SPARTAN: A Spectral and Information Theoretic Approach to Partial-Scan

Omar I. Khan Michael L. Bushnell Suresh K. Devanathan Vishwani D. Agrawal
{*okhan, bushnell, mdsuresh*}*@caip.rutgers.edu* *vagrawal@eng.auburn.edu*
Dept. of ECE and CAIP Research Center, Rutgers University Dept. of ECE, Auburn University
96 Frelinghuysen Road, Piscataway, NJ 08854-8088 Auburn, AL 36849

Abstract*—We propose a new partial-scan algorithm, the first to use toggling rates of the flip-flops (analyzed using DSP methods) and Shannon entropy measures of flip-flops to select flip-flops for scan. This improves the testability of the* circuit-under-test *(CUT). Entropy is maximized throughout the circuit to maximize the information flow (the* principle of maximum entropy*), which improves testability. We propose using partial-scan for testing, to maximize* fault coverage *(FC), reduce* test volume *(TV), reduce* test application time *(TAT), and reduce* test power *(TP) but we allow for full-scan during silicon debug. Full-scan is commonly used for testing, to reduce sequential* automatic test-pattern generation *(ATPG) to the complexity of combinational ATPG, but comes with serious TV, TAT, and TP overheads. Partial-scan significantly reduces circuit delay, when compared to full-scan, because critical flip-flops in the circuit data path do not have the extra hardware for full-scan, and therefore are roughly 5% faster, and use 10% less area. This is particularly critical for microprocessors. The HITEC ATPG program generated results for this new partial-scan algorithm [1].*

I. Introduction

Rapid advancements in chip fabrication technology allow hundreds of millions of transistors on a single VLSI chip, at the cost of increased testing difficulty. An Intel Pentium 4 microprocessor has 55 million transistors. The testing cost is alleviated by improving the circuit testability, using *design-for-testability* (DFT) techniques. Logic DFT consists of *ad-hoc* and structured techniques. Structured DFT is further split into scan (full and partial) and *test point insertion* (TPI) methods.

Most complex circuits today employ full-scan to improve testability because it reduces the ATPG problem to a combinational one and full-scan designs are very useful for silicon debug. Also, scan chain insertion and test generation can be fully automated, and almost always require little manual DFT. Disadvantages of using full-scan are long scan shifting sequences, over testing, and higher power consumption. Long scan chains need long shift sequences, which consume much time because scan-in and scan-out of shift sequences is done at a lower clock rate to avoid circuit burnout. Using full-scan also causes the detection of redundant faults, because in a fully scanned circuit, *any* arbitrary state can be achieved, but in normal operation, only a tiny fraction of these states are legal functional states [2]. This can cause unnecessary rejection of otherwise fault-free circuits, resulting in yield loss. We ran SEST [3], a deterministic sequential ATPG, on benchmark s9234 (with 6927 total faults) and it reported 6909 untestable faults in s9234. We then ran the wavelet sequential ATPG [4] on s9234 with full-scan and the ATPG detected 6475 faults (93.47% fault coverage), thus over testing the circuit. The remaining 452 faults are redundant in full-scan mode, but of the 6475 faults, only 18 faults need to be tested in functional mode [5]. Since we are shifting sequences through all flip-flops, the circuit in test mode can consume up to 200% of the normal circuit power [6].

Partial-scan mthods can be categorized as: structure based [7–11], testability-measure based [12], and test-generation based [13] methods. Of the three, structure based methods have been most successful. Some partial-scan algorithms use a combination of testability measures, structural information, and test-generation information [14]. However, only some of these methods have been used, because of inadequate fault coverage, severe problems and long computation times for sequential ATPG, and the need, therefore, to repeat partial-scan insertion many times. This leads to a number of cycles of partial-scan, sequential ATPG, and ad hoc testability insertion, which takes too long compared with full-scan and combinational ATPG. Also, most of the prior partial-scan methods have not emphasized high fault coverage, but rather selected the minimal number of flip-flops to scan. The industry prefers full-scan and combinational ATPG, because high fault coverage and rapid ATPG are more important to them than scanning fewer flip-flops. For the present, the comparison metric is full-scan and combinational ATPG. Therefore, we abandon the focus of the prior work on selecting the minimal number of flip-flops, and instead focus on maximizing the fault coverage, and minimizing TV and TAT. This means that we will compare our results in this work to *mpscan* [12].

Because of the success of spectral sequential ATPG tools, we propose a spectral partial-scan algorithm, SPARTAN, which dissects the sequential circuit states in the spectral domain. Spectral sequential ATPG tools have attained the highest fault coverages on sequential benchmark circuits, with drastically shorter test sequences in drastically less CPU time than deterministic sequential ATPG tools. SPARTAN uses spectral analysis and entropy analysis for *scan flip-flop* (SFF) selection by logic simulation with an analysis of the CUT structure. SPARTAN's results are compared to Xiang and Patel's partial-scan algorithm, *mpscan* [12], because *mpscan* has higher FC and shorter *test vector length* (TL) compared to prior partial-scan algorithms that use only structural information and testability analysis. SPARTAN achieved a higher FC (an average of 97.6%) compared to *mpscan* (an average of 96.7%). The average TV achieved by SPARTAN is 349 Kbits and the average TAT is 327971 clock cycles.

The paper is organized as follows. Section II reports prior

1-4244-1128-9/07/$25.00

work on information theory, information theoretic approaches to testing, and partial-scan. Section III describes SPARTAN. Section IV provides results for *ISCAS-89* benchmarks. Section V concludes and Section VI discusses future work.

II. Prior Work

A. Entropy and Testing

Information theory is useful in combating noise-related errors of communication. Information theoretic measures such as entropy also apply to physics (statistical mechanics), mathematics (probability theory), electrical engineering (communication theory), and computer science (algorithmic complexity). Equation 1 shows the general definition of entropy for a *random variable* (RV) X with a *probability mass function* (PMF) $p(x)$. A PMF is the discrete version of a *probability density function* (PDF). Variable x varies over all possible values of X.

$$H(X) = -\sum_{x \in X} p(x)\ \log_2(p(x)) \tag{1}$$

Entropy is the number of bits (information), on average, required to encode a RV.

EXAMPLE 1 – Consider a RV X that has a uniform distribution and has 32 equiprobable outcomes. The entropy of X is:

$$H(X) = -\sum_{x=1}^{32} p_x\ \log_2(p_x) = -32 \times (\frac{1}{32}\log_2(\frac{1}{32})) = 5\ bits \tag{2}$$

which agrees with the number of bits needed to describe X.

For logic circuits, each line is a RV with 2 possible outcomes – logic 0 and logic 1. Entropy of a wire is:

$$H(X) = -((1-p)\log_2(1-p) + p\ \log_2(p)) \tag{3}$$

where p is the probability of logic 1 occurring at a line. Equation 3 is plotted in Figure 1 with $H(X = p)$ on the ordinate and p on the abscissa. Maximum entropy occurs when 0 and 1 are equally likely.

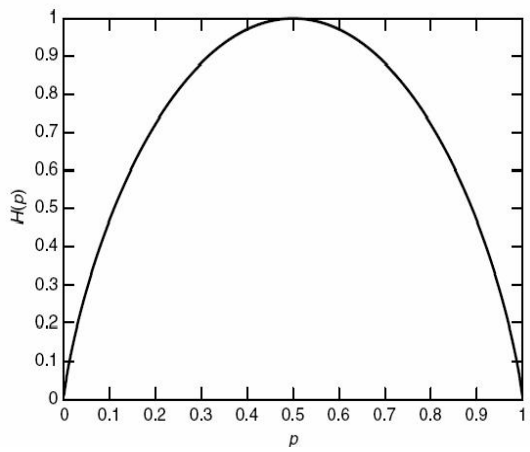

Figure 1: $H(p)$ vs. p.

Dussault proposed the first information theoretic testability measure [15]. *Mutual information* (MI), $I(X;Y)$, in Equation 4 is a measure of dependence between two RVs. $H(X|Y)$ is the *conditional entropy* of a RV X if we know the value of a second RV Y (see Equation 5) [16]. Equation 6 gives $I(X;Y)$, the reduction in the uncertainty of X due to the knowledge of Y. $I(X;Y)$ is 0 if X and Y are independent RVs. For binary logic, $X, Y \in (0,1)$. Equations 7 – 9 show the proposed testability measures. Circuit inputs (outputs) are represented by the RV vector X (Y).

$$I(X;Y) = \sum_{\substack{x \in X \\ y \in Y}} p(x,y) log_2\ \frac{p(x,y)}{p(x)p(y)} = I(Y;X) \tag{4}$$

$$H(X|Y) = \sum_{y \in Y} p(y) H(X|Y=y) \tag{5}$$

$$I(X;Y) = H(X) - H(X|Y) \tag{6}$$

$$Observability = \frac{1}{H(X|Y)} \tag{7}$$

$$Controllability = \frac{1}{H(Y|X)} \tag{8}$$

$$Testability = I(X;Y) \tag{9}$$

$H(X|Y)$ gives the uncertainty in inputs X given that outputs Y are known. The less the uncertainty, the greater the amount of information about inputs reaching the outputs, implying higher observability. Equation 7 gives the observability since observability and uncertainty $H(X|Y)$ are inversely proportional. Similarly, Equation 8 gives the controllability. Overall circuit testability is given by $I(X;Y)$, where large values indicate high testability.

Agrawal proposed an information theoretic approach to testing digital circuits [17] and derived the probability $P(T)$ of detecting a stuck-at fault by a vector sequence T as:

$$P(T) = 1 - 2^{-\frac{H_o T}{k}} \tag{10}$$

where k is the number of lines through a circuit partition where the detectable fault exists and H_o is the entropy at the output of the circuit. Consider a 2-input AND gate with inputs $i1$, $i2$, and output Z. If the probability of logic 0 (logic 1) occurring at the inputs is 0.5 (0.5), the entropies at the inputs, $i1$ and $i2$, are:

$$H_{i1} = H_{i2} = -0.5\ \log_2(0.5) - 0.5\ \log_2(0.5) = 1.0 \tag{11}$$

Therefore, the total information present at the inputs is $1+1=2$. The probability of logic 0 (logic 1) at output Z is 0.75 (0.25) and the entropy of Z is:

$$H_Z = -0.25\ \log_2(0.25) - 0.75\ \log_2(0.75) = 0.811 \tag{12}$$

So, the AND gate has information loss of $2.0 - 0.811 = 1.189$. Agrawal proposed an ATPG method that reduces the loss (by increasing entropy) of information and maximizes $P(T)$ in the circuit by adjusting the probabilities of 0 and 1 at the inputs.

Thearling and Abraham proposed information theory based testability measures at the functional level [18] and partitioned the circuit, to improve testability, using entropy measures.

B. Partial-scan and DFT

Williams and Angell [19] first incorporated full-scan into a circuit for testing. Agrawal *et al.* selected a near optimal subset of scan flip-flops [20]. The method tried to achieve a required test coverage with minimum overhead through the use of a modified *path-oriented decision making* (PODEM) automatic test pattern generation program [21]. Cheng and Agrawal [8] presented

graph-theoretic algorithms to select a minimal set of flip-flops for eliminating feedback cycles and reducing sequential depth. Bhawmik *et al.* based the PASCANT algorithm for selection of flip-flops on identification and elimination of *strongly connected components* (SCCs) in a circuit graph [22].

Chakradhar *et al.* [7] solved the *minimum feedback vertex set* (MFVS) problem for partial-scan with branch-and-bound partitioning and pruning techniques using *integer linear programming* (ILP). Table 1 shows the number of flip-flops that were selected by different algorithms. Columns *LR*, *Pa*, *Op*, and *PSCAN* report data obtained from the Lee and Reddy [23], PASCANT [22], Opus [24], and PSCAN algorithms.

Table 1: Partial-scan results.

Circuit	FF	Scan Flip-Flops				CPU Sec.		
		LR	Pa	Op	PSCAN	Pa	Op	PSCAN
s953	29	5	5	5	5	0.0	0.1	0.1
s1196	18	0	0	0	0	0.0	0.4	0.0
s1238	18	0	0	0	0	0.0	0.4	0.0
s1423	74	21	37	22	21	0.3	1.0	0.9
s1494	6	5	5	5	5	0.0	0.5	0.1

Xiang and Patel [12] presented a multi-phase partial-scan algorithms (*opscan* and *mpscan*). Partial-scan is divided into a *critical cycle breaking* phase and a *partial flip-flop selection with respect to conflict resolution* phase.

Trischler [25] introduced a simple testability measure to select scan flip-flops, which is the first testability-analysis based partial-scan method. Lin *et al.* used test point insertion to scan combinational logic paths [26]. Cheng and Lin devised a timing-driven test point insertion method for full and partial-scan *built-in self-testing* (BIST) schemes [27].

Abadir and Breuer created the *I-path* to analyze an RTL circuit description to introduce DFT structures [28]. An I-path is a data channel over which test data flows. It is a subcircuit with n-bit-wide input and output data ports and control lines such that data on the input port are reproduced unchanged (but possibly delayed) on the output port when the control lines are appropriately exercised. Buses, MUXes, and parallelin/parallel-out registers have I-paths, as do cascaded combinations of them.

C. Spectral Techniques

Darmala and Karpovksy have proposed fault detection techniques in combinational circuits using spectral transforms [29, 30]. In many fields such as communications, digital signal processing, etc., analysis in the spectral (frequency) domain provides a better intuition of the signal properties than analysis in the time domain. Spectral techniques have also been used for fault detection [31] and sequential ATPG [4, 32].

III. The SPARTAN Algorithm

The algorithm analyzes the CUT and selects SFFs using three measures: spectral analysis of the flip-flop oscillations, entropy from information theory, and logic gate circuit level information to measure observability and controllability. A high quality SFF set will enhance the testability of the CUT, which allows the ATPG method to attain high fault coverage with a shorter test set. We use the spectral analysis and entropy combination because spectral analysis incorporates only functional information and does not use any structural information of the CUT.

A. Spectral Analysis

A.1 *Data Structure*

SPARTAN uses an *s-graph*, the logic circuit graph, and logic simulation. In a sequential circuit, each s-graph node represents a flip-flop and an edge between two nodes is a path through combinational logic between the flip-flops. When the s-graph is condensed, each node of a condensed s-graph is called an SCC.

A.2 *Spectral Algorithm*

Overview. Because of the success of spectral ATPGs, we propose to dissect the function and structure of sequential circuit states in the spectral domain. The *spectral coefficients* (SCs) for each flip-flop in the CUT, which are coefficients of the flip-flop states in the spectral domain, are used as a controllability measure. A low toggling rate flip-flop will have small SC values and thus is difficult to control. If the average value of the SCs of a flip-flop is low (low toggling), the flip-flop has low controllability and will be a good SFF candidate (see Example 3). A low toggling flip-flop has bad controllability because regardless of which vectors are used, then the ATPG will have to apply many vectors to set the flip-flop to a desired state in order to observe or sensitize faults. To compute observability, we first find for each flip-flop the *primary outputs* (POs) in its fanout cone. We then compute the SCs of all POs. Then we perform a SC comparison of each flip-flop with each PO in its fanout cone. If the SCs of a flip-flop are identical to SCs of any one of POs in its fanout cone, we consider that flip-flop observable. We employ an order 4 *Radamacher-Walsh transform* (RWT) (an order 3 RWT is shown in Equation 13) for analyzing states of flip-flops and POs. First, the state machine is initialized to a randomly-chosen state. Flip-flop states and PO logic values are obtained via logic simulation with 50K random vectors [31]. Procedure 1 shows the spectral analysis and Figure 3 presents the flow chart of the spectral analysis. Please note that *spectral correlation coefficient* and SC will be used interchangeably.

Spectral Correlation Coefficients. The *spectral correlation coefficients*, for each flip-flop, are obtained by dividing the flip-flop state over the last n clock periods into chunks (the state vector is divided into smaller column vectors), $\mathbf{Z}$, and pre-multiplying each chunk with the RWT of appropriate dimension (see Equation 14). $\mathbf{S}$ is the SC vector. The chunk size for the flip-flop states depends on the dimension of the RWT used. We use a 16×16 matrix for our analysis. Matrix dimensions up to 512×512 were tried, but larger matrices had inconsequential effect on the quality of the SFF set. Also, it takes less time to perform spectral analysis using a 16×16 transform. Since the transform entries $\in \{+1, -1\}$, we have to translate the flip-flop states to $+1$ or -1. We translate $0 \rightarrow -1$ and $1 \rightarrow +1$ because when we calculate the spectral coefficients (a SC is the number of agreements minus the number of disagreements with the row elements of the transform matrix), logic 0 and logic 1 should be given equal and opposite magnitude. This ensures that both logic 0 and logic 1 are weighted equally and provide a contribution in the SCs. For example, if we have a vector with logic 0's and logic 1's and if we extract spectra (using any matrix transform) from this vector, the logic 0's in the vector will have no contribution in the SCs.

Figure 2 shows the wave representation of the order 3 (8×8) RWT. When a bit stream is analyzed using an order 3 transform, each coefficient tells us how well the bit stream correlates with each wave in Figure 2 (or matrix row in Equation 13).

$$RWT(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad (13)$$

$$RWT(3) \times \begin{bmatrix} \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{S} \end{bmatrix} \quad (14)$$

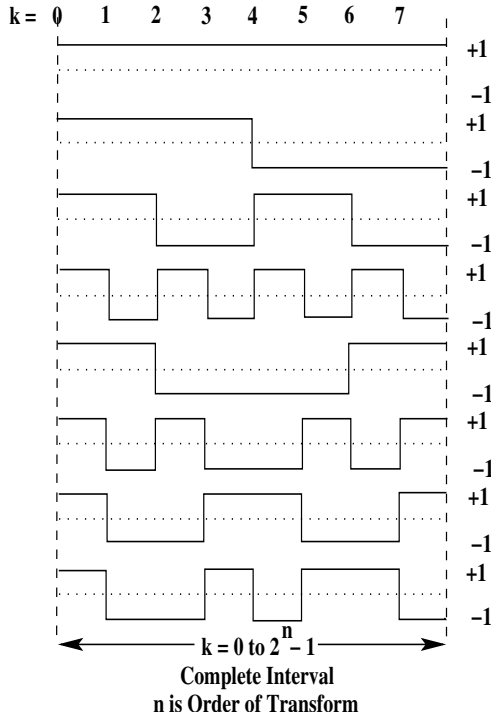

Figure 2: Orthogonal Radamacher-Walsh functions for $n = 3$.

EXAMPLE 2 – In this example we will use the 8×8 RWT for analysis. To compute the first SC (s_0), row 1 of the 8×8 RWT is first multiplied with column vector **Z**, comprised of the flip-flop state over the last 8 clock periods, to obtain the SC with the first 8 states of the flip-flop. This 8 bits of state is then multiplied by rows 2 through 8 of RWT(3) to obtain the remaining 7 SCs, $s_1 - s_7$, with the RWT basis vectors. RWT(3) is then multiplied with bits 2 to 9, bits 3 to 10, and so on of the flip-flop states. The partial SCs of the flip-flop obtained, after each row of the RWT is multiplied with the state over the last 8 clock periods, are aggregated and normalized by the number of times multiplication was performed by row 1 to obtain the first SC s_0. Example 3 elaborates the spectral analysis procedure. Multiplication of a row with a chunk **Z** is the calculation of the correlation of the data vector with that matrix row. Here the data vectors are the first row of the RWT and a state chunk **Z**.

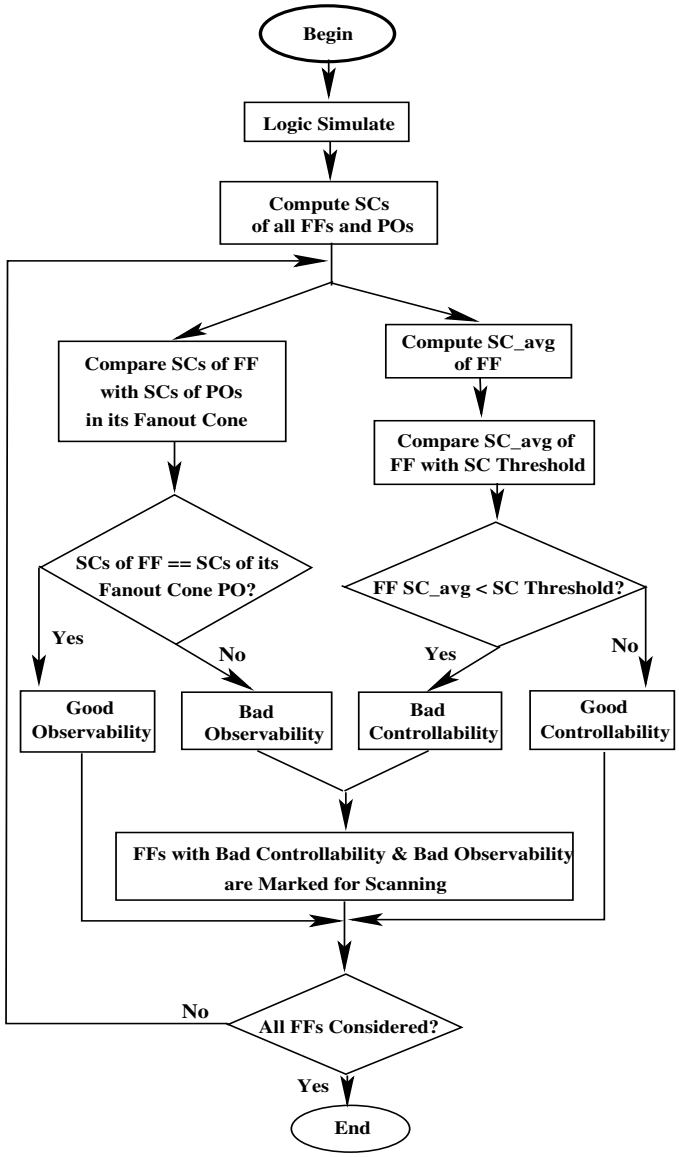

Figure 3: SPARTAN's spectral SFF selection procedure.

EXAMPLE 3 – We use the RWT in Equation 13 for spectral analysis of a flip-flop F. After ten random vectors at *primary inputs* (PIs) of the CUT, F goes through states $V = [1, 0, 1, 1, 0, 1, 1, 0, 0, 0]$. After translation, we get $V' = [1, -1, 1, 1, -1, 1, 1, -1, -1, -1]$. To compute s_0, we multiply row 1 of the RWT with translated bits 1-8, 2-9, and 3-10 of V'. For bits 1-8 the partial SC is $(1 \times 1) + (1 \times -1) + (1 \times 1) + (1 \times 1) + (1 \times -1) + (1 \times 1) + (1 \times 1) + (1 \times -1) = 2$. For bits 2-9 the partial SC is $(1 \times -1) + (1 \times 1) + (1 \times 1) + (1 \times -1) + (1 \times 1) + (1 \times 1) + (1 \times -1) + (1 \times -1) = 0$ and bits 3-10 give us $(1 \times 1) + (1 \times 1) + (1 \times -1) + (1 \times 1) + (1 \times 1) + (1 \times -1) + (1 \times -1) + (1 \times -1) = 0$. The aggregate of partial sums is $2 + 0 + 0 = 2$ and after normalization we get $s_0 = \frac{2}{3}$. Normalization is by 3 because there were 3 multiplication operations of row 1 and three 8-bit chunks (**Z**) of V'. This flip-flop will have 8 SCs. Similarly, SCs are computed for other flip-flops.

Procedure 1 – Spectral Analysis (see Figure 3):

1. Logic simulate the CUT with 50K random vectors to obtain valid FF states and PO values.

2. Compute SCs of all FFs and POs.
3. Set $SC_{avg\_MAX} = 0$ and $SC_{avg\_MIN} = 0$.
4. For each flip-flop:
   (a) Compute the average of SCs, SC_{i_avg}, for flip-flop i with $k = 16$ using Equation 15.
   (b) If $SC_{i\_avg} \geq SC_{avg\_MAX} \Rightarrow SC_{avg\_MAX} = SC_{i\_avg}$.
   (c) If $SC_{i\_avg} \leq SC_{avg\_MIN} \Rightarrow SC_{avg\_MIN} = SC_{i\_avg}$.
   (d) Compare SCs of FF i with SCs of POs its fanout cone.
      i. If SCs of FF i == SCs of any fanout cone PO $\Rightarrow$ $Good\_Observability(i) = FALSE$.
5. Compute the "SC threshold" using Equation 16.
6. For each flip-flop:
   (a) If $SC_{i\_avg} < SC\ Threshold$ && $Good\_Observability(i) == FALSE \Rightarrow scan\_mark(FF(i)) = TRUE$.

$$SC_{i\_avg} = \frac{\sum_{j=1}^{k} SC_j}{k} \quad (15)$$

$$SC\ Threshold = \text{average}(\ SC_{avg\_MIN} + SC_{avg\_MAX}) \quad (16)$$

□

After Procedure 1 (Figure 3), spectral analysis gives a list of candidate flip-flops recommended for scanning. We will scan only a subset of this list. If a flip-flop only has paths to it from other flip-flops, we automatically scan it. Next, we set the *scanning threshold* (ST) to 25% of the total FFs in the circuit. This gives us the size of the subset (of initial candidate FF list from Procedure 1) that will eventaully be scanned. ST can be varied and 25% gave the best results. Once ST is set, we start scanning FFs from the list of FFs marked as candidates for scanning by Procedure 1, starting with FFs belonging to the largest SCC because it will help us break bigger cycles in the circuit. Once all the marked FFs in a SCC are scanned, the SCC is marked as 'visited' and cannot be revisited. If the number of scanned FFs is less than ST, we move to subsequent SCCs of the circuit. The procedure is Procedure 1a (see Figure 4).

Procedure 1a – SFF Selection After Spectral Analysis:

1. Number of SFFs, $numScanFF = 0$.
2. Largest SCC size, $maxSCCSz = 0$.
3. Each SCC's scan flag, $SCC(i)\_scanned = FALSE$.
4. while $numScanFF < ST$
   (a) For each SCC i:/*Locate largest SCC not visited yet.*/
      i. If $SCC(i)\_scanned$ == $FALSE$
         A. If $SCC(i)\_size > maxSCCSz \Rightarrow maxSCCSz = SCC(i)\_size$ and $SCC\_max = SCC(i)$.
   (a) For each flip-flop, $FF(i)$, in SCC_max:
      i. If $scan\_mark(FF(i)) == TRUE$
         A. $scan(FF(i)) = TRUE$ and $numScanFF++$.
         B. If $numScanFF > ST \Rightarrow$break for-loop (Step 4a).
   (b) $SCC(i)\_scanned = TRUE$.

A.3 *Computational Complexity*

Complexity of Procedure 1. Assume that the cut has N logic gates, L lines, F flip-flops, P POs, and P_{FO} POs in a flip-flop's fanout cone. Also, the CUT is logic simulated using V vectors. To find SCCs, we need to perform *depth-first search* (DFS) on the s-graph. The complexity if DFS is $O(N+L)$. The complexity of Step 1 is $O(VN)$. Step 2 has a cost of $O((F+P) \times (V - (16-1)) \times 256)$. The loop in Step 4 compares each flip-flop's SCs with SCs of all POs in its fanout cone. Since there are 16 coefficients for each PO and P_{FO} POs in each flip-flop's fanout cone, we get a total complexity of $O(F \times (16 \times P_{FO}))$. The complexity of the Step 6 loop is $O(F)$. Steps 3 and 5 use constant time. So, the total computational complexity of Procedure 1 is approximately $O((N+E)+VN+(F+P)V+F \times P_{FO})$.

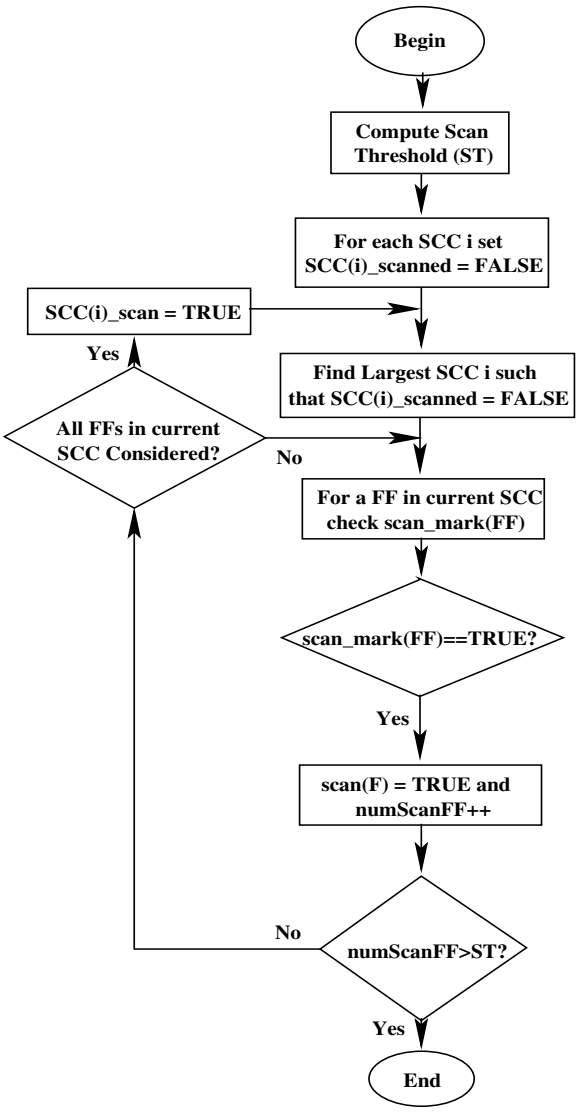

Figure 4: Procedure 1a – SPARTAN's spectral SFF selection.

Complexity of Procedure 1a. Assume that there are S SCCs in the circuit, F_{SCC} flip-flops in each SCC, and ST is the scan threshold. The Step 4 loop's complexity in Procedure 1a is $O(ST(S+F_{SCC}))$.

B. Entropy Analysis

Agrawal's Work. Agrawal showed that the testability of a circuit can be improved by increasing entropy at the POs [17]. SPARTAN uses entropy H as a testability measure to select scan flip-flops such that the entropy of unscanned flip-flops in the CUT is enhanced. We conjecture that increasing the entropy of a line l in the circuit is akin to increasing the controllability of l. Figure 1 shows that entropy is maximum when logic 0 and logic 1 are equally likely, i.e., the probabilities of logic 0 ($p(0)$) and logic 1

($p(1)$) occurring are equal ($p(0) = p(1) = 0.5$). Then, controlling l to 0 or 1 from PIs should become easier, thus increasing l's controllability. The entropy analysis uses random-pattern testability analysis since $p(1)$ and $p(0)$ are required to compute entropy [33]. Using entropy analysis is analogous to using Fourier analysis. Given the time domain data, Fourier analysis gives the spectral components of the data. Similarly, $p(0)$ and $p(1)$ of each line in the circuit are like time domain data and the entropy analysis gives us the amount of information on each line.

Cycle Breaking. Breaking cycles in the s-graph of the CUT improves testability [7, 8, 34, 35]. Entropy analysis uses the SCC information for selecting the best scan candidates because SCCs capture direct and indirect paths from a flip-flop to every other flip-flop in the same SCC. Each flip-flop in a SCC lies on at least one cycle. Flip-flops from different SCCs cannot lie on any cycle together because if these two flip-flops were reachable from each other, they would be included in the same SCC and not different SCCs. Therefore, we perform entropy analysis on one SCC at a time in an effort to break cycles (since flip-flops from two distinct SCCs cannot be in any cycle together) in all the SCCs and improve the CUT's testability.

Probability Calculation. We use logic simulation to compute the probabilities, which are then used to compute flip-flop entropies. *Parker and McCluskey's probability expressions* (PMEs) to compute probabilities can also be used [33]. Even though PMEs provide a quick method for approximating probability values, we used probability values obtained via logic simulations because they are more accurate. Table 2 shows the probability and entropy values, of benchmark s820, for the deterministic method (PME) and logic simulation. The first column is the flip-flop number, the second is the probability, $p_{sim}(1)$, of a logic 1 obtained via logic simulation and the third is the corresponding entropy value H_{sim}. Similarly, the fourth column is the probability, $p_{det}(1)$, of a logic 1 obtained using PMEs [33] and the fifth column is the corresponding entropy value H_{det}. The sixth column is the difference in entropy values for the two methods. Entropy values for the two methods are not identical because PMEs assume that each line in the CUT is independent, which is not accurate. Since logic simulation accounts for correlation among signal lines in the CUT, probability values obtained via logic simulation are more accurate.

Table 2: Experimental and deterministic values for probability of logic 1 ($p(1)$) and entropy (H) for s820.

DFF	Simulation		Deterministic		% Entropy
	$p_{sim}(1)$	H_{sim}	$p_{det}(1)$	H_{det}	Difference
1	0.329	0.914	0.272	0.844	7.66
2	0.036	0.222	0.019	0.136	38.74
3	0.037	0.227	0.046	0.267	17.62
4	0.044	0.260	0.076	0.387	48.85
5	0.137	0.576	0.106	0.487	15.45
Avg.		0.440		0.424	25.66

Entropy for Scan Flip-Flop Selection. First, we initialize the state machine to a random state, and we logic simulate it using 50,000 random vectors to compute the probabilities. After all of the probabilities are computed, they are used to compute the entropy, $H_i(Q)$, using Equation 3 and conditional entropies, $H_i(D|PI_j)$, and $H_i(Q|PO_k)$, using Equation 5 for each flip-flop i. $H_i(Q)$ is the information at flip-flop i. $H_i(D|PI_j)$ is the information (controllability) on flip-flop i's input given a logic value (0 or 1) on the j^{th} PI, PI_j, in its fanin cone. Similarly, $H_i(Q|PO_k)$ is the amount of information (observability) on flip-flop i's output given a logic value on the k^{th} PO, PO_k, in its fanout cone. The number of controllability values ($H_i(D|PI)$) for flip-flop i depends on the number of PIs in its fanin and the number of observability values ($H_i(Q|PO)$) depends on the number of POs in its fanout. Then, for each flip-flop i, we compute the average controllability and observability values using Equations 17 and 18, where p (q) is the number of PIs (POs) in the fanin (fanout) cone of flip-flop i. We maximize the information flow through the circuit and unscanned flip-flops. Increasing the information flow in the circuit is analogous to increasing the testability.

$$H_{i\_avg}(D|PI) = \frac{\sum_{x=1}^{p} H_i(D|PI_x)}{p} \quad (17)$$

$$H_{i\_avg}(Q|PO) = \frac{\sum_{x=1}^{q} H_i(Q|PO_x)}{q} \quad (18)$$

So far we have $H_i(Q)$, $H_{i\_avg}(D|PI)$, and $H_{i\_avg}(Q|PO)$ values for each flip-flop i. Now we need average values $H(Q)_j$, $H(D|PI)_j$, and $H(Q|PO)_j$ for any SCC_j. Equations 19, 20, and 21 compute the average entropy values for any SCC_j where m is the number of nodes in SCC_j.

$$H(Q)_j = \frac{\sum_{i=1}^{m} H_i(Q)}{m} \quad (19)$$

$$H(D|PI)_j = \frac{\sum_{i=1}^{m} H_{i\_avg}(D|PI)}{m} \quad (20)$$

$$H(Q|PO)_j = \frac{\sum_{i=1}^{m} H_{i\_avg}(Q|PO)}{m} \quad (21)$$

Similarly, other SCCs' average entropies are computed. For each SCC (of size > 1), the algorithm checks to see which flip-flops in the SCC, if scanned, will cause an increase in the average entropy values of the SCC. It does so by making the flip-flop's input a *pseudo-primary output* (PPO) and its output a *pseudo-primary input* (PPI). It then performs logic simulation to to recompute the probabilities of the circuit. This flip-flop is marked as 'tried.' Initially all flip-flops are marked as 'not tried.' Since any flip-flop in a SCC has a direct or indirect path to every other flip-flop in that SCC, scanning a flip-flop will affect the probabilities of all of the unscanned flip-flops in that SCC and possibly other SCCs.This means that scanning a flip-flop in SCC_j will change the entropies of the unscanned flip-flops and thus the average entropy of the SCC_j will change. The new average entropy values of SCC_j become $H(Q)'_j$, $H(D|PI)'_j$, and $H(Q|PO)'_j$. The changes in average entropy values are given in Equations 22, 23, and 24.

$$\triangle H(Q)_j = H(Q)'_j - H(Q)_j \quad (22)$$

$$\triangle H(D|PI)_j = H(D|PI)'_j - H(D|PI)_j \quad (23)$$

$$\triangle H(Q|PO)_j = H(Q|PO)'_j - H(Q|PO)_j \quad (24)$$

If any one of the average entropy values ($H(Q)'_j$, $H(D|PI)'_j$, and $H(Q|PO)'_j$) of SCC_j, after selecting the flip-flop (currently being considered for scanning), increases at all, the flip-flop is marked

as a good candidate for scanning because it increased the controllability and observability of other flip-flops in the SCC. The remaining flip-flops in the SCC are then 'tried' for scanning, one at a time, and the ones that increase the average entropy values of the SCC are marked as good choices for scanning. After all of the flip-flops have been tried for scanning, the flip-flops that were marked for scanning are scanned (and converted into *pseudo-primary inputs* (PPIs) and PPOs). Procedure 2 is the entropy analysis algorithm (flowchart in Figure 5).

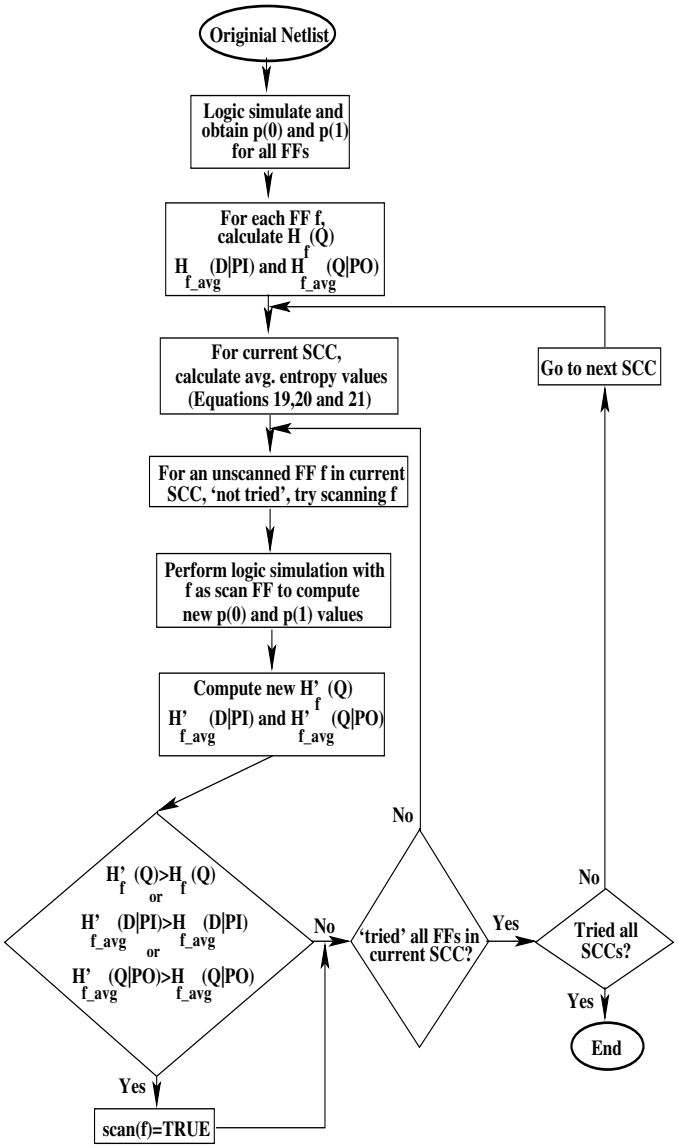

Figure 5: Procedure 2 – SPARTAN's entropy SFF selection.

Procedure 2 – Entropy Analysis:

1. Logic simulate to compute values of $p(0)$ and $p(1)$ for all flip-flops.
2. For each flip-flop f:
   (a) Use $p_f(0)$ and $p_f(1)$ to calculate entropy values $H_f(Q)$, $H_{f\_avg}(D|PI)$, and $H_{f\_avg}(Q|PO)$.
3. For each SCC S:
   (a) Calculate average entropy values of SCC using Equations 19, 20, and 21.
      i. For each unscanned and 'not tried' flip-flop f in the current SCC:
         A. Try scanning f (mark it 'tried') by converting its input (output) into a PPO (PPI). Perform logic simulation and compute new probabilities.
         B. Backup S's original average entropy values $H(Q)_S$, $H(D|PI)_S$, and $H(Q|PO)_S$. Then compute S's new average entropy values $H(Q)'_S$, $H(D|PI)'_S$, and $H(Q|PO)'_S$ from updated probability values.
         C. If $\frac{H(Q)'_S - H(Q)_S}{H(Q)_S} > 0.0$ || $\frac{H(D|PI)'_S - H(D|PI)_S}{H(D|PI)_S} > 0.0$ || $\frac{H(Q|PO)'_S - H(Q|PO)_S}{H(Q|PO)_S} > 0.0$ by scanning f, 'flag' f as a candidate for scanning.
      ii. If the number of 'flagged' flip-flops is $= 0$:
         A. No scan flip-flop is chosen from the SCC S – try the remaining untried SCCs. □

B.1 *Computational Complexity*

Computational Complexity of Procedure 2. Assume there are N logic gates, F flip-flops, and S SCCs in the CUT. Each SCC has F_{SCC} flip-flops and the CUT is logic simulated with V vectors. The complexity of Step 1 is $O(VN)$, and of Step 2 is $O(F)$. For each SCC, Step 3 calculates the average entropy in Step 3a, tries each flip-flop as a scan candidate in Step 3aiA, and logic simulates (to compute updated probability values) in Step 3aiB. The complexity for average calculation is $O(F_{SCC})$ and the complexity of trying each flip-flop and logic simulating after each try is $O(F_{SCC}(VN))$. So, the total complexity of Step 3 is $O(S(F_{SCC}(VN+1)))$. The total complexity of Procedure 2 is approximately $O(VN(SF_{SCC}+1))$.

C. Combined Analysis

Note that entropy analysis is applied to SCCs with cardinality > 1 and spectral analysis is done on all flip-flops. SPARTAN first performs the spectral analysis and then the entropy analysis. A flip-flop is selected for scan if it is picked by both the spectral and the entropy analysis, and if it is more than 4 levels away from a PI and more than 4 levels away from a PO. Figure 6 shows SPARTAN's flow chart and Algorithm 1 is the algorithm.

Algorithm 1 – The SPARTAN Algorithm.

Input: Circuit netlist.
Output: Modified circuit netlist with scan flip-flops.

SPARTAN (CIRCUIT)
{
1. Construct the s-graph from netlist;
2. Condense s-graph to obtain SCCs;
3. Perform logic simulation with 50K random vectors;
4. Perform spectral analysis via Procedures 1 and 1a in Section III A to obtain SCs for all flip-flops;
5. Select spectral analysis based scan flip-flops;
6. Perform entropy analysis via Procedure 2 in Section III B to obtain entropy for all flip-flops;
7. Select entropy analysis based scan flip-flops;
8. Flip-flops selected by both methods are scanned;
9. If more scanning is required, go to Step 3. Else, exit.

}

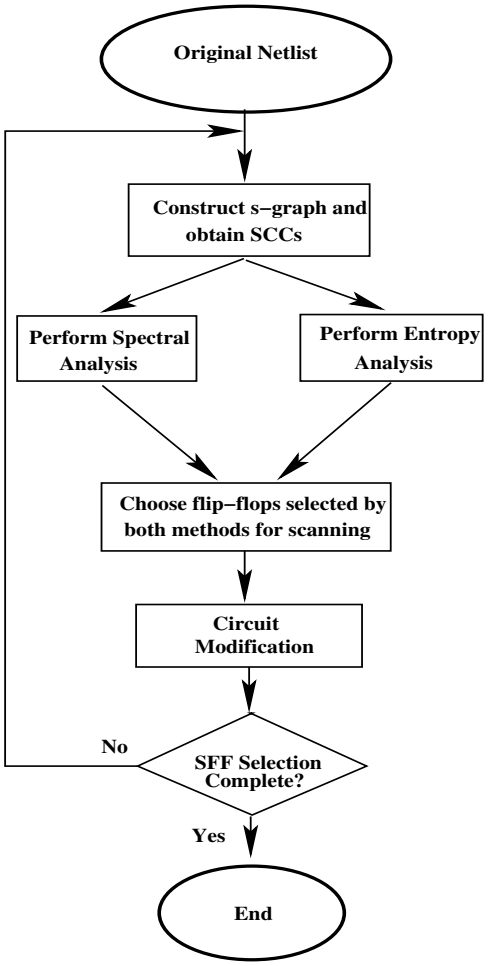

Figure 6: SPARTAN's SFF selection procedure.

IV. **Results**

We first evaluated whether spectral analysis was sufficient for partial-scan insertion. Table 3 shows results for partial scanning using only spectral analysis. We simulated 50,000 random vectors for each circuit, because prior work used that number. This could be improved by making the number proportional to the number of PIs. The other parameter, ST (scanning threshold), was set to 25% of the flip-flops, which gave the best results, on average. The best result (FC, TL, TV, and TAT) for each benchmark is shown in bold in each table. We compare our results to the *mpscan* results, as they had the highest fault coverage of any partial-scan method previously reported. HITEC is used to generate test patterns for all three tables discussed below.

We added entropy analysis to spectral analysis because the entropy analysis uses structural (SCC) and functional (logic probabilities) information to select flip-flops [22]. The two analyses together will use both structural and functional analysis to improve the quality of the scan set. Table 4 shows results with SPARTAN using a combination of spectral and entropy analysis to select scan flip-flops for *ISCAS-89* benchmarks. Column 4 gives the number of SFFs, column 5 is the *fault coverage* (FC), column 6 is the *test efficiency* (TE) or *fault efficiency*, column 7 is the TL, column 8 is the *test volume* (TV), and column 9 is the *test application time* (TAT) for SPARTAN.

$$TV = \#\,of\ test\ vectors \times (\#\,PIs + \#\,of\ SFFs) \quad (25)$$

$$TAT = \#\,of\ SFF \times TL \quad (26)$$

Similarly, columns 11 to 16 of Table 4 report the results for Xiang and Patel's partial-scan algorithm, *mpscan* [12]. Method *mpscan* has better results for TL, TV, and TAT than SPARTAN for smaller circuits, but SPARTAN gets a better result than *mpscan* as circuit size increases. SPARTAN gets an average of 97.6% FC and 99.7% TE, while *mpscan*'s average FC is 96.7%. SPARTAN's average TV is 349 Kbits, whereas *mpscan*'s average TV is 361 Kbits. The TAT is the number of clock cycles required to shift data into the scan chain (see Equation 26). SPARTAN's average TAT is 327971 cycles and *mpscan* attained 306376 cycles. SPARTAN's average TAT is 6.58% longer than *mpscan*, but SPARTAN has a higher FC and TE and lower TV and TAT than *mpscan* for the larger benchmarks. SPARTAN has a higher FC for s9234 and 34.6% lower TV and 23.3% lower TAT. For s15850, SPARTAN has a higher FC and TE and 21.4% lower TV and 1.1% lower TAT. For s38417, SPARTAN also has a higher FC and TE and 11.4% lower TV and 6.4% lower TAT.

Table 5 shows SPARTAN results after compaction was applied to the HITEC vectors generated for the circuit with DFT using the spectral ATPG compactor [4]. This not only greatly compacted the vectors, but also slightly improved FC. In Table 5, on average, SPARTAN's TV is 23.5% lower than *mpscan* and TAT is 15.1% is shorter than *mpscan*. Although data and control lines have different testability requirements, we have no information about which lines are data vs. control in the ISCAS '89 circuits, so we treat both kinds of lines identically. Also, this partial-scan method can support multiple scan chains.

V. **Conclusion**

We proposed a new partial-scan algorithm that is the first algorithm that uses both spectral and entropy analysis to select scan flip-flops. Difficult to control flip-flops have an adverse effect on the testability of the circuit. The first idea here is to scan flip-flops that greatly increase the controllability of unscanned flip-flops. Low toggling flip-flops are difficult to control from PIs. Spectral analysis helps the algorithm to find flip-flops that have low activity. From the results shown in Table 4, we can safely deduce that spectral analysis gives us information regarding the toggling activity of each flip-flop.

The second idea involves scanning flip-flops such that information flow through unscanned flip-flops from PIs is improved. In other words increasing information flow (entropy) through the circuit improves circuit testability. If the amount of information passing through a flip-flop is high, the controllability and observability of the flip-flop is high. The entropy analysis combined with spectral analysis gave us a higher FC and TE than *mpscan*. The TV and TAT are comparable to *mpscan* and SPARTAN does better than *mpscan* on the large benchmarks.

VI. **Future Work**

We need a formal proof that the increase in entropy of a flip-flop is analogous to increasing its controllability. We are

Table 3: *ISCAS-89* benchmark results for spectral analysis only compared to *mpscan* [12].

Ckt.	PIs	FFs	SPARTAN							*mpscan* [12]					
			SFF	FC (%)	TE (%)	TL	TV (bits)	TAT (cycles)	CPU Time (s)	SFF	FC (%)	TE (%)	TL	TV (bits)	TAT (cycles)
s298	3	14	4	**99.00**	100.00	161	1127	644	4	2	98.7	100.0	160	**800**	**320**
s344	9	15	7	**99.70**	100.00	75	**1200**	525	7	3	99.4	100.0	141	1692	**423**
s349	9	15	7	**99.10**	100.00	52	**832**	**364**	4	3	98.9	100.0	161	1932	483
s382	3	21	13	**99.75**	100.00	108	1728	1404	17	6	99.0	100.0	168	**1512**	**1008**
s386	7	6	4	**100.00**	100.00	217	2387	868	9	4	**100.0**	100.0	205	**2255**	**820**
s400	4	21	10	**97.90**	100.00	113	**1582**	**1130**	9	4	95.8	100.0	394	3152	1576
s444	3	21	13	**96.80**	100.00	92	**1472**	1196	11	6	96.2	100.0	193	1737	**1158**
s526	3	21	15	**99.80**	100.00	182	**3276**	**2730**	23	10	99.3	100.0	273	3549	**2730**
s641	35	19	12	**100.00**	100.00	147	6909	1764	19	1	99.4	100.0	286	**5720**	**286**
s713	35	21	9	**93.10**	100.00	186	**8184**	1674	13	1	92.9	100.0	310	11160	**310**
s820	18	5	2	**100.00**	100.00	679	13580	1358	6	2	**100.0**	100.0	602	**12040**	**1204**
s953	16	29	6	**100.00**	100.00	252	**5544**	1512	19	3	**100.0**	100.0	339	6441	**1017**
s1423	17	74	58	**98.10**	99.70	246	**18450**	**14268**	276	41	**98.1**	99.7	397	23026	16277
s1488	8	6	4	**100.00**	100.00	406	**4872**	1624	23	2	**100.0**	100.0	639	6390	**1278**
s1494	8	6	4	**99.20**	100.00	391	**4692**	1564	22	2	99.1	100.0	622	6220	**1244**
s5378	35	179	79	97.00	99.90	1100	125400	86900	123	50	**97.2**	100.0	1023	**86955**	**51150**
s9234	19	228	177	**94.30**	98.70	1332	**261072**	**235764**	33480	97	93.0	98.6	3114	361224	302058
s13207	31	669	336	**85.92**	98.40	7452	2734884	2503872	6390	58	85.6	95.8	9805	**872645**	**568690**
s15850	77	534	365	89.41	99.00	3402	1503684	1241730	46140	180	**94.8**	99.9	4527	**1163439**	**814860**
s35932	35	1728	433	**89.80**	99.99	456	213408	197448	1617	150	**89.8**	100.0	252	**46620**	**37800**
s38417	28	1636	946	**95.20**	96.50	5393	5252782	5101778	5040	400	94.5	96.0	11573	**4953244**	**4629200**
s38584	12	1452	651	89.50	94.80	7568	5017584	4926768	31615	-	-	-	-	-	-
Avg. w/o s38584				**96.90**	**99.60**	**1069**	484145	447624	4441		96.7	99.5	1675	**360560**	**306376**

Table 4: *ISCAS-89* benchmark results for SPARTAN (spectral with entropy) compared to *mpscan* [12].

Ckt.	PIs	FFs	SPARTAN							*mpscan* [12]					
			SFF	FC (%)	TE (%)	TL	TV (bits)	TAT (cycles)	CPU Time (s)	SFF	FC (%)	TE (%)	TL	TV (bits)	TAT (cycles)
s298	3	14	11	**99.70**	**100.00**	93	1302	1023	27	2	98.7	**100.0**	160	**800**	**320**
s344	9	15	7	**99.70**	**100.00**	75	**1200**	525	55	3	99.4	**100.0**	141	1692	**423**
s349	9	15	8	**99.10**	**100.00**	35	**595**	**280**	57	3	98.9	**100.0**	161	1932	483
s382	3	21	17	**99.00**	**100.00**	142	2840	2414	59	6	**99.0**	**100.0**	168	**1512**	**1008**
s386	7	6	4	**100.00**	**100.00**	217	2387	868	11	4	**100.0**	**100.0**	205	**2255**	**820**
s400	4	21	16	94.60	**100.00**	109	**2180**	1744	19	4	**95.8**	**100.0**	394	3152	**1576**
s444	3	21	18	**96.80**	**100.00**	171	3591	3078	43	6	96.2	**100.0**	193	**1737**	**1158**
s526	3	21	19	**99.80**	**100.00**	176	3872	3344	143	10	99.3	**100.0**	273	**3549**	**2730**
s641	35	19	9	**99.40**	**100.00**	235	10340	2115	79	1	**99.4**	**100.0**	286	**5720**	**286**
s713	35	21	9	**92.90**	**100.00**	231	**10164**	2079	65	1	**92.9**	**100.0**	310	11160	**310**
s820	18	5	2	**100.00**	**100.00**	679	13580	1358	10	2	**100.0**	**100.0**	602	**12040**	**1204**
s953	16	29	23	**100.00**	**100.00**	234	9126	5382	127	3	**100.0**	**100.0**	339	**6441**	**1017**
s1423	17	74	63	**98.50**	99.30	227	**18160**	**14301**	673	41	98.1	**99.7**	397	23026	16277
s1488	8	6	5	**100.00**	**100.00**	347	**4511**	1735	127	2	**100.0**	**100.0**	639	6390	**1278**
s1494	8	6	4	**99.20**	**100.00**	445	**5340**	1780	124	2	99.1	**100.0**	622	6220	**1244**
s5378	35	179	114	**98.96**	99.90	1165	173585	132810	2443	50	97.2	**100.0**	1023	**86955**	**51150**
s9234	19	228	199	**93.40**	**98.60**	1231	**268358**	**244969**	4658	97	93.0	**98.6**	3114	361224	302058
s13207	31	669	496	**98.00**	**100.00**	2360	1243720	1170560	8915	58	85.6	95.8	9805	**872645**	**568690**
s15850	77	534	407	**95.10**	**99.96**	1980	**958320**	**805860**	34075	180	94.8	99.9	4527	1163439	814860
s35932	35	1728	477	**89.80**	99.99	298	152576	142146	101435	150	**89.8**	100.0	252	**46620**	**37800**
s38417	28	1636	1249	**96.00**	**96.70**	3482	**4446514**	**4349018**	43670	400	94.5	96.0	11573	4953244	4629200
Avg.				**97.60**	**99.70**	663	**349155**	327971	9372		96.7	99.5	1675	360560	**306376**

currently working on an algorithm that will use spectral and entropy analysis for test point insertion.

Acknowledgments. This material is based on work supported by the National Science Foundation under Grant Nos. 0098304 and 0429743. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We thank R. Sethuram, T. J. Chakraborty, H. V. Venkatanaraynan, and R. Ausoori for valuable discussions.

REFERENCES

[1] T. Niermann and J. Patel, "HITEC: A Test Generation Package for Sequential Circuits," in *Proc. of the European Conf. on Design Automation (EDAC)*, pp. 214–218, Feb. 1991.

[2] J. Rearick, "The Case for Partial Scan," in *Proc. of the Int'l Test Conf.*, pp. 1032–1032, Oct. 1997.

[3] X. Chen and M. L. Bushnell, "Dynamic State and Objective Learning for Sequential Circuit Automatic Test Generation Using Recomposition Equivalence," in *Proc. of the IEEE Int'l. Symp. on Fault-Tolerant Computing (FTCS)*, pp. 446–445, June 1994.

[4] S. Devanathan and M. L. Bushnell, "Sequential Spectral ATPG Using the Wavelet Transform and Compaction," in *Proc. of the Int'l. Conf. on VLSI Design*, pp. 407–412, Jan. 2006.

[5] S. Chakradhar, V. D. Agrawal, and S. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Trans. on Computer Aided-Design*

Table 5: *ISCAS-89* benchmark results for SPARTAN (spectral and entropy) after compaction of HITEC vectors compared to *mpscan*.

Ckt.	PIs	FFs	SPARTAN							*mpscan* [12]					
			SFF	FC (%)	TE (%)	TL	TV (bits)	TAT (cycles)	CPU Time (s)	SFF	FC (%)	TE (%)	TL	TV (bits)	TAT (cycles)
s298	3	14	11	**99.70**	-	69	966	759	1	2	98.7	100.0	160	**800**	**320**
s344	9	15	7	**99.70**	-	70	**1120**	490	1	3	99.4	100.0	141	1692	**423**
s349	9	15	8	**99.10**	-	35	**595**	**280**	1	3	98.9	100.0	161	1932	483
s382	3	21	17	**99.00**	-	139	2780	2363	1	6	**99.0**	100.0	168	**1512**	**1008**
s386	7	6	4	**100.00**	-	156	**1716**	**624**	1	4	**100.0**	100.0	205	2255	820
s400	4	21	16	94.60	-	109	**1880**	**1504**	9	4	95.8	100.0	394	3152	1576
s444	3	21	18	**96.80**	-	129	2709	2322	1	6	96.2	100.0	193	**1737**	**1158**
s526	3	21	19	**99.80**	-	120	**2640**	**2280**	1	10	99.3	100.0	273	3549	2730
s641	35	19	9	**99.40**	-	128	**5632**	1152	1	1	**99.4**	100.0	286	5720	**286**
s713	35	21	9	**92.90**	-	144	**6336**	1296	1	1	**92.9**	100.0	310	11160	**310**
s820	18	5	2	**100.00**	-	452	**9040**	**904**	3	2	**100.0**	100.0	602	12040	1204
s953	16	29	23	**100.00**	-	145	**5655**	3335	1	3	**100.0**	100.0	339	6441	**1017**
s1423	17	74	63	**98.50**	-	154	**12320**	**9702**	9	41	98.1	99.7	397	23026	16277
s1488	8	6	5	**100.00**	-	249	**3237**	**1245**	7	2	**100.0**	100.0	639	6390	1278
s1494	8	6	4	**99.20**	-	324	**3888**	1296	11	2	99.1	100.0	622	6220	**1244**
s5378	35	179	114	**98.96**	-	822	122478	93708	68	50	97.2	100.0	1023	**86955**	**51150**
s9234	19	228	199	**93.40**	-	958	**208844**	**190642**	122	97	93.0	98.6	3114	361224	302058
s13207	31	669	496	**98.10**	-	1823	960721	904208	732	58	85.6	95.8	9805	**872645**	**568690**
s15850	77	534	407	**95.10**	-	1383	**669372**	**562881**	722	180	94.8	99.9	4527	1163439	814860
s35932	35	1728	477	**89.80**	-	165	84480	78705	143	150	**89.8**	100.0	252	**46620**	**37800**
s38417	28	1636	1249	**96.40**	-	2885	**3684145**	**3603365**	3004	400	94.5	96.0	11573	4953244	4629200
Avg.				**97.60**	-	498	**275741**	**260146**	231		96.7	99.5	1675	360560	306376

of Integrated Circuits and Systems, vol. 12, no. 7, pp. 1015–1028, July 1993.

[6] P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, and H. Wunderlich, "A Modified Clock Scheme for a Low Power BIST Test Pattern Generator," in *Proc. of the VLSI Test Symp.*, pp. 306–311, April 2001.

[7] S. Chakradhar, A. Balakrishnan, and V. D. Agrawal, "An Exact Algorithm for Selecting Partial Scan Flip-Flops," in *Proc. of the* 31^{st} *ACM/IEEE Design Automation Conf.*, pp. 81–86, June 1994.

[8] K.-T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. on Computers*, vol. 39, no. 4, pp. 544–548, Apr. 1990.

[9] R. Gupta, R. Gupta, and M. A. Breuer, "The BALLAST Methodology for Structured Partial Scan Design," *IEEE Trans. on Computers*, vol. 39, no. 4, pp. 538–544, Apr. 1990.

[10] S. Park, "A Partial Scan Design Unifying Structural Analysis and Testabilities," *Int'l. J. on Electronics*, vol. 88, no. 12, pp. 1237–1245, Dec. 2001.

[11] S. Tai and D. Bhattacharya, "A Three Stage Partial Scan Design Method to Ease ATPG," *J. of Electronic Testing: Theory and Applications (JETTA)*, vol. 7, no. 11, pp. 95–104, Nov. 1995.

[12] D. Xiang and J. Patel, "Partial Scan Design Based on Circuit State Information and Functional Analysis," *IEEE Trans. on Computers*, vol. 53, no. 3, pp. 276–287, Mar. 2004.

[13] S. Sharma and M. Hsiao, "Combination of Structural and State Analysis for Partial Scan," in *Proc. of the Int'l. Conf. on VLSI Design*, pp. 134–139, Jan. 2001.

[14] X. Lin, I. Pomeranz, and S. M. Reddy, "Full Scan Fault Coverage with Partial Scan," in *Proc. of the IEEE Design Automation and Test in Europe Conf.*, pp. 468–472, 1999.

[15] J. A. Dussault, "A Testability Measure," in *Proc. of IEEE 1978 Semiconductor Test Conf.*, pp. 113–116, Oct. 1978.

[16] T. Cover and J. Thomas, *Elements of Information Theory*. John Wiley & Sons Inc., Hoboken, New Jersey, 2006.

[17] V. D. Agrawal, "An Information Theoretic Approach to Digital Fault Testing," *IEEE Transactions on Computers*, vol. C-30, no. 8, pp. 582–587, Aug. 1981.

[18] K. Thearling and J. Abraham, "An Easily Computed Functional Level Testability Measure," in *Proc. of the Int'l. Test Conf.*, pp. 381–390, Oct. 1989.

[19] M. J. Williams and J. B. Angell, "Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic," *IEEE Trans. on Computers*, vol. C-22, pp. 46-60, pp. 46–60, Jan. 1973.

[20] V. D. Agrawal, K.-T. Cheng, D. Johnson, and T. Lin, "Designing Circuits with Partial Scan," *IEEE Design & Test of Computers*, vol. 5, pp. 8–15, Mar. 1988.

[21] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," in *Proc. of the* 25^{th} *Int'l. Symp. on Fault-Tolerant Computing*, pp. 337–343, June 1995.

[22] S. Bhawmik, C. Cheng, K.-T. Cheng, and V. D. Agrawal, "PASCANT: A Partial Scan and Test Generation System," in *Proc. of the IEEE Custom Integrated Circuits Conf.*, pp. 17.3/1–17.3/4, May 1991.

[23] D. Lee and S. Reddy, "On Determining Scan Flip-Flops in Partial Scan Designs," in *Proc. of the Int'l. Conf. on Computer-Aided Design*, pp. 322–325, Nov. 1990.

[24] V. Chickermane and J. Patel, "A Fault Oriented Partial Scan Design Approach," in *Proc. of the Int'l. Conf. on Computer-Aided Design*, pp. 400–403, Nov. 1991.

[25] E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," in *Proc. of the IEEE Int'l. Test Conf.*, pp. 153–162, Oct. 1980.

[26] C.-C. Lin, M. Sadowska, K.-T. Cheng, and M. Lee, "Test Point Insertion: Scan Paths Through Combinational Logic," in *Proc. of the* 33^{rd} *ACM/IEEE Design Automation Conf.*, pp. 268–273, Jun. 1996.

[27] K.-T. Cheng and C.-J. Lin, "Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST," in *Proc. of the Int'l Test Conf.*, pp. 506–514, Oct. 1995.

[28] M. Abadir and M. Breuer, "A Knowledge-Based System for Designing Testable VLSI Chips," *IEEE Design and Test of Computers*, vol. 2, pp. 56–68, Aug. 1985.

[29] T. Darmala and M. Karpovsky, "Fault Detection in Combinational Networks by Reed-Muller Transforms," *IEEE Trans. on Computers*, vol. 38, no. 6, pp. 788–797, June 1989.

[30] M. Karpovsky, R. Stankovic, and J. Astola, "Spectral Techniques for Design and Testing of Computer Hardware," in *Proc. of the Int'l Workshop for Spectral Techniques in Logic Design*, pp. 1–34, June 2000.

[31] S. L. Hurst, D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*. Academic Press Inc., London, 1985.

[32] J. Zhang, M. L. Bushnell, and V. D. Agrawal, "On Random Pattern Generation with the Selfish Gene Algorithm for Testing Digital Sequential Circuits," in *Proc. of the Int'l. Test Conf.*, pp. 617–626, October 26-28 2004.

[33] K. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. on Computers*, vol. 24, no. 1, pp. 668–670, Jun. 1975.

[34] J. Saund, M. Hsiao, and J. Patel, "Partial Scan Beyond Cycle Cutting," in *Proc. of the IEEE Int'l. Symp. on Fault-Tolerant Computing*, pp. 320–328, 1997.

[35] D. Xiang and J. Patel, "A Global Partial Scan Design Algorithm Using State Information," in *Proc. of the Int'l. Test Conf.*, pp. 548–557, Oct. 1996.