Evaluation of Heuristic Techniques for Test Vector Ordering

H. Hashempour
Northeastern University, Dept. of ECE
Boston, MA, 02115
hhashemp@ece.neu.edu

F. Lombardi
Northeastern University, Dept. of ECE
Boston, MA, 02115
lombardi@ece.neu.edu

ABSTRACT

Vector reordering is an essential task in testing VLSI systems because it affects this process from two perspectives: power consumption and correlation among data. The former feature is crucial and if not properly controlled during testing, may result in permanent failure of the device-under-test (DUT). The latter feature is also important because correlation is captured by coding schemes to efficiently compress test data and ease memory requirements of Automatic-Test-Equipment (ATE), while reducing the volume of data and lowering the test application time. Reordering however is NP-complete. This paper presents an evaluation of different heuristic techniques for vector reordering using ISCAS85 and ISCAS89 benchmark circuits in terms of time and quality. For this application, it is shown that the best heuristic technique is not the famous Christofides or Lin-Kernighan, but the Multi-Fragment technique.

Categories and Subject Descriptors: J.6.1 [Computer-Aided Design]: VLSI Testing
General Terms: Algorithms
Keywords: ATE, SoC, Test Data, Compression, Power Consumption, Test Vector Ordering

1. INTRODUCTION

Recently, there has been a tremendous growth in the development and application of intellectual property (IP) cores [1]. These cores are provided by third party vendors and are often shipped with test data so that the core integrator can apply the data to the design after manufacturing to ensure its correct operation. As the complexity of these Sea Of Cores (System On Chip) systems increases, testing has become a significant bottleneck.

During testing, all cores must be tested to ensure that they work properly; this requires a considerable power consumption which may often be higher than for normal operation [2]. A higher power consumption (i.e. more energy is taken from the supply), means that more energy is dissipated through the substrate of the circuit, thus resulting in an increased heat dissipation which may burn devices. Several techniques have been studied to address this problem. A test generation technique for low power, has been discussed in [3]. Another approach is based on scheduling so that during testing, maximum power consumption is kept under a certain threshold to avoid burning the DUT [4]. Previous works ([5] [6] [7]) have also suggested reordering of test vectors so that the Hamming distance between adjacent vectors is minimal. They have empirically proved that a minimal Hamming distance translates into a lower activity, thus reducing the power consumption of the DUT. Additionally, the volume of test data can also increase rapidly. This increase affects testing: 1) test time increase is proportional to data volume and 2) the memory requirement of the test equipment system is also increased. To solve the increasing volume, many works have suggested to compress test data [8] [9]. This technique relies on pre-processing vectors by reordering them (i.e. minimizing the Hamming distance between adjacent vectors), differentiating each vector with its successor, and finally applying a coding scheme (e.g. Run-Length coding) to compress the processed data. Reordering is an essential task which is needed to address correlation extraction among test data for compression.

For the above two issues, vector reordering is a critical task for manufacturing test of VLSI systems. Vector reordering however translates into a traveling salesman problem (TSP) which is a known NP-complete problem. The TSP instance for vector reordering is represented by a complete graph with a large number of nodes. Therefore it is essential to find a good heuristic solution in terms of time and quality. The objective of this work is to evaluate a number of different heuristic approaches for this problem in terms of execution time and quality and present one that results in best overall performance for vector ordering application. The rest of this paper is organized as follows: In Section 2. basic concepts and definitions are presented. Section 3. examines a number of heuristic approaches and presents their time complexity and quality. Section 4. presents the simulation results for a number of heuristics used in reordering vectors for a number of benchmarks circuits, followed by conclusions in Section 5.

2. BASIC DEFINITIONS

Consider a test set for a combinational (or full-scan sequential) circuit given by $V = \{v_1, v_2, \cdots, v_n\}$ where $|V| = n$. Each vector v_i is formed by a fixed ordered set of bits b_j, i.e. $v_i = (b_1, b_2, \cdots, b_m)$. The *Hamming Distance* between each two vectors $v_i = (b_k)$ and $v_j = (c_k)$ is defined as the number of 1s obtained from the operation $v_i \; XOR \; v_j$ where XOR is the bitwise XOR operation, i.e.

$$H_D(v_i, v_j) = \sum_{k=1}^{m} b_k \oplus c_k \quad (1)$$

The *Total* Hamming distance of a given ordering (sequence) is calculated by finding the Hamming distance between each adjacent pair of vectors starting from the first vector. As-

GLSVLSI'04, April 26–28, 2004, Boston, Massachusetts, USA.

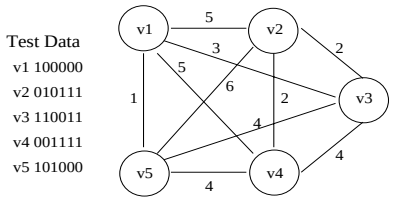

Figure 1: Example of graph construction

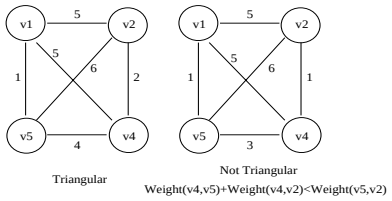

Figure 2: Examples of triangular property

sume an ordering π of input vectors $\{1, 2, \cdots, n\}$. The initial ordering is defined by $\pi(i) = i$. The total Hamming distance for a generic ordering π is given by

$$TH_D = \sum_{i=1}^{n-1} H_D(v_{\pi(i)}, v_{\pi(i+1)}) \quad (2)$$

The reordering problem consists of finding an ordering of vectors that gives a minimal total Hamming distance. To solve this problem, a graph is generated by assigning a vertex to each vector and an edge between each two vertices (vectors). The fully connected graph is weighted; each edge weight is equal to the Hamming distance of the connecting vectors [7] [8]. This graph is undirected because the XOR operation has the exchange property. Figure 1 shows a graph constructed for the sample test data.

If a cycle that traverses only once all nodes of this graph is found, and the sum of its edge weights is minimal over all possible cycles (minimum Hamiltonian cycle), then the cycle also finds the optimal order of the test vectors. This corresponds to the *traveling salesman problem* (TSP) in which a traveling salesperson visits all cities and returns to the original city, with the shortest path. The ordering found from TSP directly affects the so-called correlation [8] among test vectors for compression. Since the total Hamming distance is minimal, and vectors are bitwise XORed, then there are long runs of 0 appearing in the test data. These long runs are used to compress the test data by employing a coding technique, such as Run-Length or Golomb [8] [9].

The ordering found by solving the TSP indirectly affects power consumption during testing. The application of vectors to a DUT in such an order, triggers minimal activity on the primary inputs of the circuit; however in general, it does not guarantee minimal activity over the internal nodes of the DUT. It has been empirically shown that the ordering with minimal total Hamming distance also produces efficient low power manufacturing test [7]. For better results, edge weights can be set according to the total switching activity that the application of a vector pair (v_i, v_j) triggers inside the chip. This results in two arcs per edge because (v_j, v_i) may be different from (v_i, v_j). As added complexity, for every possible combination of vectors, logic simulation of the circuit is required to obtain the edge weights. This is computationally more expensive compared to the calculation of the Hamming distances.

Assume that the DUT has a total of N nodes including primary inputs, internal nodes, and primary outputs. The *switching activity* triggered over the DUT when two vectors (v_i, v_j) are applied in the same order, is defined as

$$S_A(i, j) = \sum_{k=1}^{N} (Node(k, v_i) \oplus Node(k, v_j)) \quad (3)$$

where the $Node(k, v_i)$ function returns a boolean value for node k when vector v_i is applied. Then, the *Total* switching activity for a given ordering π is calculated as:

$$TS_A = \sum_{i=1}^{n-1} S_A(\pi(i), \pi(i+1)) \quad (4)$$

In general, the directed graph generated by mapping vectors to vertices and $S_A()$ to arcs, is still not a precise model for measuring the power consumption of a given ordering π, because different nodes may have different capacitance. However, it is better than using an undirected graph in which the edge weights are only represented by the $H_D()$ function.

2.1 Graph Model and Complexity Bounds

In this section, we study the characteristics of the graph obtained for the reordering problem and discuss several different graph models. This study is useful in analyzing the complexity and quality of few possible heuristic solutions to TSP. Consider the minimum tour length for a TSP instance I and denoted to as $Min(I)$. Additionally, consider the optimal solution of a heuristic H with tour length of $H(I)$ for a TSP instance I. If there is no restriction on I, then the following theorem holds [10]

THEOREM 1. *No polynomial time TSP heuristic H can guarantee $\frac{H(I)}{Min(I)} \leq 2^{p(n)}$ for any fixed polynomial $p(n)$ and all instances I* [1].

Many TSP instances are representative of real applications and have the property of *Triangularity* [2]. This means that the shortest path between two vertices is always the direct edge between the two, i.e. given three arbitrary vertices v_i, v_j, v_k, the following condition holds [11] [12]: $Weight(v_i, v_j) + Weight(v_j, v_k) \geq Weight(v_i, v_k)$. Figure 2 shows examples of triangular and non triangular graphs. This yields the following result [13]

THEOREM 2. *There exists an $\epsilon > 0$ such that no polynomial time TSP heuristic H can guarantee $\frac{H(I)}{Min(I)} \leq 1 + \epsilon$ for all instances I having the triangular property*[1].

For example, Christofides heuristic [14] (which is discussed in more detail in the next section) guarantees $\epsilon < 0.5$. Another important class of TSP instances, is referred to as *Euclidean* instances in which the vertices are located in a plane and their distance is described using the so-called l_2 norm, i.e., for two vertices i, j located at geometric positions (x_i, y_i) and (x_j, y_j) respectively, the distance is given

[1] Assuming $P \neq NP$

[2] Some authors call this, the *metric* property.

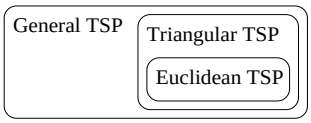

Figure 3: Relation among triangular, euclidean, and general TSP instances

by $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. The Euclidean TSP instances are a subset of the triangular TSP instances, i.e. an Euclidean TSP instance is a triangular TSP instance, but not vice versa [11]. This is shown graphically in Figure 3.

The category of TSP for vector reordering by considering edge weights with the H_D function, is characterized in [5] as follows

THEOREM 3. *The TSP instance constructed for test vector reordering (assuming edge weights are established by the $H_D()$ function), is triangular.*

The case in which edge weights are set according to the transitions over all internal nodes using the $S_A()$ function for the power consumption of the DUT, may be either triangular or non triangular, depending upon the delay model used in the analysis. Under the general delay model, the graph may not be triangular [5].

3. SOLUTIONS TO THE TSP

Many heuristic criteria are available for solving TSP. Two classes of heuristics for TSP can be identified as given below [11]: 1) Tour Construction Heuristics: The heuristic criterion gradually constructs an ordering (possibly for a minimal tour); 2) Local Search Heuristics: The heuristic criterion starts by an initial ordering (tour) and gradually refines it into a new and possibly better ordering. There are a number of algorithms in each category. Table 1 shows sample algorithms which fall in each category [11] [12].

Tour Construction	Local Search
Nearest-Neighbor	2-Opt
Christofides	3-Opt
Multi-Fragment	k-Opt
Nearest-Addition	Genetic
Clark-Wright	Lin-Kernighan

Table 1: Sample heuristic algorithms and their class

For example, the *Nearest-Neighbor* [11] approach is in the class of Tour Construction heuristics in which an ordering is gradually built by adding edges from the TSP graph. In this algorithm, the salesperson starts from any city (graph node), moves to the nearest neighbor city and follows this rule until it traverses all cities and returns to the initial city.

3.1 Execution Time Complexity

In general, execution time complexity of local search heuristics is worse than that of a tour construction. An analysis of worst case time complexity of a number of TSP heuristics is reported in Table 2. n is the number of nodes in the TSP graph. These results are valid assuming the TSP instance is triangular [11] [12].

3.2 Tour Quality

In addition to time complexity, another performance measure is quality. Quality is related to the tour length that the heuristic produces. The heuristic tour length is always greater than or equal to the minimum tour length, hence quality is defined as the length ratio between the heuristic tour and the minimum tour. The quality of the algorithms for TSP is summarized in Table 2 [11] [12].

Algorithm	Quality	Time Complexity
Nearest-Neighbor	$0.5(\lfloor log_2 n \rfloor + 1)$	$O(n^2)$
Multi-Fragment	$0.5(\lceil log_2 n \rceil + 1)$	$O(n^2 log(n))$
Clark-Wright	$\lceil log_2 n \rceil + 1$	$O(n^2 log(n))$
Christofides	1.5	$O(n^3)$
Nearest-Addition	2	$O(n^2)$

Table 2: Worst case quality and execution time complexity of TSP heuristics

For example, the Nearest-Neighbor heuristic algorithm [11] for a TSP of 1000 nodes, guarantees that the heuristic solution length is less than or equal to $0.5(\lfloor log_2 1000 \rfloor + 1)$ or $0.5(\lfloor 9.966 \rfloor + 1) = 5$ times the minimum tour length. No better guarantee is possible. The best tour construction heuristic, which was proposed by Christofides [14], guarantees a worst-case tour length of 1.5 times the minimum tour length. The importance of Christofides algorithm is that quality is independent of the number of nodes. Together, these two measures (time and quality) can lead to a fair comparison among heuristics.

4. EVALUATING HEURISTIC CRITERIA

We have considered two well known heuristics, one from the class of tour construction and one from the class of local search algorithms. The first heuristic is Christofides, which has been used for vector ordering to reduce power consumption [5]. The second heuristic is Lin-Kernighan which has been used in [15] for vector ordering in data compression. Additionally, we have considered a number of heuristics including Nearest-Addition, Nearest-Neighbor, Clark-Wright[3], and Multi-Fragment [11] [12]. We have created a triangular graph model for the vector set of each of the ISCAS85 and (full-scan version) ISCAS89 circuits, and generated fully-specified vectors using HITEC [16].

The GNU TSP solver program (tsp-solve) was compiled in an Alpha workstation. Two heuristics (Nearest-Neighbor and Multi-Fragment) were also implemented in C. The first was not implemented in the GNU TSP solver; the second was implemented for checking the results of the Multi-Fragment heuristic with the TSP solver. The original source code of the TSP solver does not cover instances with $n > 2400$, so we have modified the code to fix this limit and allow benchmark circuit s38417 to be examined.

The graph model of each benchmark circuit was provided to the TSP solver and the C program to find the execution time and quality of the solution. Table 3 shows the results. The results for the Nearest-Neighbor heuristic are from the C program and not the TSP solver. There are two sets of results for the Multi-Fragment heuristic; one is from the C implementation (Multi-Fragment2), and one is from the TSP solver (Multi-Fragment1). The quality in the table is measured in terms of minimal tour length obtained by the heuristic divided by the lower bound of the tour length (if the problem is solved using a relaxed integer linear programming). This is valid because it is not possible to obtain the optimum tour and follow the definition of quality (presented

[3]Also called Savings heuristic.

	Nodes	Nearest-Neighbor		Multi-Fragment1		Multi-Fragment2		Christofides		Clark-Wright		Nearest-Addition		Lin-Kernighan	
Circuit		Time	Quality	Time	Quality	Time	Quality	Time	Quality	Time	Quality	Time	Quality	Time	Quality
c1355	198	1.17s	1.91	0.05s	1.02	0.05s	1.02	0.13s	1.24	0.05s	1.26	0.03s	1.06	57s	1.00
c1908	138	0.45s	1.71	0.02s	1.03	0.02s	1.02	0.03s	1.16	0.02s	1.28	0.00s	1.08	15s	1.00
c2670	102	1.11s	1.19	0.01s	1.01	0.01s	1.01	0.02s	1.06	0.01s	1.08	0.00s	1.02	0.00s	1.00
c3540	350	4.99s	1.87	0.18s	1.02	0.14s	1.03	0.28s	1.19	0.18s	1.26	0.14s	1.06	170s	1.00
c432	100	0.22s	1.54	0.01s	1.02	0.01s	1.02	0.01s	1.15	0.01s	1.19	0.00s	1.07	17s	1.00
c499	184	0.99s	1.73	0.04s	1.02	0.04s	1.02	0.07s	1.21	0.05s	1.23	0.02s	1.06	32s	1.00
c5315	248	5.6s	1.53	0.10s	1.01	0.08s	1.01	0.13s	1.14	0.10s	1.11	0.08s	1.03	53s	1.00
c7552	452	23s	1.70	0.35s	1.01	0.27s	1.01	0.66s	1.17	0.37s	1.12	0.59s	1.04	195s	1.00
c6288	48	0.05s	1.41	0.00s	1.02	0.00s	1.02	0.00s	1.14	0.00s	1.16	0.00s	1.06	1s	1.00
c880	128	0.57s	1.49	0.02s	1.02	0.02s	1.02	0.03s	1.13	0.03s	1.19	0.01s	1.03	8s	1.00
s1196	378	4.91s	2.19	0.19s	1.02	0.17s	1.01	0.33s	1.21	0.19s	1.32	0.13s	1.10	166s	1.00
s13207	1196	525.57s	2.83	3.10s	1.02	3.18s	1.02	6.86s	1.36	6.86s	1.17	25.63s	1.05	1338s	1.00
s15850	1216	487.27s	2.71	3.13s	1.02	3.08s	1.02	7.04s	1.32	6.83s	1.16	24.07s	1.05	1441s	1.00
s35932	104	8.47s	1.34	0.02s	1.01	0.02s	1.01	0.02s	1.12	0.02s	1.07	0.01s	1.10	0.00s	1.00
s38417	2712	6562.21s	3.92	19.85s	1.03	20.33s	1.03	43.91s	1.41	46.10s	1.26	181.49s	1.12	32357s	1.00
s38584	2072	3275.16s	3.59	8.58s	1.01	10.97s	1.02	28.75s	1.36	30.86s	1.23	135.44s	1.09	9917s	1.00
s5378	590	43.38s	1.90	0.62s	1.01	0.50s	1.01	1.11s	1.21	0.69s	1.15	1.58s	1.04	157s	1.00
s9234	1110	199.77s	2.16	2.21s	1.01	2.45s	1.01	6.05s	1.24	4.57s	1.11	13.39s	1.03	1966s	1.00

Table 3: Comparing different heuristics for time and quality

in Section 3.2).

Heuristic	Average Quality	Average time
Nearest-Neighbor	2.04	619s
Multi-Fragment1	1.02	2.14s
Multi-Fragment2	1.02	2.30s
Christofides	1.21	5.30s
Clark-Wright	1.19	5.39s
Nearest-Addition	1.07	21.26s
Lin-Kernighan	1.00	2661s

Table 4: Average quality and time of heuristics

The Multi-Fragment heuristic[4] performs very close to the minimal tour with a very short execution time. For quality, the Multi-Fragment heuristic performs better than all, but one heuristic, i.e. Lin-Kernighan. However, the timing comparison between the Multi-Fragment and Lin-Kernighan heuristics shows that the Multi-Fragment execution time is significantly smaller than that of Lin-Kernighan. Table 4 shows the average execution time and quality of each heuristic over the 18 circuits of Table 3. The Multi-Fragment heuristic starts by sorting all edges in the TSP graph in ascending order of length. A minimal tour is then constructed by selecting *safe* edges in the order. An edge is safe if by adding it to the current constructed tour, it does not create a loop of length less than n (where n is the number of nodes) and does not create a node of degree 3 (the degree of a node is the number of edges incident upon it).

5. CONCLUSION

An experimental evaluation of test vector ordering heuristics has been presented using quality and execution time as figures of merit. It has been shown that the Multi-Fragment heuristic performs better than Christofides and Lin-Kernighan heuristics in terms of time using realistic benchmark vector sets. The Multi-Fragment heuristic also outperforms the Christofides heuristic in terms of quality and achieves performance very close to Lin-Kernighan. We recommend ordering algorithms to use the Multi-Fragment heuristic for near-minimal ordered sets of vectors that result in both reduced power consumption and enhanced data compression ratio.

[4]Also called Greedy heuristic.

6. REFERENCES

[1] R. K. Gupta and Y. Zorian., "Introducing Core-Based System Design," *IEEE Design & Test,* Vol. 14, No. 4, pp. 15-25, Oct-Dec 1997.

[2] F. Corno, M. Rebaudengo, M. Reorda and M. Violante, "Optimal Vector Selection for Low Power BIST," *Proc. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems,* pp. 219-226, 1999.

[3] S. Wang and S. K. Gupta, "ATPG for Heat Dissipation Minimization During Test Application," *Proc. IEEE Int. Test Conf.,* pp. 250-258, 1994.

[4] R. M. Chou, K. K. Saluja and V. D. Agrawal, "Power Constraint Scheduling of Tests," *Proc. Int. Conf. on VLSI Design,* pp. 271-274, 1994.

[5] V. Dabholkar, S. Chakravarty, I. Pomeranz and S. Reddy, "Techniques for Minimizing Power Disspation in Scan and Combinational Circuits During Test Application," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 17, No. 12, pp 1325-1333, Dec. 1998.

[6] P. Flores, J. Costa, H. Neto, J. Monteiro and J. M-Silva, "Assignment and Reordering of Incompletely Specified Pattern Sequences Targeting Minimum Power Dissipation," *Proc. Int. Conf. on VLSI Design,* pp. 37-41, 1999.

[7] P. Girard, C. Landrault, S. Paravossoudovitch and D. Severac, "Reduction of Power Consumption During Test Application by Test Vector Ordering," *IEE Electronics Letters,* Vol. 33, No. 21, pp. 1752-1754, Oct. 1997.

[8] A. Jas and N. A. Touba, "Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs," *Proc. IEEE Int. Test Conf.,* pp. 458-464, 1998.

[9] A. Chandra and K. Chakrabarty, "Test Data Compression for System-on-a-Chip Using Golomb Codes," *Proc. IEEE VLSI Test Symp.,* pp. 113-120, 2000.

[10] S. Sahni and T. Gonzalez, "P-Complete Approximation Problems," *Journal of ACM,* Vol. 23, No. 3, pp. 555-565, July 1976.

[11] E. Aarts and J. K. Lenstra, "Local Search In Combinatorial Optimization," *John Wiley & Sons Pub.,* 1997.

[12] E. L. Lawer, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, "The Traveling Salesman Problem," *John Wiley & Sons Pub.,* 1985.

[13] S. Arora, C, Lund, R. Motwani, M. Sudan and M. Szegedy, "Proof Verification and Hardness of Approximation Problems," *Proc. IEEE Symp. on Foundations of Computer Science,* pp. 14-23, 1992.

[14] N. Christofides, "Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem," *Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University,* 1976.

[15] H. Hashempour and F. Lombardi, "ATE-Amenable Test Data Compression with no Cyclic Scan Register," *Proc. IEEE Int. Conf. on Defect and Fault Tolerance in VLSI Systems,* pp. 151-158, 2003.

[16] T. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *Proc. European Design Automation Conf.,* pp. 214-218, 1991.