Behavioral Synthesis

Raul Camposano
Synopsys, Inc. Mountain View, CA 94043

Extended Abstract

Since the 1988 tutorial on behavioral (high-level) synthesis which appeared in the DAC [1] proceedings (and later in the proceedings of the IEEE [2]) much has happened in the field. Behavioral synthesis is now considered mainstream EDA as evidenced by the number of articles at conferences, journals and books. May be even more significant, several products are being offered commercially, and the number of designers using behavioral synthesis is probably in the hundreds. The initial promise of behavioral synthesis, to dramatically increase productivity by elevating the level of design, has been fulfilled. The increase in productivity of behavioral design versus RTL design is typically quoted at 5 times. What came as a surprise to most researchers in the field, is that this is achieved without impacting the quality of results (area, timing)! If anything, behavioral designs are slightly smaller and faster that their RTL counterparts, mainly because much more architectural exploration can be afforded at the behavioral level, thus providing a better starting point for synthesis.

Yet, there is one caveat. Behavioral synthesis is not as general purpose as RTL or logic level synthesis. Applications that are "good" to be designed at the behavioral level are characterized by a high algorithmic content. Without really getting into more detail on what this exactly means, let me give some examples: graphical applications, printers, disk controllers, parts of ATM, a wide range of DSP applications, complex arithmetic computations, etc. Applications which are not really suitable for behavioral design are for example high performance computers, controllers described by totally specified finite state machines, combinational logic, etc. Overall however, behavioral synthesis can be considered "general purpose" (if this can be quantified at all, what is meant is that a majority of designs done today at the RTL are suitable for behavioral synthesis).

Rather than attempting to give a survey of the field which would require a very large amount of space, the rest of this presentation focuses on a few selected problems which are addressed in more depth in several papers in this volume. The topics covered are memory synthesis, code generation, incorporating interconnect delays at a high level, pre synthesis optimization, and high level power optimization in different flavors. Test and fault tolerance at a high level are covered in Ken Wagner's tutorial [3]. This tutorial presents the problems in context - the reader is referred to the full papers in this volume for any detail.

Memory synthesis is becoming increasingly important. In the general case, the problem is to map the data in the behavioral description to memory elements such as registers, register files and memories. The classical register allocation problem is to determine the smallest amount of shared registers for a given set of variables with associated lifetimes (derived from a control-data flow graph), e.g. [4]. These registers can be clustered into register files. If the amount of data in the behavioral description is very large, typically represented by vectors and multi-dimensional arrays, it is necessary to map it to random access memories [5]. In special cases, other memory structures are more optimal. Ercanli and Papachristou [6] show how shifting register files can be used advantageously in conjunction with expansion scheduling.

Mehendale et. al. [7] address the problem of optimizing code for single register, accumulator based DSP architectures, which are quite common today. Generating code for embedded processors will become increasingly important [8]; this is an area where HW and SW design meet. The particular problem addressed in [7] deals with linear transforms of the form Y=AX where Y are the outputs, X the inputs and A the transformation matrix containing only 0, 1 and -1, i.e. this is a linear, multiplication free transform.

One of the big challenges at present is what is often called "deep sub micron" design. The challenge arises from the fact that most delay in small technologies is due to the interconnect rather than due to the circuits themselves. This has multiple causes such as the resistance of wires becoming comparable to the internal resistance of the drivers, and, to complicate matters further, the cross capacitance of wires may dominate the wire to ground capacitance. The meaning for behavioral synthesis is that although it aims at elevating the level of abstraction, it has to take into account increasingly the physical design (low-level). The paper by Monahan and Brewer [9] shows how to take into account some of the effects of interconnect in the data path at a high level.

Li and Gupta [10] address the problem of pre-synthesis. It is widely acknowledged that very much like software, synthesis in general is sensitive to the style of the input specification (the HDL program). Functionally equivalent HDL programs written in different styles can yield to widely different quality of results (area, speed). The method presented uses timed decision tables for pre synthesis in a way which is relatively insensitive to individual programming styles.

After optimization for area and timing, power is gaining more and more importance. The drive for portable, battery operated devices and the limitation of circuit density by power dissipation in small technologies are probably the most important causes for this trend [11]. Power optimization at the behavioral level potentially yields more savings than at lower levels. Significant savings of power result from architectural

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

changes which allow to reduce the clock speed, shutting down inactive parts of a design using techniques such as gated clocks or isolation with transparent latches. Power optimization is a "hot" topic at present, and consequently this volume contains numerous in this important area.

Raghunathan et.al. [12] address the problem of glitch propagation. Stopping glitches as close to the source as possible maximizes power savings. The techniques used include enhancing data correlation, clocking control signals and gating clock signals. The power reduction reported ranges from 17% to 26%.

Another approach to minimizing power is to use multiple clocks. Each of the multiple clock phases runs at a fraction of the speed than a single clock would do, operating the clocked modules only during the corresponding duty cycle, thus reducing power. Papachristou et.al. [13] show how 3 clocks allow to reduce power consumption by 2-3 times over a single, non gated clock.

An alternative way to look at power management is to shut down pieces of a design while they are not used. Monteiro [14] et.al. present a scheduling algorithm that exploits the slack of operations to obtain a schedule that allows to manage power in this way. The savings reported amount to roughly 40%.

A more specialized approach is introduced by Srivastava and Potkonjak [15]. They minimize power in linear systems unfolding the computations and implementing them maximally fast. This allows to reduce the supply voltage, which reduces the power consumption by a factor of 2.7 in a single processor implementation and by a factor of 15.6 in the multiprocessor case. In the case of a custom ASIC data path implementation, the reported savings is an impressive 30 times.

A topic closely related to power minimization is electromigration reliability. The mean time to failure (MTF) due to electromigration is inversely proportional to the current density J. Maximizing MTF means minimizing J, which is different from minimizing power - although lower power consumption means lower current densities on average. Dasgupta and Karri [16] present a scheduling and binding algorithm to maximize MTF or power. They show that for a set of 3 examples, the "best-reliability" architecture has on average a 48% higher MTF than the lowest power architecture obtained by their algorithm.

Summarizing, although the classical problems of scheduling, allocation and binding have been thoroughly explored in the past, behavioral synthesis is still an active and vital area of research. The cross section of problems addressed in the present volume prove this point.

References

[1] M.C. McFarland, A.C. Parker, R. Camposano, *Tutorial on High-Level Synthesis*, Proc. 25th Design Automation Conference, pp.330-336, June 1988,California

[2] M.C. McFarland, A.C. Parker, R. Camposano, *The High-Level Synthesis of Digital Systems*, Proceedings of the IEEE, Vol.78, no.2, pp.301-318, February 1990

[3] K. Wagner, *Putting High-Level DFT Synthesis in perspective*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[4] F.J. Kurdahi, A.C. Parker, *REAL: A Program for Register Allocation*, Proc. 24th Design Automation Conference, June 1987

[5] M. Balakrishnan, A.K. Majumdar, D.K. Banerji, J.G. Linders, *Allocation of Multi-Port Memories in Data Path Synthesis*, IEEE Transactions on Computer-Aided Design, pp. 536-540, Vol t 7/4, April 1988

[6] E. Ercanli, C. Papachristou, *A Register File and Scheduling Model for Application Specific Processor Synthesis*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[7] M. Mehendale, G. Venkatesh, S.D. Sherleker, *Optimized Code Generation of Multiplication Free Linear Transforms*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[8] S. Liao, S. Devadas, K. Keutzer, S. Tjang, *Instruction selection using binate covering for code size optimization*, Proc. ICCAD'95, 1995

[9] C. Monahan, F. Brewer, *STEM: Concurrent Analyisis Tool for Data Path Timing Optimization*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[10] J. Li, R.K. Gupta, *HDL Optimizations using Timed Decision Tables*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[11] K. Keutzer, *The impact of CAD on the Design of Low Power Digital Circuits*, IEEE Symposium on Low Power Electronics, Oct. 10-12, San Diego, *1994*

[12] A. Raghunathan, S. Day, J. Jha, *Register Transfer Level Power Optimization Techniques with Emphasis on Glitch Analysis and Reduction*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[13] C. Papachristou, M. Spining, N. Nourani, *An Effective Power Management Scheme for RTL Design Based on Multiple Clocks*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[14] J. Monteiro, S. Devadas, P. Ashar, A. Mauskar, *Scheduling Techniques to Enable Power Management*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[15] M. Srivastava, M. Potkonjak, *Power Optimization in Programmable Processors and ASIC Implementations of Linear Systems: Transformation-Based Approach*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada

[16] A. Dasgupta, R. Karri, *Electromigration Reliability Enhancement via Bus Activity Distribution*, Proc. 33rd Design Automation Conference, June 1996, Las Vegas, Nevada