Synthesis-for-Testability Using Transformations

Miodrag Potkonjak, Sujit Dey, Rabindra K. Roy

C&C Research Laboratories, NEC USA, 4 Independence Way, Princeton, NJ 08540

Abstract - We address the problem of transforming a behavioral specification so that synthesis of a testable implementation from the new specification requires significantly less area and partial scan cost than synthesis from the original specification.

A two-stage objective function, that estimates the area and testability of the final implementation, and also captures enabling effects of the transformations, is developed. Optimization is done using a new randomized branch and bound steepest descent algorithm. Application of the transformation algorithm on several examples demonstrates significant simultaneous improvement in both area and testability of the final implementations.

I Introduction

A *Motivation*

Recently, the importance of addressing testability early in the design process has been established [3]. While several scheduling, assignment, and allocation algorithms for testability enhancement have been proposed, only a very limited attention has been paid to exploring the relationship between other high level synthesis tasks, such as transformations and partitioning, and testability.

Clearly, it is important to evaluate the potential of transformations for testability improvements during the high level design process. In this paper, we present a technique which uses a variety of transformations to reduce the area overhead required by Design-For-Testability (DFT) techniques to make the final implementation high testable. During testability optimization, area minimization and timing (throughput) constraints are simultaneously targeted.

In the last few years, numerous high level synthesis approaches which address testability at the behavioral level have been reported [3]. While several systems target Built-In-Self-Test (BIST) and hierarchical testability as the test strategy, a majority of high level synthesis techniques explore the relationship between hardware sharing and sequential Automatic Test Pattern Generation (ATPG) methods. The presence of loops in a sequential circuit is a major source of problems for sequential ATPG. Partial scan is an effective technique to break loops in the circuit by scanning a subset of flip-flops (FFs) [3]. Empirical evidence shows that breaking all non-trivial (with at least two FFs) sequential cycles is an effective heuristic for making a circuit highly testable [3]. In this paper, we minimize the number of scan registers required to break all non-trivial loops in the datapath, and use this number as a measure of testability.

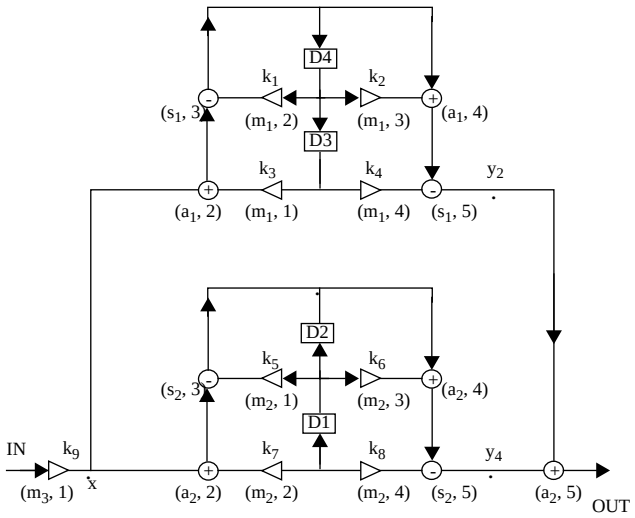

Fig. 1: Motivational example for demonstrating use of transformation for testability: 4th order parallel IIR filter.

We illustrate the use of transformations for testability optimization using the control data flow graph (CDFG) of the 4th order parallel IIR filter, shown in Fig. 1. It is assumed that each operation takes one clock cycle. The available time is six control cycles. The critical path is also six cycles long. To meet the timing constraints, the minimal resource allocation requires 3 multipliers (three multiplications have to be scheduled in the first control step), 2 adders (two additions must be scheduled simultaneously in the second control step), and 2 subtracters (two subtractions must be scheduled in third control step simultaneously).

The result of scheduling and assignment, using an existing behavioral test synthesis system, BETS [4], is shown in Fig. 1. For instance, the first operation, a multiplication by k_9, is assigned to multiplier m_3, and scheduled in control cycle 1, shown by the tuple $(m_3, 1)$. BETS performs scheduling and assignment considering simultaneously testability overhead and area. The resulting minimum feedback set contains four scan registers shown shaded in Fig. 2. As reported in Table 1 (row 4IIR.20), the resulting circuit is highly testable.

It can be shown that it is impossible to reduce the number of scan registers using any other scheduling algorithms. Consider the lower biquad in Fig. 1. In order to break the loops in the corresponding part of the datapath, at least two scan registers are required. If the register file of the transfer unit on which delay D1 is assigned is selected, then all the CDFG loops are broken. However, in this case another scan register is required to break assignment loops which are formed when the two '+' operations have to be assigned to

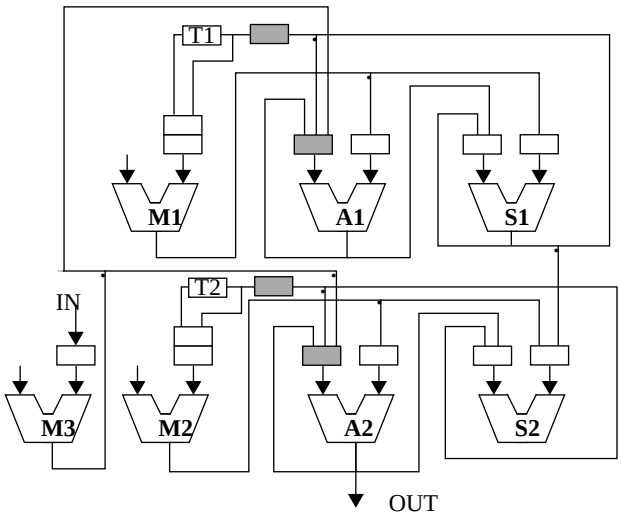

Fig. 2: The datapath of the initial 4th order parallel IIR filter. When the shaded registers are scanned, all loops in the datapath are broken.

the same adder A2. If the input variable to the transfer unit is not selected for scanning, then it is obvious that at least two scan registers are needed just for breaking the CDFG loops. Similarly, it can be shown that a minimum of two scan registers are needed for the upper biquad in Fig. 1. Note that the variables corresponding to delays can not share the same scan register because they are simultaneously alive in the first control step.

Consider now how transformations can be used to simultaneously reduce the area and testability cost of the design. A sequence of transformations shown in Fig. 3.a-c is applied, so that eventually the CDFG shown in Fig. 3.c is obtained. First, algebraic transformations are applied to the CDFG in Fig. 1. to obtain the functionally equivalent CDFG shown in Fig. 3.a. It can be shown that there is a correspondence between the coefficients c_i and k_i used in the two structures. The next transformation applied is the scaling operation [33], where two feedforward cuts are multiplied by β and $1/\beta$, to obtain the CDFG shown in Fig. 3.b. Finally, retiming is used to relocate the delay D3 to three new positions D5, D6, and D7. In addition, the CDFG is pipelined by introducing a delay D8, which is retimed back across the multiplication by c_9. The final CDFG, shown in Fig. 3.c, is significantly more suitable for BETS to produce a testable implementation.

The schedule and assignment obtained by BETS is shown in Fig. 3.c. Although not targeted, the throughput is also improved, since the critical path is reduced to five control steps, and the schedule uses five control steps. The resulting datapath is shown in Fig. 4. As shown in row 4IIR.20 in Table 1, only one scan register is sufficient to break all loops in the datapath, and the data path is highly testable. The hardware requirements are reduced to only two multipliers, one adder and one subtracter.s.

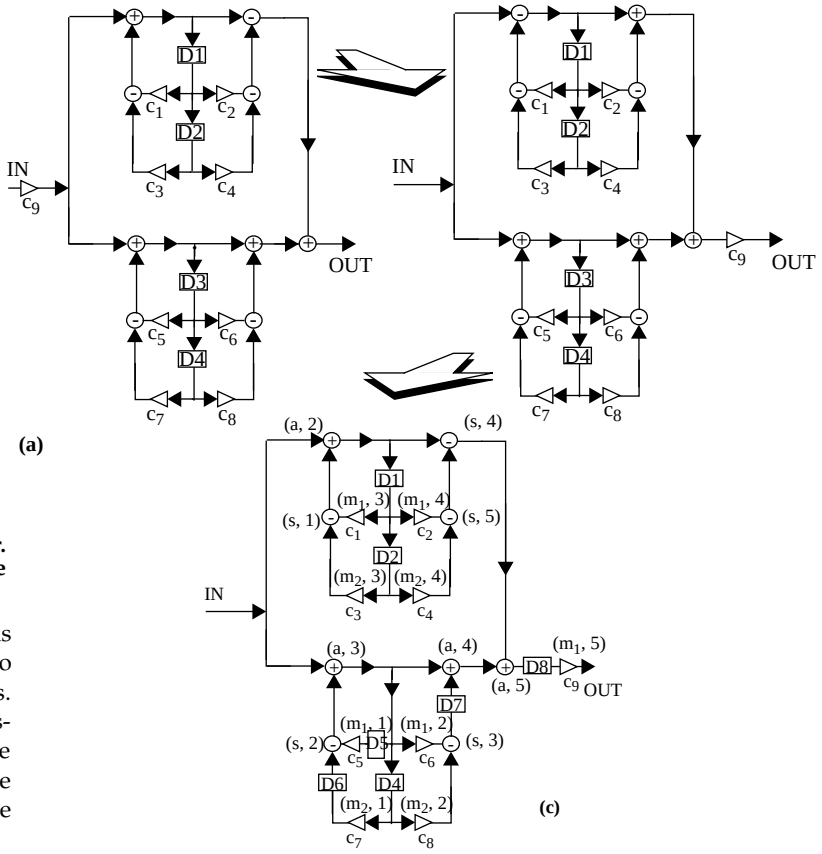

Fig. 3: The sequence of transformations for the simultaneous optimization of testability and area. First (a) associativity and the inverse element law are applied; next (b) scaling transformation; and (c) retiming. The corresponding datapath is shown in Fig. 4.

B *Paper Organization*

The rest of the paper is organized in the following way. In the remainder of this section, we outline the computational and hardware models used and the testing strategy targeted. After a survey of the related work in Section 2, we give an overview of the new approach in Section 3. We explain the transformational mechanism in Section 4. Next,

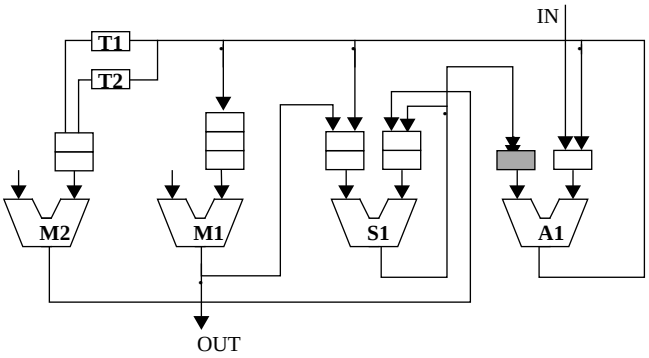

Fig. 4: The datapath of the transformed 4th order parallel IIR. Only one scan register (shaded) is required to break all loops in the datapath, compared with 4 scan register in the initial design.

we present two components of the optimization approach: an objective function and an optimization algorithm, and describe the HITS system, the CAD implementation of the proposed transformation approach. Finally, we demonstrate the effectiveness of HITS on a set of benchmark examples.

C *Computational Model, Hardware Model, and Testability Assumptions*

We assume a synchronous data flow computation model [8], which is often used in high level synthesis. 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. The register file model is widely used in general purpose architectures as well as custom ASIC designs.

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 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 [14, 2].

II Previous Related Work

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

The mandatory tasks during high level synthesis are allocation, scheduling, and assignment [27], all of which have been shown to have significant impact on the testability of the synthesized designs. 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 in [12, 6]. 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 loops [4].

Transformations have been successfully used in high level synthesis for optimization of variety of goals [36, 5, 33].

III Simultaneous Area and Testability Improvement Using Transformations

The proposed behavioral synthesis approach for simultaneous optimization of area and testability was mainly influenced by principles of reusability, modular design, and easy user-interaction. The outlined goals resulted in the following organization (Fig. 5.) of the HITS (HIgh level synthesis system for TeStability and area optimization) system. HITS has three main components: library of transformations, objective function calculation routines, and the optimization algorithm.

The search mechanism invokes the move generation algorithm. which generates sequences of transformations and applies them using the library of transformations, and evaluates them using the fast objective function. Finally, it selects one of the sequences and propose it to the search mechanism. The search mechanism considers the proposed aggregate transformations, and after optional use of accurate objective functions, decides about accepting the transformation. The search is continued or terminated by the search mechanism strategy.

IV Transformational Mechanisms

In this section, we introduce transformations which were found to be particularly effective in addressing testability at the behavioral level. Transformations are classified into three groups: *atomic, global*, and *integrated*.

A *Atomic Transformations*

Atomic transformations consists of a single axiomatic transformation rule at a single position in the CDFG. All atomic transformations can be categorized into three groups: *algebraic laws, redundancy manipulation techniques*, and *control flow transformations*. We currently use four atomic algebraic transformations (commutativity, associativity, distributivity, and the inverse element law), one redundancy manipulation transformation (common subexpression replication), and one control flow transformations (local retiming). All the listed transformations are retargeted to optimize testability. The details of examples how those transformations are used to optimize te3stability are given in the technical report version of this paper. Here, for the sake of brevity, we explain in more details local retiming.

Retiming is a transformation where the number of delays on each output edge of an operation is reduced by one, while the number of delays on each input edge is increased by one, or vice versa. Fig. 6. introduces the application of retiming for testability. The assumed available time is two control cycles. The only type of operation which belongs to both loops is shift. However, due to the timing constraints

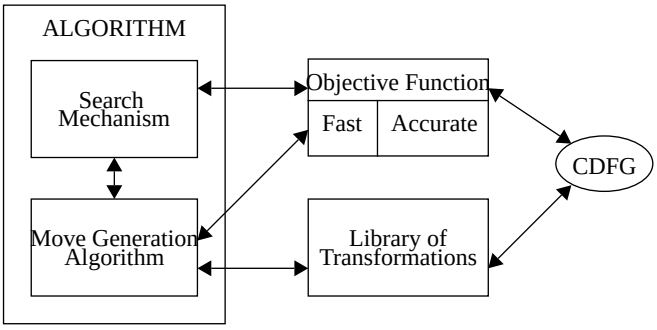

Fig. 5: The conceptual structure of the new approach for testability optimization using transformations.

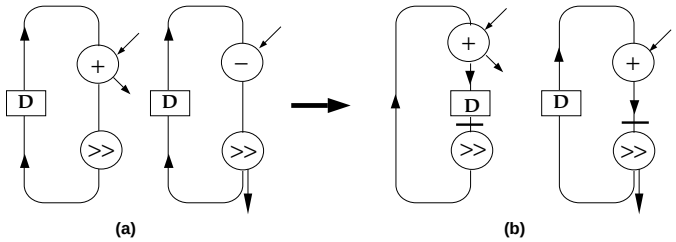

Fig. 6: Retiming for testability optimization. The crossed edges in the transformed CDFG are variables stored in the same scan register.

imposed by the data dependences, both shifts must be scheduled in the second control step and their input variables cannot share the same scan register.

Consider now the functionally equivalent CDFG after the application of retiming, where in the left loop the delay is moved across the shift operation. Now the two shift operations can be scheduled in two different control steps, thereby allowing sharing of the scan registers, denote by the crossed edges in Fig. 6.b. Therefore, the test overhead is reduced from two scan registers to one.

B Global Transformations

Global transformations consists of application of a single transformation mechanism on several places (most often the whole) of the computation. As global transformations we use algebraic speed-up (CZAR) [7], the Leiserson-Saxe retiming and loop unfolding. We illustrate the use of this class of transformations using loop unfolding.

Time loop unfolding is a popular control flow transformation. While initially only one iteration of the computation is considered, after the unfolding by factor k, (k+1) iteration are simultaneously considered during the consequent high level synthesis steps.

Time loop unfolding can directly reduce testability overhead. Fig. 7. supports this claim. Initially, the assumed available time is 2 control steps. At least two scan registers are needed. This is so, because the only common operation present in both CDFG loops is addition. Since both the additions must be scheduled in the first control step, they can not

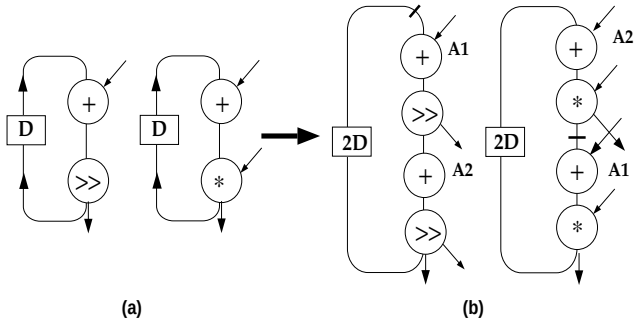

Fig. 7: Unfolding for scheduling. Although unfolding is mainly an enabling transformation, in several scenarios it directly results in the reduction of the required number of scan registers. The crossed edges in the unfolded CDFG can be stored in the same scan register.

share the same execution unit, and their input variables cannot share the same register.

However, if two iterations are considered simultaneously, as provided by time loop unfolding and shown in Fig. 7.b, only one scan register is required to break all loops in the corresponding datapath. After the application of unfolding, four control steps are available for scheduling two iterations of the computations. Note that unfolding enabled two variables from two different iterations of the two loops to share the same scan register, since they do not have overlapping lifetimes.

C Integrated Transformations

Integrated transformations consist of several transformations being simultaneously applied on a part of the computation or the whole computation. Currently, we use only one integrated transformation, introduced in [15]. The application domain of this transformation are linear computations. An arbitrary linear computation (after the application of this transformation) is transformed in one of several canonical forms (e.g. cascade form or parallel form). While, the canonical forms do not necessarily result in a highly testable design, it can be shown that the consequent application of local transformations often result in CDFGs which have highly testable implementations.

V Objective Function

In order to satisfy these demanding and contradictory requirements, we selected an objective function which explicitly targets the three key aspects of evaluating the design at the behavioral level.

1. **Testability Cost (MFVS).** Estimated test hardware requirements, in our case the size of the minimum feedback set of registers which break all loops in the datapath;
2. **Scheduling Difficulty (SD)**. Estimated scheduling difficulties due to Area (amount and structure of datapath components) and Timing (throughput) goals and constraints
3. **Transformation Difficulty (TD).** Suitability of the CDFG for the consequent application of transformations.

Testability Cost. (MFVS) is equal to the number of feedback vertices required to break all loops, besides self-loops, in the data-dependency and compatibility graph (DDCG). The DDCG and the testability cost are reused from the BETS system [5], where they were supporting scheduling ands assignment for testability.

SD is proportional to the expected cost of the datapath area. First, the expected quantities of primary datapath components are estimated using the compatibility edges of the DDCG. For each type of hardware unit, the estimated number of instances is the proportional to the number of corresponding instances in the DDCG and the inverse value of the

cumulative sum of the weights of all corresponding compatibility edges. This information is used as the input in the statistical model for layout prediction developed in [2].

TD is proportional to the weighted change in the number of further transformations which can be applied on the graph after the application of a particular transformations. The weight is proportional to importance of applying consequent transformation on the particular node of the CDFG. Experimentally derived formula for quantifying the importance has the following as input parameters: the slack of the operation, size of the strongly connected component to which it belongs, the cardinality of the transitive fan-in and transitive fan-out set, and the cost and delay (number of control steps required that the operation is executed).

The three components are weighted by three empirically derived factors. We conclude the section by the pseudo-code description of the fast objective function:

Fast_Objective_Function FOF (CDFG) {
Form DDCG;
SD = Scheduling_Difficultiy();
MFVS = Testability_Cost();
TD = Transformation Difficult();
*FOF = γ_1 * SD + γ_2 * MFVS + γ_3 (t) * TD;*
}

VI Optimization Algorithm

As we already described in Section 3, the optimization algorithm has two components: move selection algorithm and search strategy. The first component, move selection algorithm, is captured by the following pseudo-code:

Move_Selection_Algorithm (CDFG) {
Randomly select the number of global and local moves;
Select and apply global moves;
Evaluate FOF; If the FOF increased by more than 10% terminate move;
Select and apply global moves;
Evaluate FOF;
}

The search algorithm uses a dynamic randomized deepest decent search mechanism and is described using the following pseudo-code:.

Search_Strategy_Algorithm
Initialization(); Integrated Transformation();
if (new_FOF < Current AOF) {
if (new_AOF < Current AOF) update Best_Cost and Current_Best_Solution; }
Initialize criteria1, criteria2;
while (criteria1 is satisfied) {
while (criteria2 is satisfied) {
Move_Selection_Algorithm (CDFG)
if (new_FOF < Current AOF) {
if (new_AOF < Current AOF) update Best_Cost and Current_Best_Solution;}
update criteria 2;
} }

The Initialization routine sets as the best current solution the initial solution, and sets as the best cost the initial cost. Criteria for stopping the Move_Selection_Algorithm (CDFG) and the whole search are either user dictated time limit or requested size of the datapath and the number of scan registers.

VII HITS and Experimental Results

The proposed transformations, objective functions and optimization algorithms are used as the core of HITS (HIgh level Transformation-based synthesis system for teStability optimization). The input to HITS is a functional specification of the application in an applicative stream-oriented Silage language [14]. The input is parsed using the Hyper parser, and the computation is stored in the Hyper CDFG format [14]. HITS operates on the Hyper CDFG format.

The transformed CDFG is given as input to the BETS behavioral test synthesis system [4], which performs allocation, scheduling, and assignment, targeting simultaneously area and testability, while satisfying throughput constraints. The output of BETS is an annotated CDFG, where each node and edge attributes have corresponding scheduling and assignment information. The annotated CDFG is further processed by the Hyper hardware mapping tool. The output of the hardware mapper is an RT-level structural VHDL description of the design. The VHDL description is translated using the BETS translator to the Logic III format, which is the input to the OASIS system. A gate-level netlist (ISCAS89 format) and physical layout are produced using the OASIS system. The gate-level netlist is used as the input to the HITEC sequential ATPG tool [11].

To evaluate the proposed transformation methodology, objective function and optimization algorithms, HITS was applied to the behavioral descriptions of the following designs: 4IIR - 4th order IIR filter, 5WDF - 5th order elliptical wave digital filter, LIN3 - 5th order linear controller, ELLIP4 - GE controller, and 8IIR - 8th order Avenhaus IIR filter.

Table 1 shows the results of applying HITS and BETS to the designs. In the sequel, *BETS* refers to the implementation of the original CDFG using BETS when both testability and area are considered during allocation, scheduling, and assignment. *HITS* refers to the implementation obtained by transforming the original CDFG using HITS, and synthesizing the transformed CDFG using BETS. The numerical suffix after the name of design indicates the word-length of the implementation. In all examples, except the last one, the *BETS* and the *HITS* design required the same number of bits. For the 8IIR example the word-length reduces after the application of transformations from 14 to 12 bits. The columns *Hardware* and *Sample Period* shows the number of execution units (multipliers, adders, and subtracters), and the number of clock cycles needed for both the *BETS* and *HITS* designs. Note that while the number of clock cycles taken by both the initial and final designs are maintained the same, the hardware needed by the final design is always less than the corresponding initial designs.

Table I: Experimental result of the application of the HITS system on 5 benchmark examples.

Design Name	Design Version	Hardware	Sample Period	Scan/Tot FF	Faults	F.C (%)	T.E. (%)	ATPG Time (sec)
4IIR.20	BETS	2M,2A,2S	5	80/340	9716	96	100	224.9
	HITS	2M,1A,1S	5	20/280	8936	95	99	503.6
5EWF.20	BETS	3M,2A	17	60/460	10916	98	100	233.2
	HITS	2M,2A	17	20/340	7023	99	100	128.1
LIN3.16	BETS	5M,3A	13	144/400	14308	99	100	193.6
	HITS	3M,2A	13	16/224	12157	95	99	4221.5
ELLIP4.24	BETS	5M,4A	8	120/480	12986	99	100	162.2
	HITS	3M,2A	8	24/336	8478	98	100	146.9
8IIR.14	BETS	2M,3A,1S	34	182/462	9964	100	100	135.1
	HITS	1M,1A	34	60/132	6233	99	100	187.7

The column *Scan/Tot FF* reports the number of scan FFs needed by BETS to synthesize a loop-free testable circuit, and the total number of flip-flops, for the *BETS* and *HITS* implementations. The number of scan FFs needed by BETS for the transformed CDFGs produced by HITS (*HITS*) is significantly smaller than the number of scan FFs needed by BETS for the original CDFGs (*BETS*). The reduction in the number of scan FFs ranges from a factor of 3 (5EWF) to a factor of 9 (LIN3). The average and median reductions are by factors of 4.81 and 4.00 respectively.

The last four columns of Table 1. demonstrate that the designs obtained using BETS and HITS are consistently highly testable using the gate-level sequential ATPG tool HITEC [11]. The total number of faults (column *Faults*), the fault coverage obtained (column *FC*), the test efficiency obtained (column *TE*), and the ATPG time taken in seconds on a SUN Sparcstation2 (column *ATPG Time*) are shown in Table 1. Fault coverage, Test Efficiency, and ATPG times remain close for both initial and final designs. Only in the case of LIN3, the test generation time was significantly higher for the final design than for the initial design, by a factor of more than 20 times, which can be attributed to the sharp drop in the number of scan FFs used for the final implementation. Note that the *HITS* design needed significantly smaller number of scan FFS than the *BETS* design to achieve comparable high test efficiencies.

VIII Conclusion

We introduce a transformation-based approach for simultaneous optimization of testability and area. Experimental results demonstrate significant savings in partial scan overhead when the original specifications are transformed by HITS before using the behavioral test synthesis system BETS [4] to synthesize 100% testable designs.

IX References:

[1] L.J. Avra, E.J. McCluskey: "High Level Synthesis of Testable Designs: An Overview of University Systems", ITC Test Seminar 1994.

[2] A.P. Chandrakasan et. al.: "Hyper-LP: A Design System for Power Minimization using Architectural Transformations", ICCAD, pp. 300-303, 1992.

[3] 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.

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

[5] S. Dey, et al.: "Synthesizing Designs with Low-Cardinality Minimum Feedback Vertex Set for Partial Scan Application", pp. 2-7, VLSI Test Symposium, 1994

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

[7] Z. Iqbal, M. Potkonjak, S. Dey, A. Parker: "Critical Path Minimization Using Retiming and Algebraic Speed-Up", 30th ACM/IEEE DAC, pp. 573-577, June 1993.

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

[9] C.E. Leiserson, J.B. Saxe, "Retiming Synchronous Circuitry", Algorithmica, Vol. 6., No. 1, pp. 5-35, 1991.

[10] 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.

[11] T.M. Niermann, J. H. Patel: "HITEC: A Test Generation Package for Sequential Circuits", EDAC, pp. 214-218, 1991.

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

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

[14] 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, 1991.

[15] M. B. Srivastava, M. Potkonjak: "Transforming Linear Systems for Joint Latency and Throughput Optimization", EDAC-94, pp. 267-271, 1994.