Techniques for Implementation of At-Speed Testable, High Performance, and Low Cost Linear Designs

Miodrag Potkonjak	Sujit Dey	Kevin T. Kornegay
CS Department	C&C Research Labs	Department of ECE
University of California	NEC USA	Purdue University
Los Angeles, CA	Princeton, NJ	West Lafayete, IN

Abstract: Linear computations are most widely used type of ASIC computations. Due to their exceptional theoretical tractability and practical importance, numerous schemes for their implementation have been proposed. The canonical schemes have been widely studied and evaluated according to a number of important criteria, including the number of operations, number of bits, area, throughput and latency, and power metrics. However, until now the testability of linear systems was not studied. After we show that all the most widely used implementation structures require a significant testing related cost, we derive a new structure which is amenable for at-speed testing with no additional hardware overhead. Furthermore, the new structure provides a high throughput, low cost, and low power implementation for an arbitrary linear computation.The key technical novelties of the paper is a novel approach for use of transformations and coordinate use of transformations and scheduling for producing highly testable implementations.

INTRODUCTION

Motivation

ASIC designs, and in particular DSP ASIC components, form one of the fastest growing segments of the semiconductor market. For example, while in recent years the compound growth of overall semiconductor market was about 20%, the compound annual growth of the DSP ASIC market was almost 40%. At the same time, it has been realized that cost of testing is an increasingly important part of DSP ASIC chip costs' [19], sometimes as high as 40% of the overall cost. As behavioral level synthesis tools mature, a large percentage of DSP ASIC designs is done using CAD environments. However, until recently very few high level synthesis tools addressed testability. Our goal in this paper is to develope a method for VLSI ASIC realization of linear computation so that final implementations have not only high throughput, low area, and low power, but also are highly testable. By considering simultaneously effects of several transformations and scheduling and assignment tasks, we have developed an approach which enables generation of high performance, low area, and low cost circuits which are highly testable with *no test hardware overhead.*

Linear systems are defined as systems which satisfy two axiomatic properties: homogeneity and additivity [17]. The first property states that if the response of the system to a signal *x* is a signal *y*, the response of the system to signal *ax* is signal *ay*. The additivity property states that if the response of the system to a signal *x* is a signal *v,* and to a signal *y* is a signal *w*, the response of the system to signal $x + y$ is $v + w$. Linear systems are the most widely studied type of systems due to their intrinsic conceptual simplicity. More importantly, linear system form by far the most dominant component of today VLSI ASIC and application specific programmable market [16]. For example, portable phone DSP functions are mainly linear computations [16]. Linear systems are modeled and optimized using linear computations. Linear computations can be characterized as computations which use only additions/subtractions and multiplications with constants. Note that this definition is not restricted to traditional arithmetic algebraic structures.

What is New?

The first procedure for synthesis of at-speed testable linear systems is presented. By considering simultaneously effects of several transformations and scheduling and assignment tasks, we have developed an approach which enables generation of high performance, low area, and low cost circuits which are highly testable with *no test hardware overhead.*

PRELIMINARIES

In this section we outline all relevant assumptions information about the targeted computational and hardware models, and am assumed testing approach.

Computational and Hardware Model

We assume a synchronous data flow computation model [12], [14], which is often used in high level synthesis for DSP applications. This model assumes a periodic computation done on a seminfinite stream of data along the time loop. A majority of DSP, video, control, and graphics applications follow the selected computational model. Synchronous data flow computation model is also well suited for the application and CAD treatment of transformations. This is mainly because the relatively strict timing discipline imposed by the model provides a convenient basis for the evaluation of changes in the structure of the computations, and enables accurate and rapid predictions of the properties of the final implementation.

Besides the assumed computation model, the selection of targeted hardware model also has numerous consequences. We adopt the dedicated register-file model, in which all registers are grouped in a number of register files. Each register file is connected to only one input of an execution unit, while each execution unit can send data to an arbitrary number of registers files. There are several reasons behind the decision to select the register file model of architecture. First, the model dictates grouping of single read, single write register files which enables area-efficient layout. This is the main reason that the register file model is widely used in general purpose architectures [19] as well as custom ASIC designs [23]. Next, it has been demonstrated that the dedicated register file model reduces the number of interconnect at the expense of a somewhat higher number of registers. As the technology scales, the reduction of interconnect is becoming more important, making the dedicated register file model even more attractive. Finally, availability of accurate and computationally inexpensive area prediction models [3] is an added attraction to use the register file model.

Testability Methodology

It is also important to explicitly state assumptions about the targeted testability methodology. We target testability assuming a single stuck-at fault model, and gate-level sequential ATPG. It is widely accepted that the elimination of all sequential loops, beside self-loops, is most often sufficient to achieve high testability. We target the elimination of all non-trivial sequential loops only in the data-path, assuming full scan of the control logic. For majority of numerically-intensive designs, the data-path completely dominates the area and FF requirements of the design [23], [3]. An effective DFT technique to make circuit testable is partial scan where a set of flip-flops (FFs) are scanned which breaks all non-trivial loops in the circuit [4], [13]. we assume employment of partial scan DFT methodology, and consider the number of partial scan FFs needed to make circuit highly testable as the measure of testability overhead.

In the rest of the paper, we will restrict our attention to the synchronous data flow model of computation and sequential gate-level ATPG of the datapath. It is important to emphasize that all the transformations presented in this paper can be directly explored in other popular computational and hardware models, as long as the elimination of non-trivial sequential loops is a dominant measure of testability.

RELATED WORK

Related work is traced along two lines of research: high level synthesis techniques for testability and transformations.

The mandatory tasks during high level synthesis include allocation, scheduling, and assignment [15],Wal91], all of which have been shown to have significant impact on the testability of the synthesized designs [9]. Existing high level synthesis for testability techniques can be broadly classified according to the testing methodology targeted: BIST, gate-level sequential ATPG, or hierarchical test pattern generation. Four most notable efforts which target BIST have been reported from Case Western Research University [7], [18], Stanford University [2], and University of California at San Diego [11].

Several research groups have developed high level synthesis systems which target sequential ATPG testability. These systems synthesize data paths without loops, by using proper scheduling and assignment, and scan registers to break data-path loops [8]. A datapath synthesized from a behavioral specification may contain several types of loops, e.g.: *CDFG, assignment,* and *sequential false loops* [8]. A CDFG loop is formed in the datapath when there exists a cycle consisting of data-dependency edges in the CDFG. The other types of loops are introduced in the data path during behavioral synthesis, specifically due to hardware sharing. For instance, when operations along a CDFG path from operation u to operation v are assigned to n separate modules, with u and v assigned the same module, an assignment loop of length n is created in the datapath. A comprehensive analysis of the formation of loops, in circuits synthesized by high level synthesis techniques, is presented in [8].

Other works addressing testability during high level design is related to use of architectural information to guide test pattern generation [25], [5], [6].

Transformations are alternations in the structure of a computation so that a particular objective is achieved, while the initially specified functional and timing dependencies between the inputs and the outputs are preserved. Transformations have been successfully used in a number of computation-related areas, including compilers, databases, VLSI algorithms, parallel algorithms, logic synthesis, and computer architecture. Transformations have been successfully used in high level synthesis for optimization of variety of goals, including area, throughput, latency, power, and transient and permanent fault-tolerance.[20], [24], [3], [10], [21].

Recently, a new transformation technique was developed which increases the complexity of the behavioral description while reducing the structural complexity of the resulting datapath [9]. Application of the new transformation technique, hot-potato, to reduce the partial scan overhead for generating easily testable data paths was demonstrated [9]. Recently, Potkonjak et al. demonstrated a high effectiveness of a transformation-based approach for simultaneous optimization of testability and area under throughput constraints [22].

TRANSFROMING AND SCHEDULING LINEAR COMPUTATIONS FOR TESTABILITY

Table 1 shows several design metrics of six different structures of the eighth order IIR Avenhaus bandpass filter [1]. The filter is widely used benchmark in DSP and high level synthesis literature. The table compares various parameters of the resulting implementations: number of registers, area, power, and word-length required for a given frequency response, for a sampling rate of 580ns. None of the different filter implementations were originally testable. Table 1 shows the number of scan flip-flops (FFs) required by BETS [8], a behavioral test synthesis system, to make the corresponding implementations 100% testable. BETS provides algorithms which support testability optimization during resource allocation, scheduling and assignment. BETS targets simultaneously both resource utilization (i.e. area for a given timing constraints) and testability. In all cases a significant test-hardware overhead (scan FFs) is induced. The analysis of the six computational structures indicates that this test hardware overhead is significant and unavoidable, due to the structure of control-data flow graph (CDFG) loops and directed paths in CDFG which have to result in assignment or false loops in the corresponding circuits [8].

structure	direct	cascade	parallel	ladder	continued fraction	wave-digital
number of scan FFs	63	84	77	112	161	108
number of registers	20	28	38	26	24	32
area [mm ^{2}]	30.91	16.90	16.40	27.58	33.71	14.79
power [nJ/sample]	44.55	24.04	17.13	38.73	55.59	15.41
wordlength	21	12	11	14	23	9

Table 1: Comparison of different structures for implementation of linear computations.

However, the following simple procedure transforms an arbitrary linear computation in a form which is highly testable with no test hardware overhead. The procedure is introduced by the following pseudo-code.

Algorithm for producing a highly testable implementation of a linear computation(CDFG) {

Express the states and primary inputs of the CDFG as independent linear combinations of inputs and states;

Unfold the CDFG k times;

Restructure all additions so that they form balanced trees and that all primary inputs are neighbors;

Schedule and assign addition to adders in a such a way that each adder has at least one addition which depends only on the primary inputs (and therefore is fully controllable);

}

The level of unfolding, k, is dictated mainly by the throughput requirements. k has to be high enough not just to satisfy throughput requirements but also that enable that to each adder can be assigned at least one addition which depends only on primary inputs. Note that as the level of unfolding increases the number of primary inputs increases, while the number of states remains unchanged. Therefore, by using unfolding, one can always create as many additions as required which depend only on the primary inputs. Probably the most important relevant observation is that all newly created primary inputs can be assigned to the same physical pins due to hardware sharing.

The initial and final CDFG (after the application of the procedure for making linear systems testable) of the second order single input single output linear system are shown in Figures 1 and 2 respectively. Figure 1 shows the second order IIR direct form filter. Figure 2 shows the same filter after unfolding and the algebraic restructuring. Note that the variables denoted by R1 and R2 are assigned to fully controllable registers. Using hardware sharing those registers are used to store the state variables and therefore provide controllability to all registers in the design. We proved that after the sufficient level of unfolding all nodes in the hardware graph can be made fully controllable and observable by using hardware sharing and connections from primary inputs and to primary outputs.

Figure 3 shows the same filter after unfolding and the application of the algorithm for producing a highly testable implementation of a linear computation. The assumed hardware model now is the dedicated register file model. Note that in this case again hardware sharing provides full controllability of all state variables.

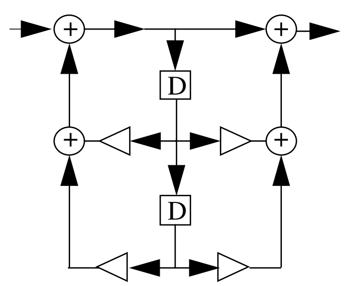

Figure 1: Illustrative Example: The second order recursive linear system

We have applied the new approach to the filter structures shown in Table 1. In each case, the resultant CDFG had an implementation which is 100% testable without the use of scan (no test area overhead), and hence can be tested at-speed. For example, after the application of the new approach to the parallel 8th order Avenhaus filter, the resulting CDFG has an implementation with an area of 9.08 mm^2 and power consumption of 8.24 nJ/sample. The design is 100% testable with no use of

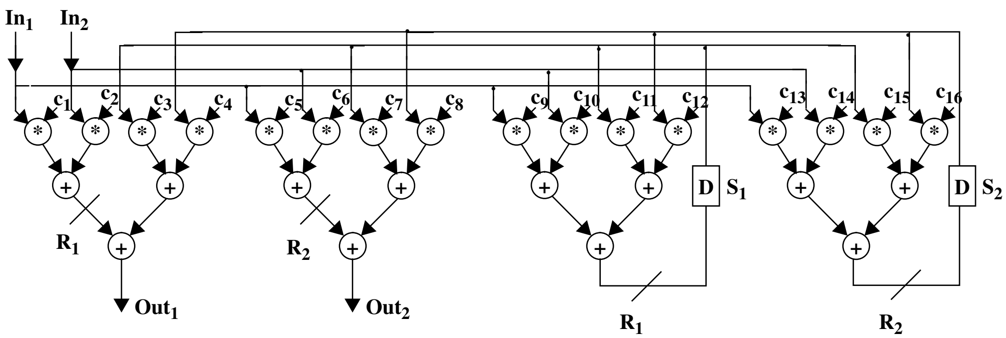

Figure 2: The second order recursive linear system form Figure 1, after the application of synthesis procedure for high throughput, low cost, low power, and high testability implementation.

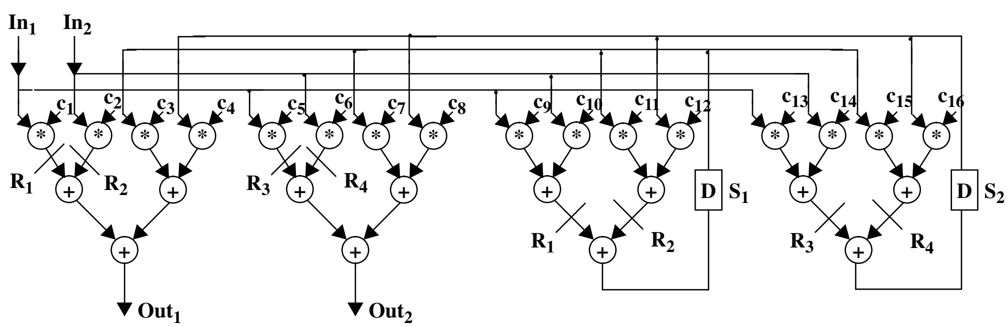

Figure 3: The second order recursive linear system form Figure 1, after the application of synthesis procedure for high throughput, low cost, low power, and high testability implementation. Dedicated register file is the assumed hardware model.

scan registers, hence at-speed testable, and no test area overhead. Similar applications on 7 other linear systems (4 linear controllers (3rd, 4th, 5th, and 3 input 3output 12th order) and 3 IIR (10th, 12th, and 18th order) filters) could also produce 100% testable circuits with no test hardware addition. Finally, we note that only a slight modification is required to apply the synthesis for test procedure on an important subclass of non-linear computations, called feedback linear computations.

SISO LINEAR COMPUTATIONS

Single input single output (SISO) linear time-invariant (LTI) systems are important, widely used special type of linear systems. In particular, SISO LTI systems with zero initial state are often studied and used in many applications. An arbitrary SISO LTI discrete system with a rational system function can be represented in any of numerous canonical forms [17]. Historically most popular canonical forms are direct, cascade, and parallel [17]. Other forms commonly used in practice include ladder, continued-fraction, and wave-digital. Recently, Srivastava and Potkonjak presented an efficient CAD-based approach for transforming a system form one canonical form to another canonical form [24].

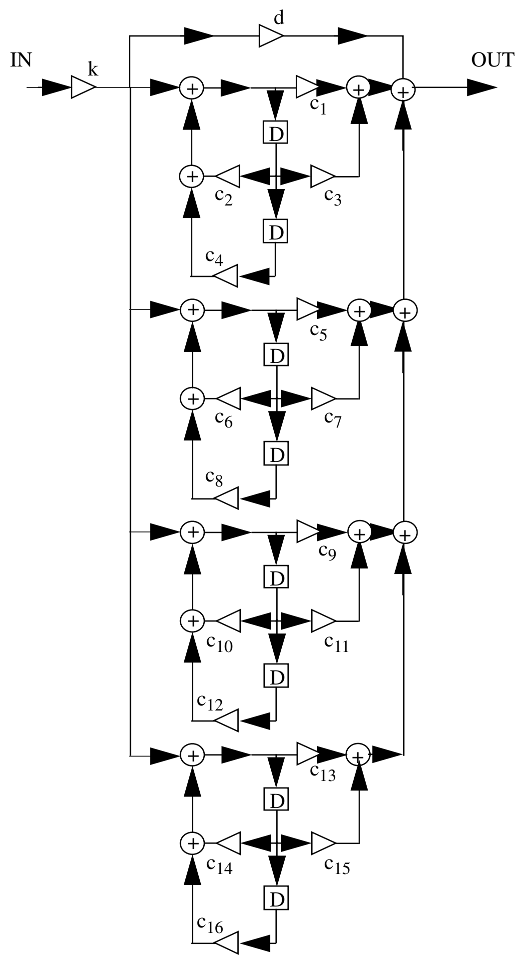

Figure 4: 8th-order parallel IIR filter: example of a starting point for testable, low cost implementation of an arbitrary SISO LTI system.

In the previous section, we presented a method which provides 100% at-speed testable solution for an arbitrary linear computations with no hardware overhead. This solution can be even further improved when a synthesis target is a SISO LTI computation. This is so because the size of the final implementation of the design can be additionally reduced and therefore both the number of faults and test vectors reduced if the required bitwidth and the number of execution and memory units of the design are reduced. To reduce the bitwidth requirements we use a preprocessing step where an arbitrary SISO LTI system is first transformed to the corresponding parallel structure. Figure 4shows an example of a such structure, 8th order IIR parallel filter.

Consequently the procedure described in the previous section is applied. The parallel structure has two advantageous properties. Firstly, it is numerically stable and therefore requires a short bithwidth. We Secondly, when unfolding is applied it results in significantly lower hardware requirements, due to the fact that there is no interaction between computations done in parallel branches of the structure, regardless of the used level of unfolding.

CONCLUSION

We introduced a procedure which transforms an arbitrary linear computation in a form which is highly testable. In some sense the procedure provides an ultimate solution for testing ASIC implementations of linear designs, since the resulting implementation can be tested at-speed with no test hardware overhead while satisfying an arbitrarily high throughput requirements. For an important special case of SISO linear computation, additional optimization steps are providing an additional level for optimization while preserving all advantages of the generic synthesis procedure. Experimental results support the theoretical analysis and optimization algorithms.

REFERENCES

[1] E. Avenhaus: "On the design of digital filters with coefficients of limited word length", IEEE Trans. on Audio and Electroacoustics, Vol. 20, pp. 206-212, 1972.

[2] L. Avra: "Allocation and Assignment in High-Level Synthesis for Self-Testable Data Paths", Proceedings of the International Test Conference, 1991.

[3] A.P. Chandrakasan et. al.: "Hyper-LP: A Design System for Power Minimization using Architectural Transformations", Intl Conf. Computer-Aided Design, Santa Clara, CA, 300-303, November 1992

[4] K.T. Cheng, V.D. Agrawal: "A Partial Scan Method for Sequential Circuits with Feedback", IEEE Trans. on Computers, Vol. 39., No. 4, pp. 544-548, 1990.

[5] C.-H. Chen, D. G. Saab: "BETA: Behavioral Testability Analysis", Proceedings of the International Conference on Computer-Aided Design, pp. 202 - 205, 1991.

[6] V. Chickermane, J. H. Patel: "An Optimization Based Approach to the Partial Scan Design Problem", Proceedings of the International Test Conference, 377-386, September, 1990.

[7] S.S.K. Chiu, C. Papachristou: "A Built-In Self-Testing Approach For Minimizing Hardware Overhead", Proceedings of the International Conference on Computer Design, 1991.

[8] S. Dey, M. Potkonjak, R. Roy: “Exploiting Hardware-Sharing in High Level Synthesis for Partial Scan Optimization”, pp. 20-25, ICCAD93, 1993.

[9] S. Dey, M. Potkonjak, R. K. Roy: “Synthesizing Designs with Low-Cardinality Minimum Feedback Vertex Set for Partial Scan Application”, pp. 2-7, VLSI Test Symposium, 1994

[10] L. Guerra, M. Potkonjak, J. Rabaey: “High Level Synthesis for Reconfigurable Datapath Structures”, ICCAD93, pp. 26-29, November 1993.

[11] I.G. Harris, A. Orailoglu: “SYNCBIST: SYNthesis for Concurrent Built-In Self-Testability”, Proc. IEEE Conference on Computer Design, 1994.

[12] E. A. Lee and D. G. Messerschmitt: “Static Scheduling of Synchronous Dataflow Programs for Digital Signal Processing”, IEEE Trans. on Computers, 1987.

[13] D.H. Lee, S.M. Reddy: “On Determining Scan Flip-Flops in Partial-Scan Designs”, pp. 322-325, 1990.

[14] E.A. Lee, T.M. Parks, “Dataflow Process Networks”, Proc. of the IEEE, Vol. 83, No. 5, pp. 773-799, 1995.

[15] M.C. McFarland, A.C. Parker, R. Camposano: “The High Level Synthesis of Digital Systems”, Proc. of the IEEE, Vol. 78, No. 2, pp. 301-317, 1990.

[16] S.K. Mitra, J.F. Kaiser: “Handbook for Digital Signal Processing”, John Wiley & Sons, Inc., New York, NY, 1993

[17] A.V. Oppenheim, R.W. Shafer: “Discrete-time Signal Processing”, Prentice Hall, Englewood Cliffs, NJ, 1989.

[18] C. Papachristou, et al.: “SYNTEST: a method for high-level SYNthesis with self TESTability, ICCAD, pp. 458-462, 1991.

[19] D.A. Patterson and J.L. Hennessy: “Computer architecture: a quantitative approach”, Morgan Kaufman Publishers, San Mateo, CA, 1989.

[20] M. Potkonjak, J. Rabaey: “Maximally Fast and Arbitrarily Fast Implementation of Linear Computations”, IEEE International Conference on Computer-Aided Design, pp. 304-308, 1992.

[21] M. Potkonjak, J. Rabaey: “Optimizing Resource Utilization Using Transformations” IEEE Transactions on CAD, Vol. 13, No. 3, pp. 277-292, March 1994.

[22] M. Potkonjak, S. Dey, R. K. Roy, "Considering Testability at Behavioral Level: Use of Transformations for Partial Scan Cost Minimization Under Timing and Area Constraints", IEEE Transactions on CAD, Vol. 14, No. 5, pp. 531-546, 1995.

[23] J. Rabaey, C. Chu, P. Hoang, M. Potkonjak: “Fast Prototyping of Datapath-Intensive Architectures”, IEEE Design and Test of Computers, Vol. 8, No. 2, pp. 40-51, June 1991.

[24] M. B. Srivastava, M. Potkonjak: “Transforming Linear Systems for Joint Latency and Throughput Optimization”, EDAC-94 European Design Automation Conference, pp. 267-271, 1994.

[25] P. Vishakantaiah, J.A. Abraham, M. Abadir: “Automatic Test Knowledge Extraction from VHDL (ATKET), Design Automation Conf., pp. 273-278, 1992.