VLSI Digital Signal Processing Education *

Keshab K. Parhi
Department of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455

Abstract

Research in the field of VLSI digital signal processing (DSP) involves study of algorithms and architectures for VLSI implementation of signal and image processing algorithms, theory of architectures and design methodologies, and computer aided design of VLSI DSP systems which can meet the area-speed-power demands of applications using available technologies. The objective in design and implementation of VLSI DSP systems is to reduce the area and power while meeting the speed requirements of the real-time applications. Although about one and a half decades old, this field is still in its youth. The challenges in meeting the speed-power demands of new applications leads to innovations in theory of architectures. At the same time use of old techniques in new applications allows us to further generalize and extend known principles. Because of these continuing advances and because of lack of text books in this field, it remains difficult to educate practicing engineers and graduate students in this field. This paper presents an overview of two ten-week quarter courses offered at the University of Minnesota to train graduate students in this field.

1 Introduction:

The development of digital signal processing (DSP) algorithms and their implementation are influenced by the integrated circuit (IC) technology, the available level of integration and the demands of the applications. DSP is used in numerous applications such as telephony, mobile radio, satellite communications, speech processing, multimedia, video and image processing, biomedical applications, radar, and sonar. All these applications require different sample rates. Real-time implementation of DSP systems requires design of hardware or code for existing hardware so that the system can meet the sample rate requirements of the application. In other words, once the system can meet the sample rate requirement of the application, there is no advantage in designing a faster or larger system. Thus, real-time does not always mean high-speed. Real-time architectures are capable of processing the signals as they are received from the signal source as opposed to storing them in buffers and processing them in batch mode. The implementation of the system is not only affected by the sample rate requirements but also by the complexity of the processing algorithms. It is obvious that a given algorithm will need to be implemented in many different ways for different applications. These ranges of algorithms and applications motivate us to study a wide variety of architecture styles.

Using VLSI technology, DSP algorithms can be prototyped in many ways. These options include (i) single or multi-processor programmable digital signal processors, (ii) use of core programmable digital signal processor with customized interface logic, (iii) semi-custom and field-programmable gate-array implementations, and (iv) full-custom dedicated hardware implementation. High-level algorithm and architecture transformations can play an important role in improving the performance of DSP systems in all of these implementation approaches. In the context of programmable processors, high-level transformations can lead to more efficient compiled code and a reduction in the number of programmable processors required in a multiprocessor environment. On the other hand, for custom implementations, the transformations can lead to reduction in silicon area and/or power consumption. Thus, it is important to both understand the effect of these transformations on DSP circuits and integrate these transformations in DSP hardware and software synthesis systems.

The field of VLSI DSP involves study of algorithms and architectures suitable for VLSI implementation, theory of architectures which can be used to transform an architecture to other equivalent architectures, VLSI implementation styles, and high-level synthesis of DSP systems. Thus, this field bridges the gap between the algorithm designer and the circuit designer. Typically the algorithm designer does not think about the implementation aspects and the circuit designer does not think about the algorithm analysis aspects. This gap often leads to inefficient designs because the opportunity to transform or modify an algorithm to design an efficient architecture may be missed. The field of VLSI DSP exploits the interactions between algorithms and implementations to design efficient systems where higher efficiency might imply lower area

*This research was supported in parts by the National Science Foundation, the Office of Naval Research, the Army Research Office, and the Advanced Research Projects Agency.

or power consumption or higher speed. Many existing algorithm topologies, particularly those containing loops as in recursive or adaptive digital filtering, may not be able to meet the sample rate requirements of high-speed applications such as radar or video. Thus, these topologies will need to be transformed to equivalent topologies or completely new topologies will need to be designed so that these can be operated at higher speed. At a lower level, computer arithmetic architectures and implementation styles can play an important role in design of efficient VLSI DSP systems. These issues also need to be understood by the VLSI signal processing engineer. Finally high-level hardware and software synthesis of VLSI DSP systems is an important topic in this field which at the current time is an active research topic.

The study of VLSI digital signal processing requires understanding of several subjects such as digital signal processing algorithms, stochastic processes, computer architecture, compiler theory, graph theory, VLSI design, digital circuits, computer arithmetic, and computer aided design tools. Depending on the level of involvement of the engineer, one needs to be familiar with a subset of these subjects and need not learn all these subjects. Nevertheless, to develop excellent system solutions, one needs to be familiar with many of these subjects. Typically a VLSI DSP engineer cannot wait to learn these subjects first before learning the field of VLSI signal processing. Thus, many required concepts from these other related fields need to be taught within the scope of the field of VLSI DSP. In addition, since the field is still in its youth and since there are no appropriate text books at this time except the edited book [1], teaching engineers and researchers the field of VLSI DSP remains a challenge. The only sources of materials for learning VLSI signal processing are published papers in journals and conferences. One needs to overcome the discontinuities and notational differences in reading papers in any given topic. Future text books are expected to make learning this field easier.

Many of the techniques used in the field of VLSI digital signal processing are derived from computer architecture and compiler design. In most computer science problems, algorithms always terminate. On the other hand, in DSP, identical computations are repeated very large number of times and these programs can be referred to as non-terminating programs. Because of the non-terminating nature, the principles used in typical computer science applications need to be appropriately modified. DSP operations are described by data-flow graphs and most of the techniques involve transforming one data-flow graph to other equivalent data-flow graphs.

In this paper, we present an overview of two ten-week quarter courses which we offer at the University of Minnesota to train our graduate students in this field. One of these courses "VLSI Digital Filters" addresses the interaction between algorithms and architectures, design of pipelined and parallel digital filters for high-speed and/or low-power VLSI implementation for non-recursive filters such as FIR digital filters and rank-order digital filters, pipelined implementation of recursive and adaptive digital filters using look-ahead and relaxed look-ahead techniques and finite word-length analysis in these filters, theory of systolic architecture design methodology, and implementation of VLSI transforms used in image processing. The second course "VLSI DSP Architectures and Synthesis" addresses theory of architecture transformations (such as unfolding, retiming, pipelining, folding, associativity, distributivity) to improve performance of architectures, VLSI implementation styles (such as bit-serial, digit-serial, and bit-parallel), computer arithmetic architectures, and high-level synthesis of VLSI DSP systems including scheduling and allocation and register minimization. While the first course requires prior understanding of the digital signal processing algorithms, the second course deals with computation aspects of signal processing and does not require involved understanding of signal processing algorithms.

2 VLSI Digital Filters:

The students taking this couse are expected to have taken at least one course in digital signal processing at the level of [2]. This course exploits the interaction between algorithms and architectures to design more efficient topologies for reducing power consumption or making these more suitable for higher speed applications. The second part of this course deals with systolic design methodology and implementation of systolic and semi-systolic architectures for digital filters and transforms used in image processing.

2.1 Pipelining, Retiming, and Parallel Processing:

In this course, we begin with review of pipelining and parallel processing. Many students who have taken one course in digital signal processing and no courses on computer architecture find this review very useful. Straightforward use of pipelining and parallel processing can increase the concurrency in systems which do not contain any feedback loops. It is shown that pipelining can lead to reduction in critical path by placing latches at appropriate feed-forward locations. This reduction in critical path can be exploited to operate the system with higher speed. In other words, pipelining transforms a topology (containing no feedback loops) to an equivalent form which is now suitable for a high-speed application while the original topology cannot meet the speed demands of the application. Pipelining reduces the critical path at the expense of increase in latches and the input-output delay referred to as system latency. How to select appropriate latches in a systematic manner to reduce the number of latches required for pipelining is not taught, since this requires a thorough understanding of retiming. Retiming can also improve the concurrency by moving delays around the system. Unlike pipelining, retiming does not alter the system latency.

Simple examples of retiming are illustrated, but the theory behind retiming is not covered in this course but is covered in detail in the "VLSI DSP Architectures and Synthesis" course. Parallel processing is another approach to increasing concurrency. In parallel processing, multiple inputs are processed to generate multiple outputs. In systems with no feedback loops, this involves duplication of hardware. While the clock speed is not increased by this approach, the sample speed is increased. This is because the parallel system processes multiple inputs in the same time as the sequential system and the sample period, which is the ratio of the clock period to the number of samples processed per clock cycle, is reduced in a parallel implementation.

In many low to moderate speed applications such as speech, audio, wireless communications and in portable computing applications, it is not important to design architectures which can be operated at higher speed. In these applications, it is important to reduce power consumption which can then increase the operation time of the system with the same number of batteries [3]. Pipelining and parallel processing can also be used to reduce power consumption without increasing the system speed [4]. In pipelining, the reduced critical path can be charged or discharged at the same speed by using lower charging/discharging current or by lowering the supply voltage. In parallel processing, the same critical path can be charged or discharged in L times longer time where L is the number of samples processed in a clock cycle such that the sample rate is still same as the sequential system. This also requires lower supply voltage which then leads to lower power consumption. Using the example of FIR digital filters, we demonstrate how pipelining and parallel processing can be used to reduce the power consumption.

In case of higher speed applications such as radar and video, it is possible to design topologies which can be operated at arbitrarily higher speed by using parallel processing and by increasing the number of samples processed in a clock cycle in an arbitrary manner, although at the expense of arbitrary increase in silicon area. A related question which one might ask is "Can the power be reduced in an arbitrary manner at the expense of increasing area?". It turns out that by using parallel processing, the first fundamental limit is reached by the supply voltage limit. In other words, for a given technology and for acceptable noise margins, there exists a lower bound on the supply voltage. This lower limit on the supply voltage dictates the maximum number of samples which can be processed by a parallel system to reduce the power consumption. Beyond this number, increasing the number of samples processed in a clock cycle can only increase the area but cannot decrease the power consumption. However, we can still reduce the power consumption beyond the supply voltage limit by designing parallel architectures where the algorithm complexity increases in a less-than-linear manner with increase in the number of samples processed per clock cycle. Design of such parallel architectures is demonstrated for FIR digital filters using linear transformations and for rank-order filters using substructure sharing [5].

2.2 Pipelining in IIR and Adaptive Digital Filters:

While it is easy to design pipelined and parallel architectures for non-recursive systems such as FIR and rank-order digital filters, it is difficult to design concurrent architectures for recursive systems using pipelining and parallel processing. This is because cutsets in recursive systems do not correspond to feedforward cutsets but feed-back cutsets. Therefore, inserting latches to pipeline these systems alters the number of loop delays which then changes the functionality of the algorithms. Therefore, one needs to increase the concurrency in these systems by first using look-ahead computation where the current state in a recursive system is updated by using a previous state but not its immediate previous state [6]. In other words, if $y(n)$ in the original system is updated by using $y(n-1)$, then in a look-ahead architecture $y(n)$ is computed using $y(n-M)$ which then allows the loop to be pipelined by M levels where the M delays will need to be redistributed or retimed. The look-ahead computation technique allows the emulation of the first order system by an M-th order system which is in contrary to the design of canonic systems in control systems where the order of system is reduced to a minimum. For higher-order recursive digital filters, the look-ahead can take two forms, clustered and scattered. These two forms differ in computation complexity and in stability properties.

No study of VLSI digital filtering is complete without understanding the performance of the novel topologies in a fixed-point finite word-length implementation [7]. In a fixed-point pipelined IIR digital filter, the inexact pole-zero cancellation leads to errors in the filter response and the look-ahead also requires large increase in the hardware complexity. To overcome both these problems, constrained filter design methods are introduced where traditional filter design methods are modified to include pipelining constraints to design filter transfer functions which are inherently pipelined.

In many applications, lattice IIR digital filters result in better finite word-length behavior [8] [9] [10] [11]. Thus, design of pipelined IIR lattice digital filters for basic, normalized, and scaled normalized lattice digital filters is studied in great depth including scaling and roundoff noise study in these structures [12].

In case of adaptive digital filters, it is not important to obtain input-output mapping of sequential and parallel structures. However, the stochastic behavior of the sequential and pipelined topologies should be similar where the stochastic properties such as stability, convergence time constant and misadjustment error of various topologies need to be analyzed. This requires an understanding of stochastic processes. It is shown that the technique of relaxed look-ahead us-

ing sum, product, and delay relaxations can be used to design families of adaptive filter topologies with different adaptation behavior [13] [14]. Using design tools, the appropriate topology can be selected to meet the adaptation behavior required for the application. Design of various pipelined adaptive filters such as lattice adaptive filters and least mean square adaptive digital filters is studied [15].

Use of look-ahead in improving the speed of Viterbi decoders is also illustrated [16] [17].

2.3 Systolic Array Design:

The next part of this course addresses the systolic array design methodology where it is shown how infinite architectures can be designed for a given algorithm using the mapping techniques [18] [19] [20] [21]. Simple examples such as convolution and matrix-vector and matrix-matrix multiplication are used to understand the systolic design methodology. This methodology is then applied to study architectures for several video coding applications such as discrete cosine transforms and motion compensation architectures [22] [23] [24].

3 VLSI DSP Architectures and Synthesis:

The course "VLSI DSP Architectures and Synthesis" addresses the interaction between architectures and integrated circuit design. This course addresses three different aspects of design which include high-level architecture transformations, VLSI implementation styles and computer arithmetic, and high-level scheduling and allocation for synthesis of hardware and software DSP systems. This course deals with computation in signal processing and does not require detailed understanding of DSP algorithms.

3.1 High-Level Transformations:

High-level architecture transformations can play an important role in improving the performance of DSP systems. This course begins with the notion of iteration bound which is the fundamental lower bound on the achievable clock or sample period in any recursive or feedback system [25]. Several algorithms are discussed to compute the iteration bound [26]. It is shown that the iteration bound can be improved by transformations such as pipelining, retiming [27] and unfolding [28]. While pipelining can increase concurrency by placing latches at feed-forward cutsets, retiming can increase the concurrency by moving available delays around the system. Unlike pipelining, retiming does not increase the latency of a system. The technique of systematic retiming can be used for pipelining also. Unfolding is another technique which transforms a program to an equivalent program which describes several consecutive iterations of the same program. Therefore, unfolding can unravel concurrency hidden in a sequential program. Unfolding is similar to loop unrolling or software pipelining used in compiler design. While these techniques can improve the concurrency or the achievable sample rate, they cannot improve the iteration bound in a recursive data-flow graph. The reverse of the unfolding process is the folding process where several computation operations in the data-flow graph are mapped (or folded) to one hardware processor [29]. The folding technique is a powerful technique to design time-multiplexed architectures for DSP data-flow graphs. In the special case of regular data-flow graphs, the folding technique reduces to systolic array design methodology.

3.2 Computer Arithmetic and Implementation Styles:

The second topic addresses VLSI implementation styles and computer arithmetic processor implementations. Three different implementation styles are considered. These include bit-serial [30] [31] [32], digit-serial [33] [34] and bit-parallel [35] [36] [37]. In case of bit and digit-serial, the operations can be either least or most significant bit/digit first. Two's complement addition and multiplication using bit-serial and bit-parallel arithmetic are discussed. It is shown that digit-serial architectures can be designed either by folding the bit-parallel architectures or by unfolding the bit-serial architectures. The use of redundant arithmetic in designing carry-free low-latency arithmetic architectures is considered next [38] [39] [40] [41] [42]. The two main disadvantages in redundant arithmetic are difficulty in sign detection/comparison evaluation and in converting redundant outputs to two's complement outputs. Redundant-to-binary conversion in least and most significant bit first modes and implementation of addition and subtraction using various implementation styles and redundant arithmetic are considered. The clock skew [37] and clocking [43] and asynchronous implementation styles [44] and wave pipelining [45] are also discussed to some extent. To demonstrate the application of these implementation methodologies, a case study of discrete wavelet transform using minimum number of registers [46] [47] is studied [48].

3.3 High-Level Synthesis:

The third topic covered by this course is high-level scheduling and allocation. This aspect begins with the study of synthesis of bit-serial systems where the only problem involved in synthesis is scheduling and no allocation is carried out [49]. The next topic involves time-constrained synthesis where the objective is to synthesize an architecture with least hardware cost to meet the specified time constraint. The dual problem of resource-constrained synthesis is also considered where the objective is to minimize the execution or iteration period of an algorithm where the number of processors is fixed. For synthesis of these systems, several scheduling and allocation strategies

are discussed [50]. These include heuristic methods such as as soon as possible, as late as possible, list scheduling [51], force directed scheduling [52], density scheduling, and iterative loop based scheduling [53]. An overview of the Cathedral silicon compiler is studied next [54] [55]. Optimal synthesis techniques using integer linear programming are also considered [56] [57] [58] [59]. Register minimization is important in high-level synthesis. To this end, register minimization using life-time analysis and register allocation schemes are considered. High-level synthesis with heterogeneous processors is also considered where critical operations are mapped to faster processors such as bit-parallel and non-critical operations are mapped to slower processors such as bit-serial. In a heterogeneous environment, data format converters must be used to reformat the data output from one processor to be input to a different type of processor. In addition, synthesis must consider the latency and cost of these data format converters. This is a challenging problem and is discussed [59].

References

[1] K. Liu and K. Yao, eds., *High-Performance VLSI Signal Processing, Vol. 1: Algorithms and Architectures, Vol. 2: System Design and Applications.* IEEE Press, in press.

[2] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

[3] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[4] K. K. Parhi, "Algorithms and architectures for high-speed and low-power digital signal processing," in *Proceedings of the 4th Int. Conf. on Communications and Control*, (Rhodes, Greece), pp. 259–270, June 1993.

[5] L. E. Lucke and K. K. Parhi, "Parallel processing architectures for rank-order and stack filters," *IEEE Transactions on Signal Processing*, vol. 42, no. 5, pp. 1178–1189, May 1994.

[6] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 1099–1135, July 1989.

[7] K.-H. Chang and W. G. Bliss, "Finite word-length effects of pipelined recursive digital filters," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1983–1995, August 1994.

[8] A. H. Gray and J. D. Markel, "Digital lattice and ladder filter synthesis," *IEEE Transactions on Audio and Electroacoustics*, vol. 21, no. 6, pp. 491–500, December 1973.

[9] A. H. Gray and J. D. Markel, "A normalized digital filter structure," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 268–277, June 1975.

[10] J. G. Chung and K. K. Parhi, "The scaled normalized lattice digital filter," in *Proceedings of the 1993 IEEE Int. Symp. on Circuits and Systems*, (Chicago, IL), pp. 483–486, May 1993.

[11] S. M. Lei and K. Yao, "A class of systolizable IIR digital filters and its design for proper scaling and minimum output roundoff noise," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 10, pp. 1217–1230, October 1990.

[12] J. G. Chung and K. K. Parhi, "Pipelining of lattice IIR digital filters," *IEEE Transactions on Signal Processing*, vol. 42, no. 4, pp. 751–761, April 1994.

[13] N. R. Shanbhag and K. K. Parhi, "A pipelined adaptive lattice filter architecture," *IEEE Transactions on Signal Processing*, vol. 41, no. 5, pp. 1925–1939, May 1993.

[14] N. R. Shanbhag and K. K. Parhi, "Relaxed look-ahead pipelined LMS adaptive filter architectures and their application to ADPCM coder," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 40, no. 12, pp. 753–766, December 1993.

[15] N. R. Shanbhag and K. K. Parhi, *Pipelined Adaptive Digital Filters.* Boston: Kluwer Academic Publishers, 1994.

[16] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS bottleneck," *IEEE Transactions on Communications*, vol. 37, pp. 785–790, August 1989.

[17] P. J. Black and T. H. Meng, "A 40-Mb/s 32-state radix-4 Viterbi decoder," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 12, December 1992.

[18] H. T. Kung, "Why systolic architectures?," *Computer*, pp. 37–46, 1982.

[19] P. R. Cappello and K. Steiglitz, "Unifying VLSI array design with linear transformations of space-time," *Advances in Computing Research*, vol. 2, pp. 23–65, 1984.

[20] S. Y. Kung, *VLSI Array processors.* Englewood Cliffs, NJ: Prentice Hall, 1988.

[21] H. V. Jagadish, S. Rao, and T. Kailath, "Array architectures for iterative algorithms," *Proceedings of IEEE*, vol. 75, no. 9, pp. 1304–1321, September 1987.

[22] M. T. Sun, T. C. Chen, and M. L. Liou, "VLSI implementation of a 16 × 16 discrete cosine transform," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 4, pp. 610–617, April 1989.

[23] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, pp. 1310–1308, October 1989.

[24] P. Pirsch, ed., *VLSI Implementations for Image Communications, Advances in Image Communication 2.* Amsterdam: Elsevier, 1993.

[25] M. Renfors and Y. Neuvo, "The maximum sampling rate of digital filters under hardware speed constraints," *IEEE Transactions on Circuits and Systems*, vol. CAS-28, no. 3, pp. 196–202, March 1981.

[26] K. Ito and K. K. Parhi, "Determining the iteration bound of data-flow graphs," in *Proc. of the IEEE Asia-Pacific Conference on Circuits and Systems*, (Grand Hotel, Taipei), December 1994.

[27] C. Leiserson, F. Rose, and J. Saxe, "Optimizing synchronous circuitry by retiming," *Third Caltech Conference on VLSI*, pp. 87–116, March 1983.

[28] K. K. Parhi and D. G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Transactions on Computers*, vol. 40, no. 2, pp. 178–195, February 1991.

[29] K. K. Parhi, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 1, pp. 29–43, January 1992.

[30] R. Lyon, "Two's complement pipeline multipliers," *IEEE Transactions on Communications*, vol. 24, pp. 418–425, April 1976.

[31] R. Jain *et al*, "Custom design of a VLSI PCM-FDM transmultiplexor from system specifications to circuit layout using a computer aided design system," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 2, pp. 183–195, February 1986.

[32] S. G. Smith and P. B. Denyer, *Serial Data Computation*. Boston, MA: Kluwer Academic, 1988.

[33] R. Hartley and P. Corbett, "Digit-serial processing techniques," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 6, pp. 707–719, June 1990.

[34] K. K. Parhi, "A systematic approach for design of digit-serial signal processing architectures," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 4, pp. 358–375, April 1991.

[35] M. Hatamian and G. L. Cash, "A 70-MHz 8-bit $\times$ 8-bit parallel pipelined multiplier in 2.5 μ CMOS," *IEEE Journal of Solid-State Circuits*, vol. 21, no. 4, pp. 505–513, August 1986.

[36] T. G. Noll *et al*, "A pipelined 330-MHz multiplier," *IEEE Journal of Solid-State Circuits*, vol. 21, no. 3, pp. 411–416, June 1986.

[37] M. Hatamian and G. L. Cash, "Parallel bit-level pipelined VLSI design for high-speed signal processing," *Proceedings of IEEE*, vol. 75, no. 9, pp. 1192–1202, September 1987.

[38] A. Avizienis, "Signed digit number representation for fast parallel arithmetic," *IRE Transactions on Electronic Computers*, pp. 389–400, September 1961.

[39] S. C. Knowles, J. G. McWhirther, R. F. Woods, and J. V. McCanny, "Bit-level systolic architectures for high-performance IIR filtering," *Journal of VLSI Signal Processing*, vol. 1, no. 1, pp. 9–24, 1989.

[40] M. J. Irwin and R. M. Owens, "Design issues in digit serial signal processors," in *Proceedings of IEEE Int. Symp. on Circuits and Systems*, pp. 441–444, May 1989.

[41] H. R. Srinivas and K. K. Parhi, "High-speed VLSI arithmetic processor architectures using hybrid number representation," *Journal of VLSI Signal Processing*, vol. 4, pp. 177–198, 1992.

[42] H. R. Srinivas and K. K. Parhi, "A fast VLSI adder architecture," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 5, pp. 761–767, May 1992.

[43] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 1, pp. 62–70, February 1989.

[44] T. H. Meng and S. Malik, eds., "Special issue on asynchronous circuit design for VLSI signal processing," *Journal of VLSI Signal Processing*, vol. 7, no. 1/2, February 1994.

[45] D. C. Wong, G. De Micheli, and M. J. Flynn, "Designing high-performance digital circuits using wave pipelining: Algorithms and practical experiences," *IEEE Transactions on Computer Aided Design*, vol. 12, no. 1, pp. 25–46, January 1993.

[46] K. K. Parhi, "Systematic synthesis of DSP data format converters using life-time analysis and forward-backward register allocation," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 39, no. 7, pp. 423–440, July 1992.

[47] K. K. Parhi, "Calculation of minimum number of registers in arbitrary life time chart," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 41, no. 6, pp. 434–436, June 1994.

[48] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Transactions on VLSI Systems*, vol. 1, no. 2, pp. 191–202, June 1993.

[49] R. I. Hartley and J. R. Jasica, "Behavioral to structural translation in a bit-serial silicon compiler," *IEEE Transactions on Computer-Aided Design*, vol. 7, no. 8, pp. 877–886, August 1988.

[50] M. C. McFarland, A. C. Parker, and R. Composano, "The high-level synthesis of digital systems," *Proceedings of IEEE*, vol. 78, no. 2, pp. 301–318, February 1990.

[51] J. M. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast prototyping of datapath intensive architectures," *IEEE Design and Test of Computers*, vol. 8, pp. 40–51, June 1991.

[52] P. G. Paulin and J. P. Knight, "Force directed scheduling for the behavioral synthesis of ASICs," *IEEE Transactions on Computer Aided Design*, vol. 8, no. 6, pp. 661–679, June 1989.

[53] C.-Y. Wang and K. K. Parhi, "High-level synthesis using concurrent transformations, scheduling, and allocation," *IEEE Transactions on Computer Aided Design*, 1995 (to appear).

[54] H. De Man, J. Rabaey, P. Six, and L. Claesen, "Cathedral-II: A silicon compiler for digital signal processing," *IEEE Design and Test of Computers*, vol. 3, pp. 13–25, December 1986.

[55] J. Vanhoof, K. Van Rompaey, I. Bolsens, G. Goossens, and H. De Man, *High-Level Synthesis for Real-Time Digital Signal Processing*. Kluwer Academic, 1993.

[56] C.-T. Hwang, J.-H. Lee, and Y.-C. Hsu, "A formal approach to the scheduling problem in high-level synthesis," *IEEE Transactions on Computer Aided Design*, vol. 10, no. 4, pp. 464–475, April 1991.

[57] C. H. Gebotys and M. I. Elmasry, "Optimal synthesis of high-performance architectures," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 3, pp. 389–397, March 1992.

[58] C. H. Gebotys, "Synthesizing embedded speed-optimized architectures," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 3, pp. 242–252, March 1993.

[59] K. Ito, L. E. Lucke, and K. K. Parhi, "Module selection and data format conversion for cost-optimal DSP synthesis," in *Proceedings of IEEE Int. Conf. on Computer Aided Design*, (San Jose, CA), November 1994.