the well known formalism of Boolean algebras [BRAYTON ET AL. 84, ASHAR ET AL. 92]. Extending synthesis to higher levels of abstraction is quite difficult due to the lack of appropriate formalisms. But most important the design space grows tremendously, i.e. given a specification, the set of alternatives for an implementation is very huge (e.g. bit-serial or bit-parallel, pipeline or broadcast, shared or distributed memory, SIMD or MIMD architecture, hardwired or microprogrammed controller, synchronous or asynchronous, bus- or multiplexer-based etc.). Hence, a successful general approach able to manage all implementation alternatives will never exist. This leads to a revised definition of high-level synthesis as depicted in figure 2 [NEWTON 92].

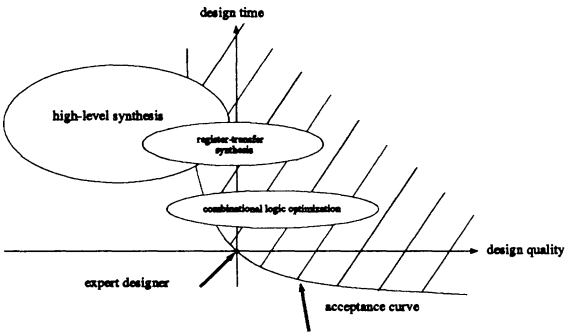

Figure 1 Acceptance of synthesis techniques (Source: A. DeGeus/SYNOPSYS).

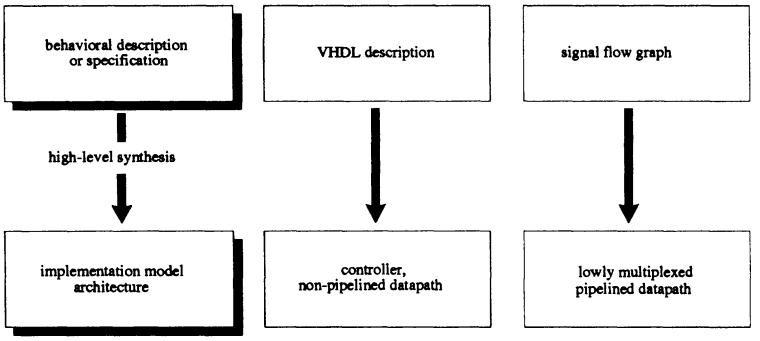

Figure 2 Definition of High-Level Synthesis.

2.2 (Re-)Defining High-Level Synthesis

High-level synthesis is considered to be the task to map a specification or a description (we do not want to elaborate on the difference between *specification* and *description* in this paper) of a system or subsystem onto an *implementation model.* This implementation model is characterized by:

- the definition of a target architecture. Such an architecture typically consists of a set of data paths represented by register-transfer operators, storage units operating under a well-defined timing protocol and control units and their interconnections.
- the definition of a proper specification medium (textual or graphic) with corresponding implementation semantics.
- the identification of the tasks to be solved (allocation, scheduling and binding) and the derivation of their mathematical optimization problems.

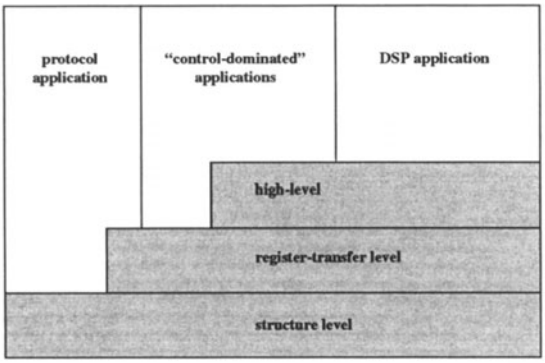

Figure 3 Application domains and proper abstraction levels.

- the order of the different tasks and their interrelations (*synthesis script*).
- the selection of optimization techniques, e.g. global optimization vs. heuristics to solve mathematical optimization problems.

Consider, for instance, a single-process VHDL description which should be mapped onto an architecture consisting of a hardwired controller and a highly multiplexed non-pipelined data-path. Alternatively, consider a signal flow graph representing a filter which should be mapped onto an architecture consisting of a set of lowly multiplexed heavily pipelined datapaths with local controllers. It is obvious that the way of specification and the implementation model completely differ in both cases. *However, many CAD researchers falsely assume that moving to higher levels of abstraction leads them away from implementation/application specific issues.* Instead, the domain dependency of CAD increases when moving to higher levels of abstraction, i.e. the implementation model becomes strongly domain dependent. Experience in the past has shown that mostly tools that respect this rule were able to provide satisfying synthesis results, for instance the Cathedral tool suite [VANHOFF ET AL. 93].

Complex systems are implemented as heterogeneous architectures with e.g. asynchronous interfaces, embedded customized DSP-cores, accelerator datapath-units for computational intensive operations, large portions of RAM, cooperating controlunits and glue logic. Hence there is no single implementation model for such a system, instead a high-level synthesis-environment is necessary which provides various implementation models. Moreover not each part of a system should be specified on a high level of abstraction. High-level synthesis techniques are not suited for every application domain. Figure 3 gives an overview of different application domains and their proper levels of abstraction.

2.3 Requirements for CAD Research

Based on these experiences, we need to redefine the requirements for successful CAD tool research and development. Tool designers, system designers and system engineers have to *cothink* in order to understand each other's problems and should not consider high-

level synthesis from the perspective of inventing yet another new algorithm, for instance for the scheduling problem. Even though scheduling, allocation and binding are indeed important issues in high-level synthesis, they are not the key problems in the high-level design process, especially if these problems are solved on RT-level granularity, where a single multiplexer input is traded off against registers. Global memory considerations and proper partitioning can influence the efficiency of an architecture much more than applying a scheduling algorithm which is optimized for the "fifth-order elliptic filter". In order to capture the "real world" high-level problems for a specific application, one has to be actively involved in the design process and needs to encapsulate application specific design knowlegde in order to be able to define the implementation model. However:

- academic researchers often do not have the possibility to access this design knowledge, since it is property of the system companies. Hence, many academic people spend all their efforts to solve very specific subproblems like scheduling or binding on sometimes unrealistic benchmarks which are tailored to these subproblems [WALKER AND CAMPOSANO 91, High-Level Synthesis Benchmark Suite 91] (the "fifth-order elliptic filter" syndrome). As a result, solutions are sometimes developed which do not contribute to real progress, or the problems addressed do not exist in the "real world".
- even CAD companies generally do not have this knowledge. Hence, it is questionable if the real innovation for the productivity increase will come from CAD vendors.

To overcome this problem, CAD research needs to be oriented more towards the "real life" design process. This requires a closer link between academia, system industry and CAD vendors (the so-called *Trinity* [DEMAN 92]). A shift from *CAd* towards *caD* has to be carried out, i.e. the *design aspect* has to be emphasized. Although this is not only restricted to high-level synthesis, it is especially true for design automation on higher abstraction levels. However, many system houses actually stop their CAD activities to rely on commercial systems. These CAD groups have to be re-oriented to keep their know-how in this field in order to efficiently interface CAD vendors, to match the specific needs of their company and to implement design methods by means of added value tools not supplied by CAD vendors [COURTOIS 93; DEMAN 92].

There is no doubt that high-level synthesis research carried out in the past has produced good results for specific problems, however, for a broader acceptance of high-level synthesis based design methodologies, the following aspects need to be taken into account in addition to the discussed issues:

- Time-to-market pressure and design risk in complex systems make designers to rely on a considerable amount of *reused* subsystems. Today complex ASICs consist of up to 50% of reused silicon, i.e. blocks adapted from previous successful designs, instead of being designed from scratch. In particular, these components must be represented in a suitable way such that high-level synthesis can access them. Hence, efficient design reuse methodologies will play a major role in future design systems [GIRCZYC AND CARLSON 93; DT ROUNDTABLE 94].
- Originally the goal of high-level synthesis was to automatically map a complete system specification down to silicon (ideal silicon compiler) [GAJSKI AND KUHN 83]. But in reality — as already pointed out — complex systems are of heterogeneous nature which combines different architectural styles and implementation paradigms on a single

ASIC and not each of its constituent subsystems is processed by high-level synthesis techniques.

This concludes that high-level synthesis can only be part of a complete design methodology and must be *well adapted* to the overall design environment. Those subcircuits of a complex system processed by high-level synthesis techniques have to satisfy stringent constraints in order to match the system environment. A correct and "user-friendly" specification of a system is a difficult task, since there is no unique specification model. There are various domain-specific entry mechanisms such as graphic statecharts which are based on the "design like you think" paradigm and are well suited for reactive control applications. On the other hand signal processing applications require efficient dataflow models. Hence the specification of complex systems requires the marriage of statechart-based and dataflow techniques.

High-level synthesis must fit into such an environment and must be able to consider timing or ordering constraints on the interfaces and throughput requirements. A consistent simulation of these subcircuits before and after synthesis in the system environment must be supported. This also implies that the circuit description before and after synthesis must be consistent with the remaining system representation. It is often stated that with the advent of VHDL [IEEE 88], the problem of a consistent description and simulation environment has been solved — at least for some specific application domains. However, VHDL was developed as a simulation language and its semantics are defined by means of an event-driven simulator which requires some notion of timing. There is a big difference between simulation and synthesis: simulation is an analysis task in which the timing (as for simulation purposes) has to be well defined, but synthesis is an identification task in which hardware primitives have to be extracted. In this context, we also speak of the "semantic gap" between simulation- and synthesis-related aspects of VHDL. As a consequence, each CAD tool vendor defines its own proprietary VHDL subset for synthesis resulting in non-portable VHDL descriptions. In addition, VHDL is not an adequate specification medium for each application as, for instance, filter specifications can be formulated efficiently in an applicative language such as Silage [GENIN ET AL. 90] instead of an imperative language such as VHDL.

Another important issue for successful high-level synthesis is the modeling of components and estimation of lower-level effects such as delay caused by interconnection. Technology and performance requirements progress so fast that the delay caused by interconnect dominates the overall circuit delay. The problem to decide what RT-level specification will result in a final chip satisfying all area and performance constraints has not yet been sufficiently solved — a prerequisite to do so is an efficient link between RT-level synthesis and layout synthesis tools. Only recently CAD tool vendors have started to tackle this problem. Moving up to a level of abstraction beyond RT-level makes this problem even more difficult. Powerful timing models and a closer link between floorplanning, RT-synthesis and high-level synthesis becomes mandatory. Many components are modelled in a very abstract way which does not reflect their real behavior. A typical case is the modeling of RAMs which in many high-level synthesis-approaches are assumed to behave like simple registers. However large memories are normally realized as DRAMs which can have different architectures (synchronous DRAMs, RAMBUS, EDO-DRAMs) and different access schemes such as burst mode or random access mode.

Interactivity is a further important issue. Due to the huge design space designer interactivity in the high-level design process becomes necessary. Designers usually have a good insight in a system behavior. This knowledge can help in the exploration of the design space. However to allow an efficient interactive high-level design process, fast and accurate estimators are necessary to feedback design decisions [JAIN ET AL. 92; WEHN ET AL. 93; OHM ET AL. 94; RABAEY AND POTKONJAK 94]. Some high-level synthesis-approaches provide powerful high-level transformations to improve the initial specification with regard to various cost criterions before starting the traditional process. Among them are HYPER [CHANDRAKASAN ET AL. 95], CATHEDRAL [VANHOOF ET AL. 93], SAW [WALKER AND THOMAS 89] and Flamel [TRICKEY 87]. These systems offer a wide variety of code transformations, partially well-known from the field of compiler construction [AHO ET AL. 86], such as loop transformations, code inlining, strength reduction, dead code elimination, constant propagation etc. Code is transformed as to optimize some objective function, for instance to improve performance by minimizing delay or, only recently in the HYPER system, to reduce power consumption.

This leads to a new approach in which the high-level design process can be considered as a source-level optimization process. One of the most recent approaches, VOTAN (**V**HDL **O**ptimization, **T**ransformation and **AN**alysis) directly operates on VHDL behavioral descriptions [WEHN ET AL. 94]. It provides a set of code transformations for optimizing control-dominated single-process descriptions subject to resource and performance constraints. Estimators permit evaluation of transformation steps without needing to go into the tedious task of RT-synthesis. In this context, scheduling is considered as yet another code transformation.

In the past application domains were mainly classified according to the predominance of dataflow or control flow. But there is a further category which is neither dataflow- nor control-dominated: *memory-dominated applications* in which memory organization and size influence the overall architecture. Typical examples are video processing algorithms such as video decompression. The overall control and troughput requirements on the various blocks of a decoding pipeline are heavily influenced by the memory bandwidth and size. In order to meet the system throughput requirement (e.g. fixed by the pixel clock), an efficient memory organization and mapping of data onto memories with a minimum number of page-overhead during reading or writing is necessary. The scheduling problem for such an application consists of managing the memory accesses from the different processing units (which are not simple ALUs, but complex blocks like an IDCT-unit or even DSP-cores) in the decoding pipeline. The processing units itself are either dataflow-dominated as for IDCT or picture resizing or control-dominated as in header extraction. This example shows the importance of real case studies. The high-level synthesis community spent a lot of effort in benchmarking which was often counterproductive by focusing research on the wrong issues. Case studies are a possible way to overcome this problem in order to address the real problems.

2.4 The Future — Challenges to CAD Tool Research

Many of the problems discussed above have not yet been sufficiently addressed by high-level synthesis research efforts. Besides, designers in industry are still educated to think and to specify circuits on the RT-level. The previous comments do not imply that the high-level synthesis story has failed. There are numerous successful approaches [CAMPOSANO

AND WOLF 91], especially in the field of DSP for low, medium and high throughput applications like the Cathedral synthesis environment [DEMAN ET AL. 90; CATTHOOR AND SVENSSON 93], the PIRAMID and PHIDEO systems [ROZA ET AL. 92], the HIS system [CAMPOSANO 91] for processor architectures, the CALLAS system [BIESENACK ET AL. 93], the OLYMPUS system [KU AND DE MICHELI 92], PUBSS [WOLF ET AL. 92] and the AMICAL system [CATTHOOR AND SVENSSON 93] for control dominated architectures and many other systems [WALKER AND CAMPOSANO 91]. Even CAD vendors have recently started to enter the high-level synthesis field. In this context, new challenges for high-level synthesis become obvious:

- Recently a trend towards specific application domain processors has become visible, especially in the field of telecommunications [PAULIN 92]. These processors are programmable by an instruction-set and optimized for *a set* of algorithms and not for *one* specific algorithm. This means that some application specific hardware is shifted into programmability and the problem of microcode synthesis for these processors has to be solved. High-level synthesis techniques can be used to efficiently solve the microcode synthesis problem for ASIPs [VAN PRAET ET AL. 94] and the derivation of ASPP architectures [ALOMARY ET AL. 93].
- Design reuse is another challenge for high-level synthesis, since designs specified on a high level of abstraction by an HDL and processed automatically by synthesis tools enhances their reuse potential. Hence, HDL and high-level synthesis methodologies change the vehicle of design reuse from a hardware to a software paradigm [GIRCZYC AND CARLSON 93; DT ROUNDTABLE 94].
- New implementation technologies like FPGAs pose new challenges to high-level synthesis due to their programmability on the hardware level. Hence, design space exploration on the hardware level and emulation in the system environment becomes practicable (rapid prototyping).
- The market for low-power electronic systems like portable computation and communication systems is growing explosively. Hence, power consumption must be considered on the architectural level and is a new facet in the design space exploration. Developing fast estimators for accurate power consumption estimation will be a primary problem, as these are the driving factors for successful search space pruning.
  Among the first systems to consider power issues was HYPER offering high-level estimators for power consumption along with behavioral transformations to reduce glitches and architectural transformations to preserve throughput while lowering supply voltage for low power.
- Low throughput signal processing tasks such as audio processing are more and more implemented on embedded DSP-cores. For design security the global controlling is often organized by an embedded RISC-core. Hence the boundary between hardware and software in complex systems is more and more blurring and the portion of embedded software increases continously. In order to meet the complete system requirements, hardware and software must be considered and developed hand in hand (hardware/software codesign).
- Moreover microelectronic circuits are used more and more in cooperation with other disciplines like in mechatronic- or microsystems. High-level synthesis has to cope with these new directions.

3 CONCLUSION

In this paper we gave a critical assessment on high-level synthesis research activities. A closer cooperation between system designers, CAD tool vendors and research groups could push the high-level synthesis methodology into industrial design environments. New challenges for high-level synthesis were outlined.

4 ACKNOWLEDGEMENTS

I would like to thank my colleagues from Siemens AG and Darmstadt University of Technology for their valuable contributions and helpful discussions. Special thanks go to Michael Münch, Darmstadt University of Technology, for his valuable suggestions.

REFERENCES

A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers — Principles, Techniques and Tools.* Addison Wesley, 1986.

A. Alomary, T. Nakata, Y. Honma, and J. Sato. PEAS-I: A hardware/Software Co-design System for ASIPs. In *Proceedings of the EURO-DAC'93*, pages 2–7, Hamburg, 1993.

P. Ashar, S. Devadas, and R. Newton. *Sequential Logic Synthesis.* Kluwer Academic Publishers, Norwell, MA, 1992.

J. Biesenack *et al.* The Siemens High-Level Synthesis System CALLAS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1(3):244–253, September 1993.

R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, Norwell, MA, 1984.

R. Camposano. Path-Based Scheduling for Synthesis. *IEEE Transactions on Computer Aided Design*, 10(1):85–93, January 1991.

A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Broderson. Optimizing Power Using Transformations. *IEEE Transactions on Computer Aided Design*, 14(1):12–31, January 1995.

B. Courtois. CAD and Testing of ICs and systems. Where are we going? INPG, 1993.

F. Catthoor and L. Svensson. *Application-Driven Architecture Synthesis.* Kluwer Academic Publishers, Norwell, MA, 1993.

R. Camposano and W. Wolf. *Trends in High-Level Synthesis.* Kluwer Academic Publishers, Norwell, MA, 1991.

H. De Man, F. Catthoor, G. Goossens, J. Vanhoof, J. Van Meerbergen, S. Note, and J. Huisken. Architecture-driven synthesis techniques for VLSI implementation of DSP algorithms. *Proceedings of the IEEE*, 78(2):319–335, February 1990.

H. De Man. Design Technology Research for the Nineties: More of the Same? In *Proceedings of the Euro-DAC*, pages 592–596, September 1992.

A D&T Roundtable. Design Reuse. *IEEE Design & Test of Computers*, 11(4):70–77, Winter 1994.

E. Girczyc and S. Carlson. Increasing Design Quality and Engineering Productivity through Design Reuse. In *Proceedings of the 30th DAC*, pages 48–54, June 1993.

D. Gajski, N. Dutt, A. Wu, and S. Lin. *High-Level Synthesis - Introduction to Chip and System Design.* Kluwer Academic Publishers, Norwell, MA, 1992.

D. Gajski and R. Kuhn. Guest Editors' Introduction: New VLSI Tools. *IEEE Computer*, 6(12):11–14, December 1983.

C. Gebotys *et. al. Optimal VLSI Architectural Synthesis.* Kluwer Academic Publishers, Norwell, MA, 1992.

D. Genin, P. Hilfinger, J. Rabaey, C. Scheers, and H. De Man. DSP Specification using the SILAGE language. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, pages 1057–1060, 1990.

High-Level Synthesis Workshop Benchmark Suite. Available via `anonymous ftp` from `mcnc.mcnc.org` in `pub/benchmark`.

Proceedings of the IEEE, February 1990. Special Issue on CAD, Vol.78, No.2.

The Institute of Electrical and Electronical Engineers, Inc., 345 East 47th Street, New York, NY 10017-2394, USA. *IEEE Standard VHDL Language Reference Manual (IEEE Std 1076-1987)*, March 1988.

Rajiv Jain, Alice Parker, and Nohbyung Park. Predicting System-Level Area and Delay for Pipelined and Non-Pipelined Designs. *IEEE Transactions on Computer Aided Design*, 11(8):955–965, August 1992.

D. Ku and G. DeMicheli. *High Level Synthesis of ASICs Under Timing and Synchronization Constraints.* Kluwer Academic Publisher, Norwell, MA, 1992.

Th. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley and Sons, New York, 1990.

P. Michel, U. Lauther, and P. Duzy. *The Synthesis Approach to Digital System Design.* Kluwer Academic Publishers, 1992.

M.C. McFarland, A.C. Parker, and R. Camposano. The High-Level Synthesis of Digital Systems. *Proceedings of the IEEE*, 78(2):301–318, February 1990.

R. Newton. Has CAD for VLSI reached a Dead End? *IFIP Transactions VLSI 91*, A-1:187–192, 1992.

S. Y. Ohm, F. J. Kurdahi, and N. Dutt. Comprehensive Lower Bound Estimation from Behavioral Descriptions. In *IEEE/ACM Digest of Technical Papers*, IEEE/ACM International Conference on CAD, pages 182–187, November 1994.

P. Paulin. DSP Design Tool Requirements for the Nineties: An Industrial Perspective. In *Proceedings of the 6th International Workshop on High-Level Synthesis*, Laguna Niguel, Nov 1992.

Special Universary Issue Electronic Design. Penton Publication, November 1992.

J. van Praet, G. Goossens, D. Lanneer, and H. De Man. Instruction Set Definition and Instruction Selection for ASIPs. In *Proceedings of the 7th International Symposium on High-Level Synthesis*, pages 11–16, May 1994.

J. M. Rabaey and M. Potkonjak. Estimating Implementation Bounds for Real Time DSP Application Specific Circuits. *IEEE Transactions on Computer Aided Design*, 13(6):669–683, June 1994.

E. Roza, J. Biesterbos, B. DeLoore, and J. VanMeerbergen. On the Application of Architectural Synthesis in the Design of High-Volume Production ICs for Consumer Applications. In *Proceedings of the Sixth International Worshop on High-Level Synthesis*, November 1992.

D. Thomas et. al. *Algorithmic and RTL Synthesis: The Systems Architect's Workbench.* Kluwer Academic Publishers, Norwell, MA, 1990.

H. Trickey. Flamel: A High-Level Hardware Compiler. *IEEE Transactions on Computer Aided Design*, 6(2):259–269, February 1987.

J. Vanhoof, K. Van Rompaey, I. Bolsens, G. Goosens, and H. DeMan. *High-Level Synthesis for Real-Time Digital Signal Processing.* Kluwer Academic Publishers, Norwell, MA., 1993.

R. A. Walker and R. Camposano, editors. *A Survey of High-Level Synthesis Systems.* Kluwer Academic Publishers, 1991.

R. A. Walker and D. E. Thomas. Behavioral Transformation for Algorithmic Level IC Design. *IEEE Transactions on Computer Aided Design*, 8(10):1115–1127, September 1989.

N. Wehn, M. Glesner, and C. Vielhauer. Estimating Lower Hardware Bounds in High-Level Synthesis. In *Proceedings of the International Conference on Very Large Scale Integration (VLSI) 93*, pages 6.4.1–6.4.10, September 1993.

N. Wehn, J. Biesenack, T. Langmaier, M. Münch, M. Pilsl, S. Rumler, and P. Duzy. Scheduling of Behavioral VHDL by Retiming Techniques. In *Proceedings of the Euro-DAC '94*, pages 546–551, September 1994.

W. Wolf, A. Takach, C. Y. Huang, and R. Manno. The Princeton University Behavioral Synthesis System. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pages 182–187, June 1992.