Finding Optimum Clock Frequencies for Aperiodic Test

by

Sindhu Gunasekar

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 4, 2014

Keywords: Aperiodic clock, Test time reduction, Automatic Test Equipment

Abstract

With scale down in technology, size and complexity of integrated circuits increase. The scan method is the most popular technique of testing sequential circuits today. In this method, flip-flops functionally form one or more shift registers in the test mode. Faults in the combinational logic can be tested by shifting test patterns in and out of the shift register. Larger circuits these days consist of many flip-flops and the scan design has large shift registers, which require many clock cycles for loading and unloading. An ATE is used to test these scan circuits for faults after fabrication. The testing cost of using an ATE increases directly with the time spent in testing the chip and adds to the final cost of the chip. In general, power constraints do not allow speeding up of the clock during test. For the complex integrated circuits today, long test times are a concern. A recently proposed methodology [37] reduces the test time on ATE using an aperiodic clock with many different frequencies. In practice, the ATE generates only a limited number of frequencies and determining an optimum set of frequencies is a discrete optimization problem. This work discusses an algorithm to obtain the optimum set of frequencies for test time reduction. The algorithm is implemented by simulating ISCAS '89 benchmark circuits and verified on the Advantest T2000GS ATE located at Auburn University, Alabama. It is observed that by using just four optimum frequencies, that the tester allows, test time reductions of up to 52% can be obtained in some benchmark circuits.

Acknowledgments

I am grateful to Prof. Vishwani D. Agrawal for all his support and patience throughout. I am thankful to Prof. Adit D. Singh and Prof. Victor P. Nelson for being on my advisory committee. I had been enrolled in almost all classes of the three of them and Prof. Charles E. Stroud, and I definitely enjoyed learning in the classes of these great teachers. I am indebted to them. These four people never stop inspiring me and I would be glad if I would grow up to be at least half as passionate, knowledgeable, intelligent, wise and patient as they are.

It is an honor to be a part of Auburn University and a privilege to learn from its professors. My work was supported by the National Science Foundation Grant CCF-1116213, for which I am grateful.

I am thankful to Dr. I. M. Arockiasamy (Salem, India), my high school teacher for the motivation and emotional support over all these years. I would like to thank all my friends for enduring me, Praveen especially, for all the help throughout my thesis, and my roommates Kavyashree and Sowmya for the wonderful people they are. I thank my parents for all their sacrifices over my overseas travel and their support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generator
DFT	Design for Testability

Chapter 1

Introduction

The VLSI manufacturing process goes through a series of steps and once fabricated, the devices have to be tested. An ATE is used to apply test patterns to an integrated circuit, and analyze the responses from the integrated circuit, and decide if the chip is either good or bad. The ATE is expensive and testing cost of an ATE increases with the time spent in testing the integrated circuit and it adds a significant fraction to the final cost of the chip. Tests cannot always be applied at a faster rate, not even as fast as the functional clock of the integrated circuit allows, because the power dissipation during testing is significant. The test vectors applied for manufacturing testing are different from the functional vectors used for verification. As a result, there is higher activity in the circuit and the power dissipation is not bounded. For proper functioning of the chip, it is inevitable that power dissipation during test not exceed the rated power.

Techniques have been proposed by Venkataramani and Agrawal [36, 37] to reduce test time on the ATE without exceeding the rated power. The activity of the circuit in each clock interval is different and hence the energy consumed in each cycle is not usually the same, and hence the power consumed is different too. A few cycles consume more power than the others, but we are confined to slow down the clock of the entire testing process to accommodate the power concern in these cycles. These authors propose a methodology to reduce test time by dynamically changing the clock interval at every cycle. This would need clocks of more than one frequency to be generated. Ideally, the test time is the minimum when as many clock periods as the number of test cycles, can be generated. But it is seen later in this work that as few as four clock periods can reduce the test time by 52%. When the minimum allowed clock interval depends on the energy consumed, and is different at every cycle, selecting only a few of the many clock intervals that would decide the frequencies of the clocks to be generated is an optimization problem. This optimization problem is discussed in this thesis.

1.0.1 Organization of Thesis

Chapter 2 of the thesis introduces various concepts that are significant in understanding the proposed work. Chapter 3 gives detailed information about previous work done to reduce test time. Chapter 4 details the algorithm proposed in this work. Chapter 5 discusses the simulation and experimental results obtained when the work was implemented on different ISCAS '89 benchmark circuits. Chapter 6 gives the conclusion of the proposed work and proposes relevant future work.

Chapter 2
Background

This chapter deals with the theory behind the topics analyzed in this thesis.

2.1 Scan Design

The size of integrated circuits has increased from a few thousands to billions of transistors in the last few decades. As the complexity of the circuit increases, the internal nodes become harder to test. Circuits are therefore designed with redundant logic to improve observability and controllability [10]. Such techniques are called Design for Testability (DFT) techniques. One such DFT technique is scan design. In a scan circuit all flip-flops are chained to form one or more shift registers. In the test mode, the necessary logic states in all flip-flops can be set to the desired states by shifting in patterns.

2.2 Testing with an ATE

After a circuit is fabricated it is tested for manufacturing defects on an automatic test equipment (ATE). The patterns required are generated with an automatic test pattern generator (ATPG). The ATPG provides the inputs to be applied and the expected output responses of a fault free circuit. A test program that specifies the patterns, voltage levels and timing information is programmed on the ATE. The fabricated chips are mounted on the tester and the patterns are applied to the primary inputs of the circuit. The responses from the primary outputs are analyzed and compared with the expected responses from the ATPG and if the responses match, the chip is considered a good one.

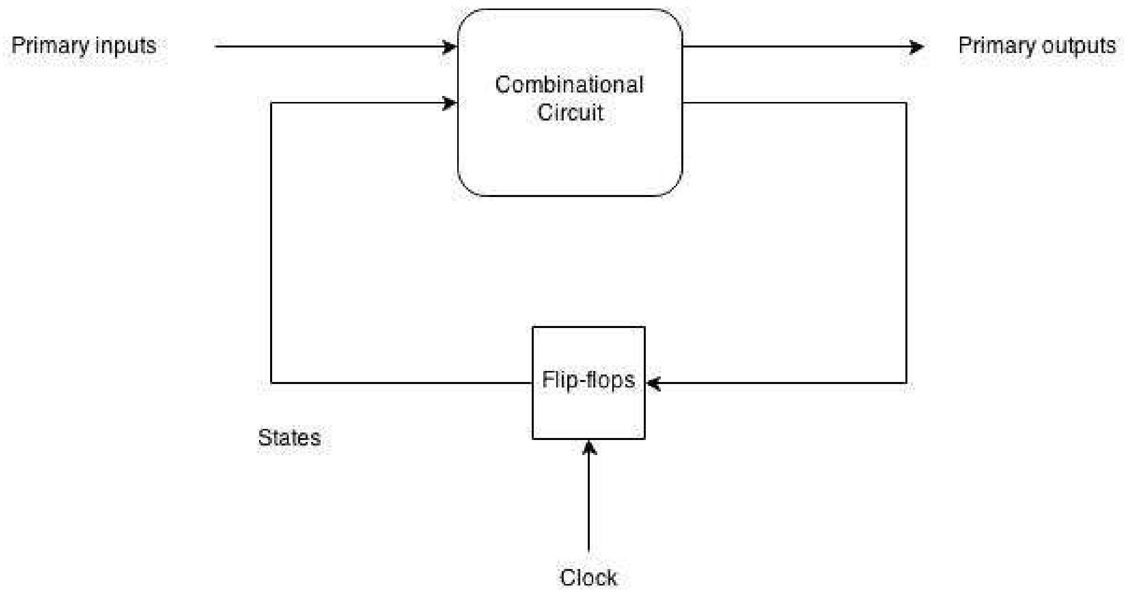


Figure 2.1: Sequential circuit

2.3 ATPG

Automatic Test-Pattern Generators generate patterns by injecting a fault into a circuit model, activating the fault and propagating the fault through the circuit to the output. A fault is activated by establishing at the fault site, a signal value that is opposite of the value produced by the fault model. If the output of the circuit is different from the value expected for the fault-free circuit, the fault is detected.

2.4 Static Timing Analysis

The design of VLSI chips is verified by timing analysis. Static timing analysis [10] analyses combinational paths in the circuit without taking into account if the path is sensitizable or not. The gate delays and interconnect delays are obtained from the technology library.

2.5 Power Dissipation

Power dissipated in a CMOS integrated circuit can be categorized as static and dynamic power. Leakage power is drawn continuously from the power supply and is called static power dissipation. Signal transitions at the nodes that occur due to logic activity or glitches comprise dynamic power dissipation. Because of high toggling activity in the circuit during testing, dynamic power dissipation is higher. Power dissipation in a CMOS circuit is given by

$$P = \frac{1}{2}\alpha f C V^2 \tag{2.1}$$

where α is the activity factor given by the fraction of the gates switching in the circuit, f is the clock frequency, C is the capacitance at the output node and V is the voltage.

2.6 Optimization problem

Optimization implies selecting the best result from a set of available alternatives under the circumstances. Optimization problems are common in many disciplines and various domains. It consists of maximizing or minimizing a function called the objective function by systematically choosing values from within an allowed set and computing the value of the function. Solutions need to be optimal or near-optimal with respect to some goals.

2.7 Global Search

Global search algorithms systematically search the entire space to determine the optimal solution.

2.8 Local Search

If the search space is small, there are algorithms that will systematically search the entire space to determine the solution. If the search space is too big for systematic search, meaningful solutions cannot be found in a reasonable time, because systematic search fails to consider enough of the search space [21]. Local search algorithms work on the current state and generally transcend only to neighbors of the current state in the search space by applying local changes until an optimal solution is found. Such algorithms are not systematic but have two major advantages: They use very little memory and usually converge sooner.

2.9 Greedy Algorithm

A greedy algorithm makes the locally optimal choice that looks best at each stage. The greedy heuristic does not always yield optimal solutions, but nonetheless it may yield locally optimal solutions that approximate a global optimal solution. The time taken to run a greedy algorithm is reasonable compared to global search algorithms.

2.10 Directed Search

Directed search optimization can be used in problems where the objective function is not differentiable, or not continuous. It does not explicitly use any information about the derivative of the objective function [18]. Directed search involves sequential examination of trial solutions by comparing each trial solution with the best solution obtained up to that time. The trial solution is a function of earlier results. Different directed search algorithms employ different strategies to determine the next trial solution [16].

2.11 Computational Complexity

Computational complexity theory classifies computational problems according to their difficulty. P is the class of problems that are solvable in polynomial time. NP is the class of problems which can be solved in non deterministic polynomial time, but can be verified in polynomial time. NP-hard problems are at least as hard as the hardest problems in NP.

2.12 NP-Completeness

A problem is NP-complete if it is in the set of NP problems and also in the set of NP-hard problems. NP-complete problems are very common in various domains.

It is efficient to recognize the class of the problems early and formulate heuristic methods for solving them. NP-complete problems need to be solved approximately instead of exactly.

Chapter 3
Prior Work

This chapter discusses previous work done that relates to the problems solved in this research work.

The scan based method is the most popular for testing sequential circuits. An ATE is used for manufacturing testing of scan circuits. An integrated circuit is packaged to handle the power dissipated during the intended function of the circuit. The ATPG vectors used for testing are generated for high fault coverages. They are functionally irrelevant signals and hence produce very high toggling activity in scan circuits. The high toggling activity leads to higher power dissipation. Hence test power is almost always higher than functional power dissipation. The integrated circuit is not designed to handle test power and testing the circuit at functional speeds might damage the circuit, or in the least lead to malfunction in the circuit during test resulting in yield loss. Test power minimization is hence a widely recognized problem and many techniques have been proposed to solve it [14, 11].

Recent works [36, 37, 40] propose a methodology for minimizing test time of scan based test using an *aperiodic clock*. When using a conventional periodic clock, the dynamic power dissipation for a given set of test vector patterns varies throughout the test, depending on the toggling activity in each clock cycle. It has been suggested [36] that if the clock interval in every cycle can be modified in a way that the dynamic power dissipation remains constant throughout the test, within allowable limits, test time can be significantly reduced. Thus, an *aperiodic clock* can be used, where the period of each cycle can be different from the period of its neighboring cycle. In the aperiodic clock test, every clock interval is either power constrained or structure constrained. The minimum clock interval for each test cycle is given by,

$$T_{test(i)} = \max\left\{T_{structure}, \frac{E_i}{P_{MAX(func)}}\right\} \tag{3.1}$$

where $T_{test(i)}$ is the minimum clock interval for the ith clock cycle, $T_{structure}$ is the structure constrained clock period, E_i is the total energy consumed during the ith cycle, and $P_{MAX(rated)}$ is the maximum rated power of the circuit, given by the specification of the circuit. This equation holds good for calculating the clock interval of scan shift cycles as well as capture cycles. The total test time of the aperiodic clock test is given by,

$$TT_{aperiodic} = \sum_{i=1}^{n} \max\left\{T_{structure}, \frac{E_i}{P_{MAX(func)}}\right\} \tag{3.2}$$

where $TT_{aperiodic}$ is the aperiodic test time for a test with n cycles.

The average power consumed in such an aperiodic test is higher than the conventional method, but test time is reduced significantly. Since energy consumed in each cycle is different, this aperiodic test methodology to reduce test time may, in the worst case, require up to n different frequencies for a circuit that has a test comprising n vectors. The ATE cannot be configured for generating as many as n different frequencies. Nevertheless, it can be seen from the later chapters that considerable reduction in test time can be obtained with fewer than n frequencies. Only a selected subset of clock periods K of the n different clock intervals can be generated and finding the optimum subset of clock periods for minimum test time is a discrete optimization problem. This work proposes an algorithm for selection of the optimum subset of clock periods for maximum reduction in test time.

Chapter 4

Algorithms

Suppose T is the set $\{t_1, t_2, ...t_n\}$ of the n clock intervals for each of the n test cycles of the scan based circuit obtained from simulation. t_i gives the minimum allowed duration of clock interval of each cycle. If the ATE allows only k different clock frequencies to be generated, finding a subset K of T, where K is the set $\{k_1, k_2, ...k_k\}$ consisting of k optimum clock periods for minimum test time, is a combinatorial optimization problem in the discrete domain. This problem, like many other optimization problems, is NP-complete. Using metaheuristics, this NP-complete problem is solved in polynomial-time to find near-optimal solutions. The proposed metaheuristics are verified by simulating the ISCAS '89 sequential benchmark circuits and experimentally on the Advantest T2000GS ATE at Auburn University. The total test time is given by

$$Total\ Test\ Time = \sum_{i=1}^{k} n_i k_i \tag{4.1}$$

where n_i number of cycles at the period k_i.

4.1 Greedy Algorithm

In the greedy algorithm for the optimization problem at hand, at each stage, one optimum clock period is found by one iteration over all the n different clock intervals. Suppose T is the set of all the n different clock intervals arranged in descending order. Initially, one optimum clock period k_1 is selected and this is the largest clock interval t_1 of the set T. The optimum set K now has one element. In the first stage, another clock period k_2 is picked from T and this is determined by performing one iteration over all the remaining $n - 1$

clock intervals to see which one gives maximum reduction in the total test time. Instead of computing the total test time at each iteration, computation time can be significantly reduced by calculating the amount of test time saved when each clock interval of the set T is added to K. At each stage, k_i is determined by that clock interval $x \in t_i, \forall\ i$ which gives a maximum saving in the test time, given by

$$S_x = (k_j - x) \times R \tag{4.2}$$

where,

S_x= saving in test time when x is added to an existing set K
k_j = clock period in the already existing set K which is just greater than x
k_h = clock period in the already existing set K which is just lower than x
R = number of test vectors in T with clock intervals less than x but greater than k_h

At each stage, the set of optimal clock periods K found in the previous stages are unchanged and one optimum value is appended to K. This is repeated until the test time reaches the lower bound, the lower bound given by the sum of all t_i, because t_i defines the minimum length of the clock interval of each cycle and the test time cannot get shorter than the summation of t_i for all cycles. The optimal value of k is then found when the test time reaches the lower bound, which from simulations is seen to be much smaller than n. The computation time of the greedy algorithm increases linearly with the number of allowed ATE frequencies k and the number of test cycles n for testing the circuit and the time complexity is given by $O(nk)$.

4.2 Iterated Local Search Optimization

Iterated local search is when a local search is performed to find a local optimum and then the solution is perturbed to help it escape the local optimum and restart the local search again, in the hope of finding the global optimum or better local optima [9]. The problem of finding k optimum clock-periods has a k-dimensional symmetrical search space and this is quite large even for smaller circuits. The local search method does not systematically search the whole search space but for the problem at hand it helps find a local minimum quickly. And since the search space of the objective function at hand is symmetrical, a local search has a higher probability of converging to the global optimum.

The method initially selects frequencies similar to the greedy algorithm, dynamically one at a time, but then perturbs the current selection and iteratively improves the solution at each step. Initially, one optimum clock period k_1 is selected and this is the largest clock interval t_1 of the set T. The optimum set K now has one element. In the first stage, another clock period k_2 is picked from T and this is determined by performing one iteration over all the remaining $n-1$ clock intervals to see which one gives maximum reduction in the total test time. The third clock k_3 is found similarly and then k_2 is changed to find a better value in the neighborhood that minimizes the objective function further. Then k_3 is again varied to search in the neighborhood for an improvement and so on. This way the algorithm searches a set of neighbors of the current assignment and selects one to be the next current assignment K. The neighbors of the current assignment are those assignments that differ in the assignment of a single variable. The neighborhood is where $k-1$ clock periods in the assignment K are unchanged and the kth clock period takes all the values in between the unchanged values on its either side. The value in the neighborhood that minimizes the objective function is then replaced in the current assignment K. It is analogous to making the $k-1$ dimensions of the search space unchanged and finding the optimum value in the kth dimension that would minimize the objective function, not forgetting the symmetry of the objective function in the k-dimensional space. The assignments are iteratively improved by varying k_1 to k_k, one by one, until the assignments no longer change.

Computing the value of the objective function (test time) at each assignment to find the optimum kth clock period is time-consuming. Instead of computing the total test time at each iteration, computation time can be significantly reduced by calculating the amount of test time saved by the addition of the kth clock when the $k-1$ clocks are unchanged, as given by eq. (4.2). The computation time of local search increases linearly with the number of test cycles n and the time complexity is experimentally found to be $O(nk)$.

4.3 Directed Search Optimization

Pattern Search Optimization is a directed search optimization technique. A pattern search algorithm uses one such strategy where one parameter is varied at any time by steps of the same magnitude, and when such increase or decrease did not improve the objective function, the step size is halved and repeated until the steps become sufficiently small. MATLAB [23] has an inbuilt function implementing the pattern search algorithm.

Suppose T is the set of the n clock intervals for each of the n test cycles of the scan based circuit obtained from simulation. If the ATE allows only k distinct frequencies to be configured, the pattern search algorithm is used to find a subset K of T, K being the set of k optimum clock periods which should be implemented for minimum test time. $T_{replaced}$ is a set that is obtained by replacing $x \in T, \forall\ i$ with a value $y \in K$ where y is the smallest value in the set K, that is greater than x. The objective function with k independent variables is

$$minimize \sum_{i=1}^{n} T_{replaced} \tag{4.3}$$

$T_{replaced}$ is the set of the aperiodic clock periods for n test cycles of the scan based circuit. It has n elements of only k different variables. The objective function has k independent variables and hence the problem at hand is k-dimensional. The algorithm finds set K that would minimize $\sum_{i=1}^{n} T_{replaced}$.

The generalized pattern search algorithm uses a set of vectors p_i called a pattern which determines which point is to be searched in each iteration. It has $2k$ vectors, where k is the number of frequencies allowed by the ATE to be configured. p_i consists of k unit vectors in each of the k dimensions and the negative of these k unit vectors. A set of vectors m_i is formed by multiplying the pattern vectors p_i by a scalar m called the mesh size.

The algorithm initially starts with a random starting point. The vector m_i is added to this point and this forms a mesh. The objective function is evaluated at each point in the mesh and the algorithm proceeds with the point that has the lowest objective function value. This point is made the current point. If the objective function at the current point is lower than the value of the objective function at the previous point, the mesh size is 2. If the objective function at the current point is higher than the value of the objective function at the previous point, the mesh size is changed to 0.5. The mesh sizes can be manipulated to improve the speed or accuracy. This process is repeated until the change in the objective function is less than the tolerance. The time complexity of pattern search optimization is given by $O(nk)$. The computation time increases linearly with the number of allowed ATE frequencies k and the number of test cycles n for testing the circuit.

4.4 Simulated Annealing Optimization

Simulated Annealing is a probabilistic metaheuristic for a global optimization problem. It only needs a single initial starting point and a unary search operation [42] [21]. It is inspired by the process of annealing in metallurgy. Annealing is a heat treatment that can improve the strength of a material. It is the process of closely controlling the temperature when cooling a material to ensure that it reaches an optimal state. Metal crystals have small defects and dislocations of ions which weaken the overall structure. When the metal is heated, the energy of the ions increases and they are allowed to move freely. The dislocations can then be removed and the structure of the crystal is reformed as the material cools down to approach an equilibrium state. The slow cooling schedule allows a new low-energy configuration to be discovered [9]. When annealing the metal, the initial temperature must not be too low and the cooling must be done sufficiently slowly so as to avoid the system getting stuck in a meta-stable, non-crystalline, state representing a local minimum of energy.

A simulated annealing algorithm uses this annealing process to solve an optimization problem. MATLAB [23] has an inbuilt function to perform the simulated annealing algorithm. The objective function eq. (4.3) is to be minimized by the algorithm and the set of optimum clock periods K is to be obtained. The algorithm uses a temperature parameter that starts off high and is slowly lowered in every iteration, storing the best point found so far. An exponential temperature function is used, given by

$$T = T_0 \times 0.95^r \tag{4.4}$$

where r is the annealing parameter. The annealing parameter is the same as the iteration number. The algorithm starts with a random trial point. At each iteration a new trial point is randomly generated and its distance from the current point is based on a probability distribution, proportional to the temperature. The probability distribution is such that the step length of each trial point is equal to the current temperature and the direction is random. If the objective function has a lower value at the new point than the current value, it replaces the current point and the iteration is continued. If the objective function has a higher value at the new point than the current value, it is still accommodated in the search space with a lower probability, which is dependent on the temperature, determined by the simulated annealing acceptance function. This avoids the search getting trapped in a local minimum and helps explore more possible solutions globally. The probability of acceptance used is

$$\frac{1}{1 + \exp(\frac{\Delta}{max(T)})} \tag{4.5}$$

where

$$\Delta = new\ objective - old\ objective$$

$$T = current\ temperature$$

This acceptance function is put to use when the value of the objective at the new trial point is larger than the old objective value. Hence Δ is positive. T is also positive and the probability of the acceptance function lies between 0 and 0.5. An annealing schedule is selected to systematically decrease the temperature as the algorithm proceeds. As the temperature decreases, the algorithm reduces the extent of its search to converge to a minimum. Reannealing helps the algorithm to escape a local minimum, where the temperature is raised after the algorithm accepts a certain number of points and the search is started again at a higher temperature. The algorithm exits when the change in the objective function is lesser than a specified tolerance. The time complexity of the simulated annealing algorithm varies from problem to problem. It is determined experimentally to be $O(nk)$ for our problem. The computation time increases linearly with the number of allowed ATE frequencies k and the number of test cycles n for testing the circuit.

Chapter 5

Simulation and Experiment

The proposed methodology was implemented using ISCAS '89 sequential benchmark circuits. The behavioral models of the benchmark circuits were synthesized using Mentor Graphics Leonardo Spectrum [4] in TSMC 180nm technology. Leonardo Spectrum gives the critical path delay through Static Timing Analysis of the circuit. Using Mentor Graphics DFT Advisor, all the flip-flops in the circuit were daisy chained to form a full scan chain. A set of ATPG test vector patterns for stuck-at faults was generated for the scan circuit using Mentor Graphics Tessent Fastscan [2]. The transistor level description of the netlist was generated using Mentor graphics Design Architect and a SPICE file was generated. Synopsys Nanosim [1] was used to read the SPICE netlist and perform a transistor level simulation at a nominal voltage of 1.8V to measure the energy dissipated per each cycle of the test. Based on these simulations, the minimum clock interval for each cycle was determined, as given by eq. (5.1).

$$T_{test(i)} = \max\left\{T_{structure}, \frac{E_i}{P_{MAX(func)}}\right\} \tag{5.1}$$

The clock interval of each cycle would be constrained by the structure and maximum rated power.

The critical path delay was found from Static Timing Analysis using Leonardo Spectrum. The maximum rated power for the ISCAS '89 benchmark circuits is not available. So, the circuits were simulated in functional mode for 1000 random vectors and the average power dissipated during these simulations was measured and considered the maximum rated power. The set of the clock intervals T for each of the n test cycles of the scan based circuit

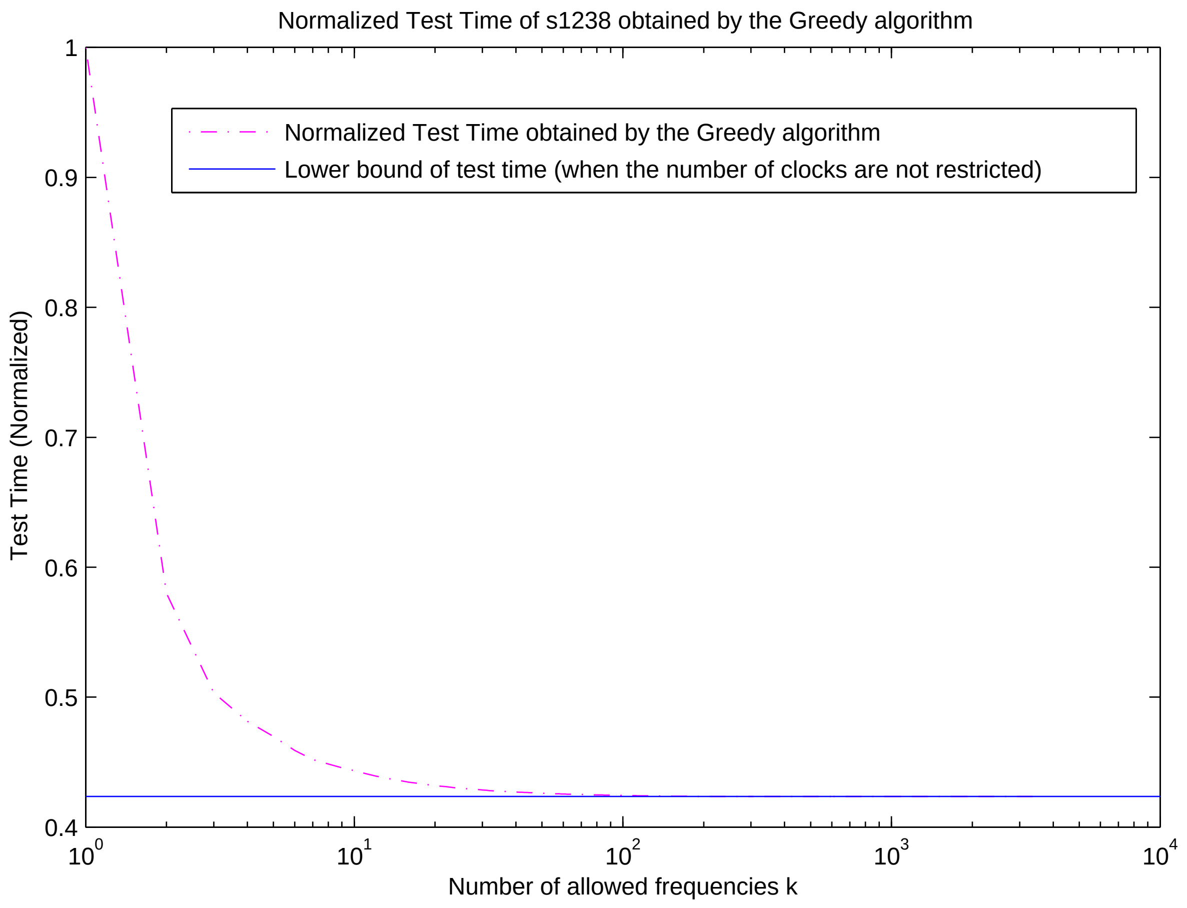
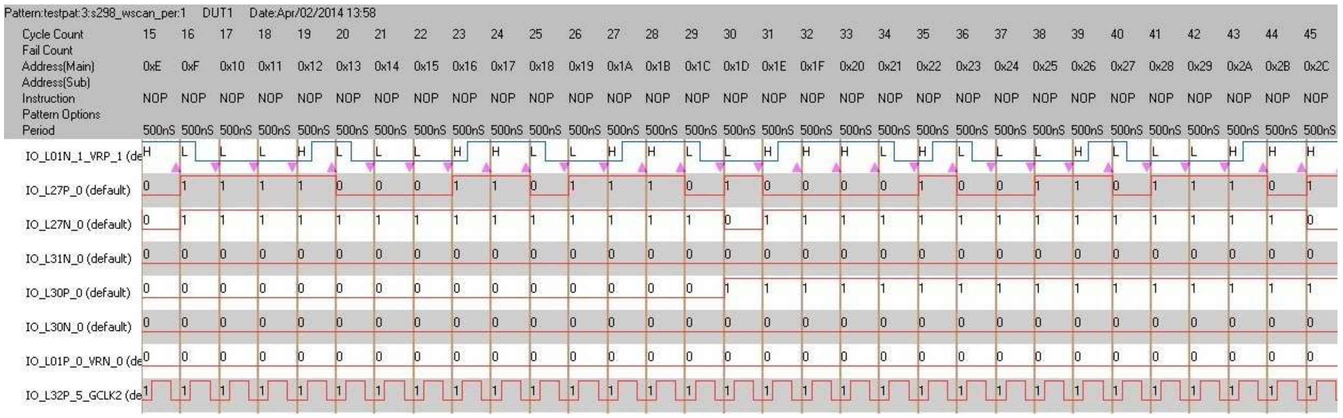


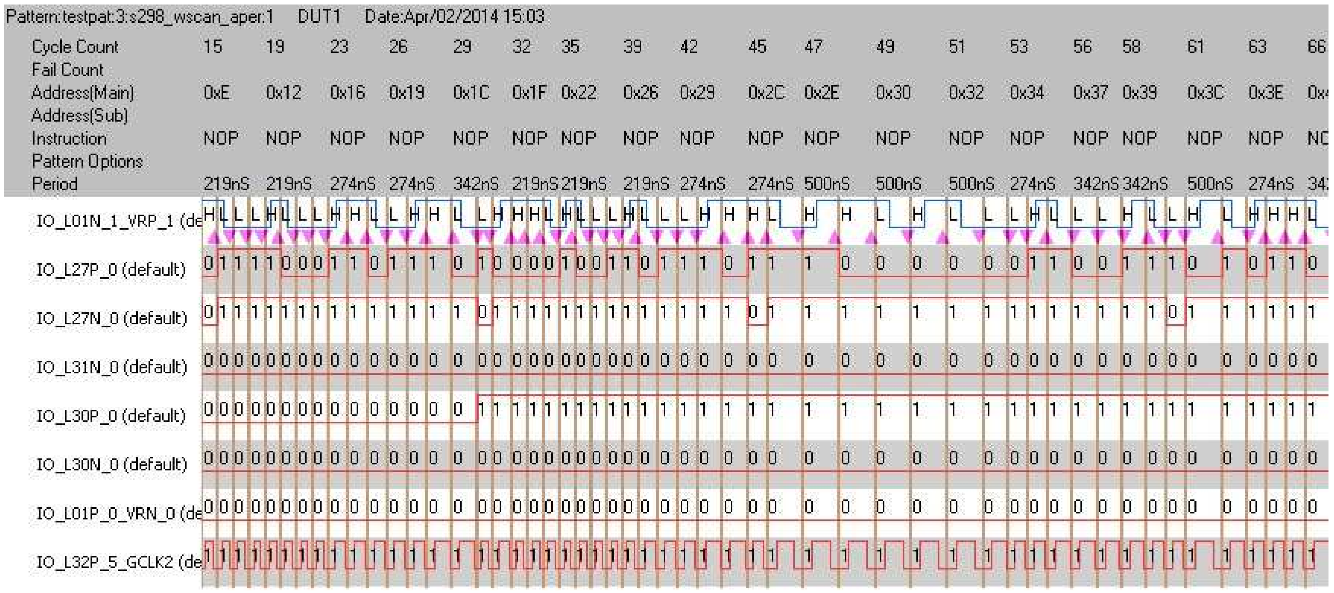
Figure 5.1: Normalized test time of s1238 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum clock frequencies.

were obtained from this simulation. The proposed algorithm was used to find the set of optimum clock periods K from T. Figure 5.1 shows the normalized test time of circuit s1238, obtained by simulation of the greedy algorithm for k optimum frequencies, in a semilog plot. The upper bound in test time is the test time when a conventional periodic clock is used, given by the largest clock interval calculated by eq. (5.1), multiplied by the number of scan test clock cycles. The optimum clock periods were implemented on the Advantest T2000GS ATE which allows four different clock frequencies. The testplan was programmed using the Open architecture Test system Programming Language(OTPL). The s298 benchmark circuit was tested on the ATE with four clock frequencies, and a Xilinx Spartan 3 FPGA XC3S50


Pattern:testpat:3:s298_wscan_per:1 DUT1 Date:Apr/02/2014 13:58

Cycle Count	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
Fail Count																															
Address(Main)	0xE	0xF	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	0x28	0x29	0x2A	0x2B	0x2C
Address(Sub)																															
Instruction	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP
Pattern Options																															
Period	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS	500nS

Figure 5.2: Periodic clock test - ATE result for 540-cycle scan test of s298 benchmark circuit showing test cycles 15 to 45 with a clock of 500ns.


Pattern:testpat:3:s298_wscan_aper:1 DUT1 Date:Apr/02/2014 15:03

Cycle Count	15	19	23	26	29	32	35	39	42	45	47	49	51	53	56	58	61	63	66
Fail Count																			
Address(Main)	0xE	0x12	0x16	0x19	0x1C	0x1F	0x22	0x26	0x29	0x2C	0x2E	0x30	0x32	0x34	0x37	0x39	0x3C	0x3E	0x4
Address(Sub)																			
Instruction	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NOP	NC
Pattern Options																			
Period	219nS	219nS	274nS	274nS	342nS	219nS	219nS	219nS	274nS	274nS	500nS	500nS	500nS	274nS	342nS	342nS	500nS	274nS	34

Figure 5.3: Aperiodic clock test - ATE result for 540-cycle scan test of s298 benchmark circuit showing test cycles 15 to 66 with a clock periods of 219ns, 274ns, 342ns and 500ns.

Table 5.1: Circuit s298 tested on the ATE T2000GS which allows 4 different frequencies

	Periodic clock	Aperiodic clock with arbitrary frequencies	Aperiodic clock with frequencies computed by the greedy algorithm
Test Time (μs)	270	184	158
Percentage of reduction in test time	0	31.85	41.48

soldered on a printed circuit board was used. The s298 benchmark circuit with full scan design was configured on the FPGA. The FPGA was configured on the run by the ATE using the configuration file generated by the Xilinx ISE tool [22]. The testplan was programmed and the test was performed on the ATE accommodating the delay overhead caused due to the analog measurement module. Figures 5.2 and 5.3 show the waveforms of the periodic and aperiodic tests from the Logic Analyzer of the Advantest T2000GS. The two figures have the same time scale. The periodic clock test is run at 500ns. The aperiodic clock test is run at 219ns, 274ns, 342ns and 500ns, the clock periods determined by the greedy algorithm. The periodic test runs for only 30 cycles (cycles 15 to 45), but the aperiodic test runs 51 clock cycles (cycles 15 to 66) in the same time of 15μs. Table 5.1 shows the percentage of reduction in test time obtained when the greedy algorithm was used to compute the aperiodic clocks, in comparison to choosing arbitrary clock periods to test the circuit on the ATE. 41.48% reduction in test time was obtained for the aperiodic test of s298 benchmark circuit with four clock periods computed by the greedy algorithm, as opposed to 31.85% with arbitrary aperiodic clocks tested on the ATE.

5.1 Results

Table 5.2 shows the reduction in test times of the ISCAS '89 benchmark circuits obtained from the greedy algorithm with k optimum frequencies. The optimum number of frequencies k required to achieve the lower bound in test time is computed using the greedy algorithm and tabulated in Table 5.2. The lower bound is calculated by the sum of all t_i, because t_i

Table 5.2: Reduction in test time obtained with aperiodic clocks using the greedy algorithm by simulating ISCAS '89 benchmark circuits, calculated from the test set.

Benchmark circuit	Total scan test clock cycles, n	Number of optimum frequencies required to achieve lower bound in test time, k	Periodic test time (μs)	Aperiodic test time (μs)	Percentage of reduction in Test Time (%)
s298	540	41	2.64	1.39	47.33
s713	773	47	3.41	2.18	36.09
s400	1076	45	4.3	2.99	31.25
s1238	3361	77	22.14	9.37	57.65
s1423	6975	66	15.41	11.11	27.89
s13207	62237	63	52.4	44.24	15.6
s15850	101707	48	241.68	174.06	27.98
s38584	224112	40	759.33	629.8	17.05

defines the minimum length of the aperiodic clock interval of each cycle and the test time cannot get shorter than the summation of t_i for all cycles. The lower bound calculations for different ISCAS '89 benchmark circuits are represented in Figures 5.1, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12 and 5.13. In all of the examples shown in this work, the test time obtained equals the lower bound for k less than 80. Hence, for a test that is n cycles long, n different frequencies are not needed to achieve maximum reduction in test time with aperiodic test.

It can be emphasized that as few as four different frequencies can help achieve up to 52% reduction in test time in a few circuits. The s1238 circuit gives a reduction of 52%, 55% and 57% when four, ten and 77 different frequencies are used, respectively with the greedy algorithm as seen in Figure 5.1. When the number of optimum frequencies k is increased from 10 to 77, or even up to 3361, the test time only reduces by an additional 2%.

The different algorithms detailed in Chapter 4 were implemented on the ISCAS '89 benchmark circuits and compared. The percentage of test time reduction obtained with the algorithms are tabulated in Tables 5.6, 5.3, 5.4, 5.5, 5.7, 5.8, 5.9 and 5.10 for the benchmark circuits for clocks k ranging from 3 to 15. Figure 5.4 shows the normalized test time obtained through different algorithms for the s1238 benchmark circuit. It can be seen that the results of the Iterated Local Search, Directed Search and Simulated Annealing coincide exactly and the greedy algorithm gives a negligibly larger test time for k less than 6. Nevertheless, all four algorithms coincide for k greater than 6. The greedy algorithm is the quickest and, from the comparison, it can be seen that it is as good as any of the other algorithms. This result can also be generalized over the other benchmark circuits simulated in this work.

CPU times for the greedy algorithm were on the order of ms for the smaller circuits and up to 11 seconds for the largest circuit s38584 with 224,112 clock cycles on a computer with Intel Core i3 CPU, 2.27GHz 4GB RAM. The CPU times for computing the discussed algorithms are compared in Figure 5.5 for the s1238 benchmark circuit. All four algorithms slow down linearly with k. The greedy algorithm is the quickest and the simulated annealing is the slowest. Figure 5.6 shows the CPU times of the greedy algorithm and iterated local search again, because they weren't apparent in figure 5.5.

Table 5.3: Percentage of reduction in test time of s298 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)			
		Greedy Algorithm	Iterated Local Search	Direct Search	Simulated Annealing
s298	3	36.77	37.33	37.31	37.32
	4	39.79	39.84	40.09	39.89
	5	41.69	42.04	42.04	41.78
	6	43.01	43.02	43.01	42.92
	7	43.64	43.72	43.70	43.65
	8	44.12	44.21	44.25	44.19
	9	44.55	44.61	44.64	44.53
	10	44.91	44.97	44.96	44.60
	11	45.16	45.17	45.20	44.93
	12	45.38	45.43	45.40	45.03
	13	45.55	45.60	45.56	45.20
	14	45.69	45.74	45.72	45.10
	15	45.80	45.82	45.80	45.67

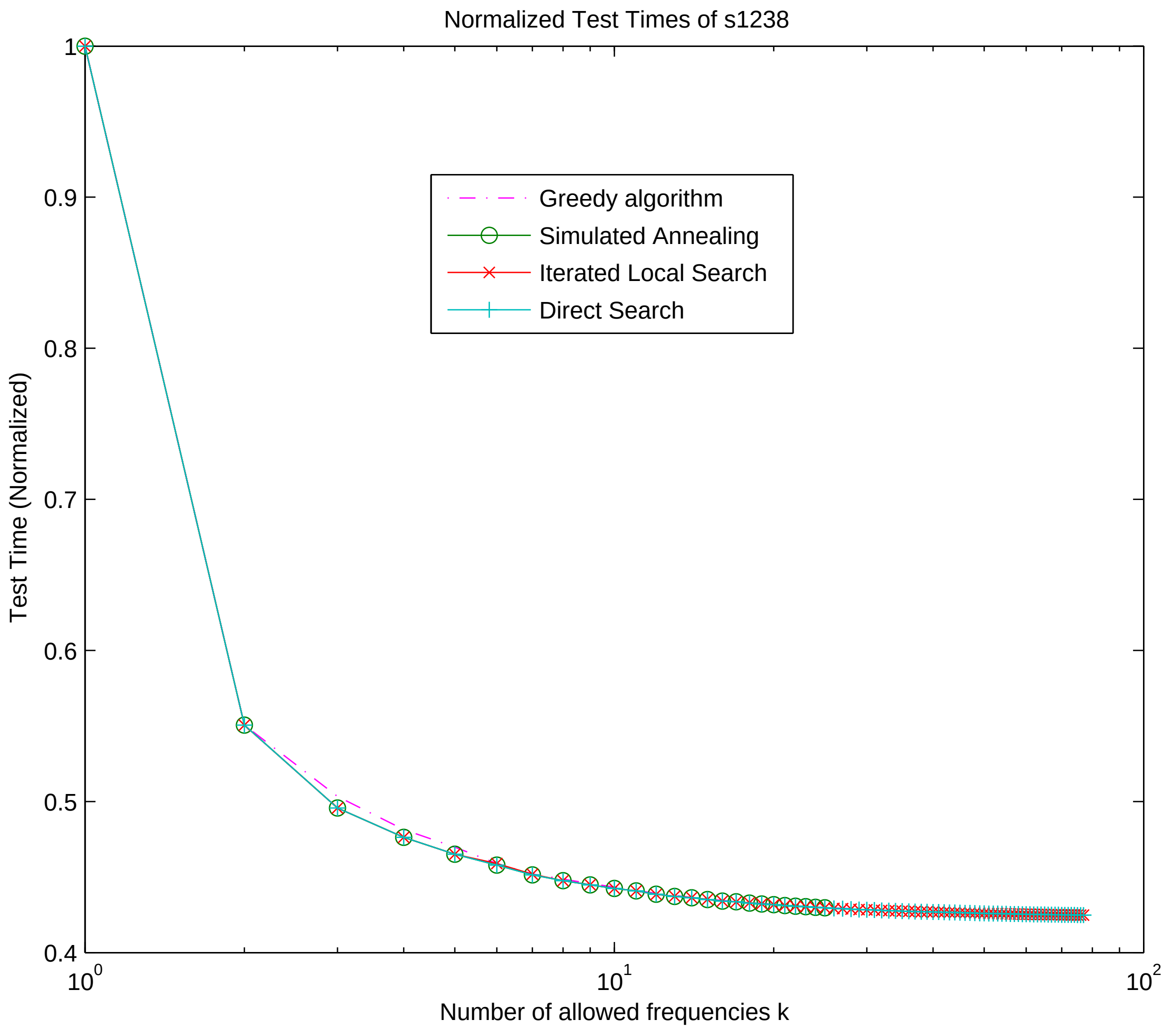


Figure 5.4: Normalized test time of s1238 ISCAS '89 benchmark circuit by different algorithms for k optimum frequencies.

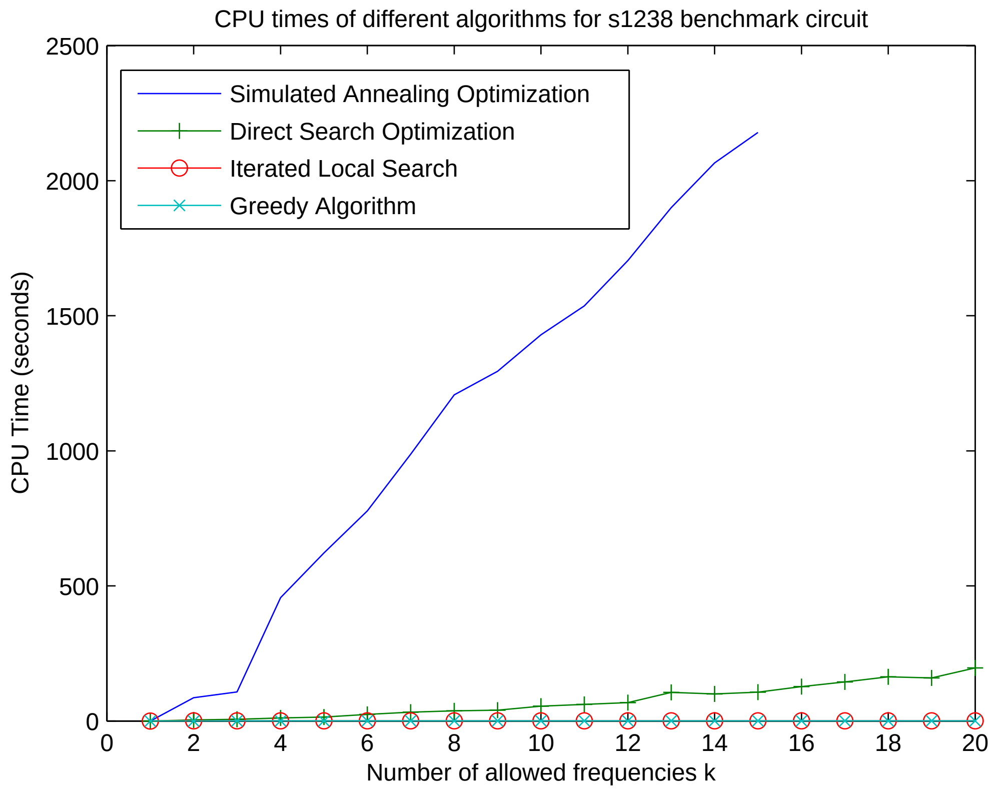


Figure 5.5: CPU times of different algorithms for s1238 benchmark circuit on an Intel Core i3 CPU, 2.27GHz 4GB RAM.

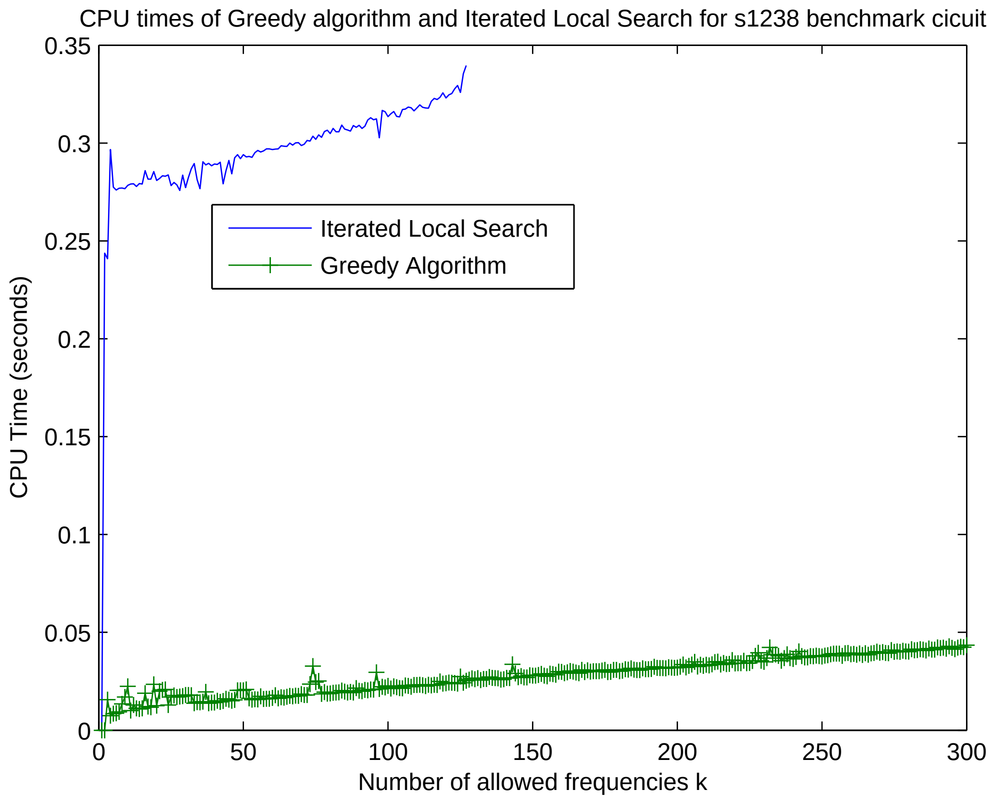


Figure 5.6: CPU times of greedy algorithm and iterated local search for s1238 benchmark circuit on an Intel Core i3 CPU, 2.27GHz 4GB RAM.

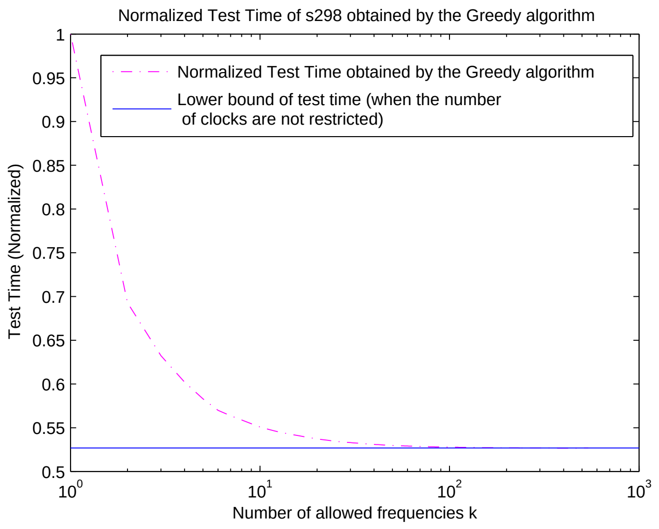


Figure 5.7: Normalized test time of s298 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum frequencies.

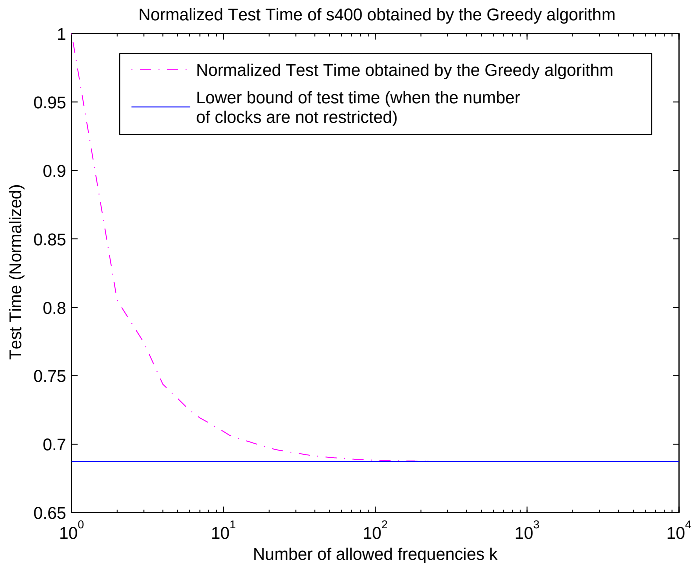


Figure 5.8: Normalized test time of s400 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum frequencies.

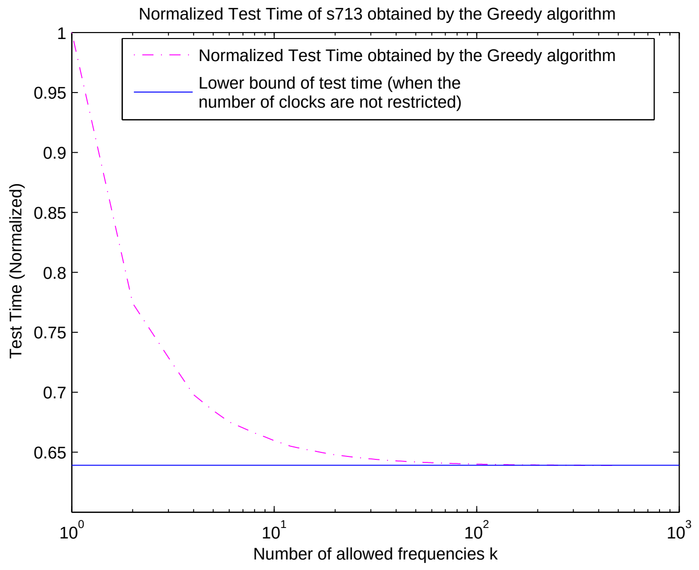


Figure 5.9: Normalized test time of s713 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum frequencies.

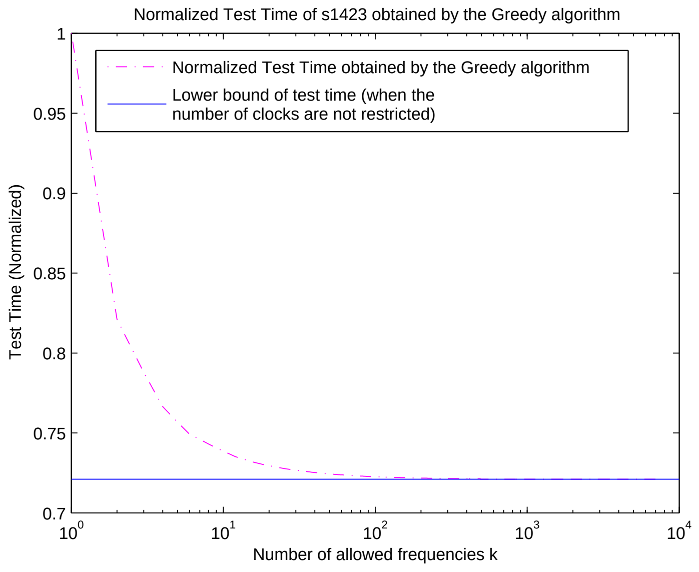


Figure 5.10: Normalized test time of s1423 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum frequencies.

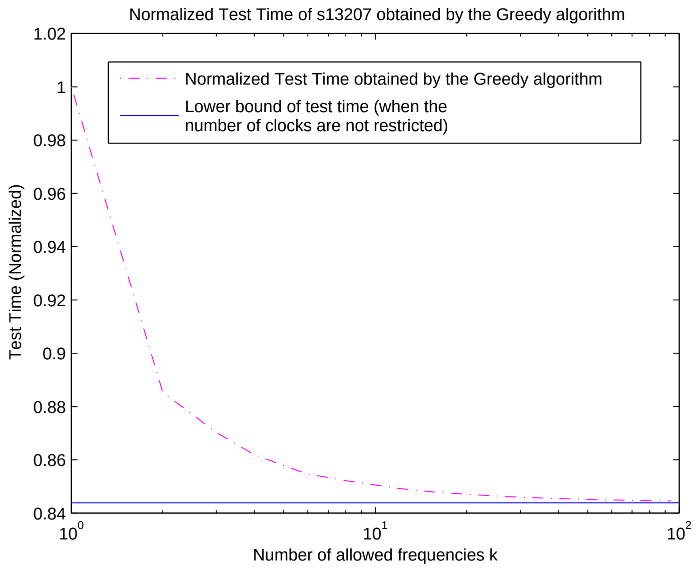


Figure 5.11: Normalized test time of s13207 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum frequencies.

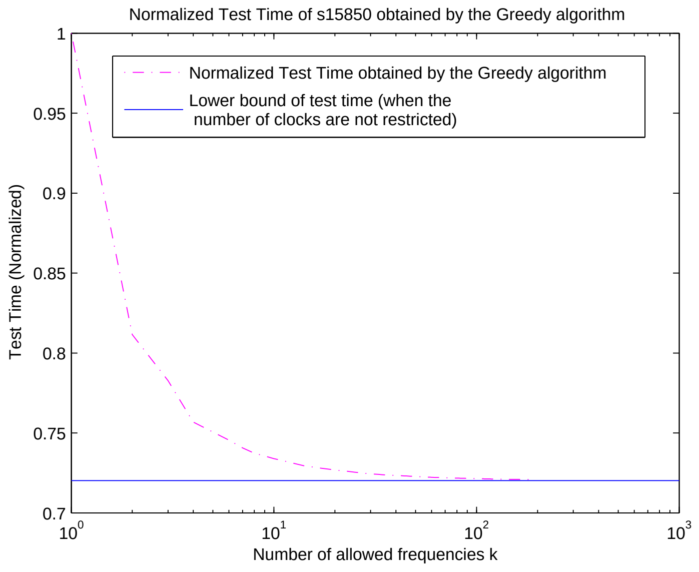


Figure 5.12: Normalized test time of s15850 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum frequencies.

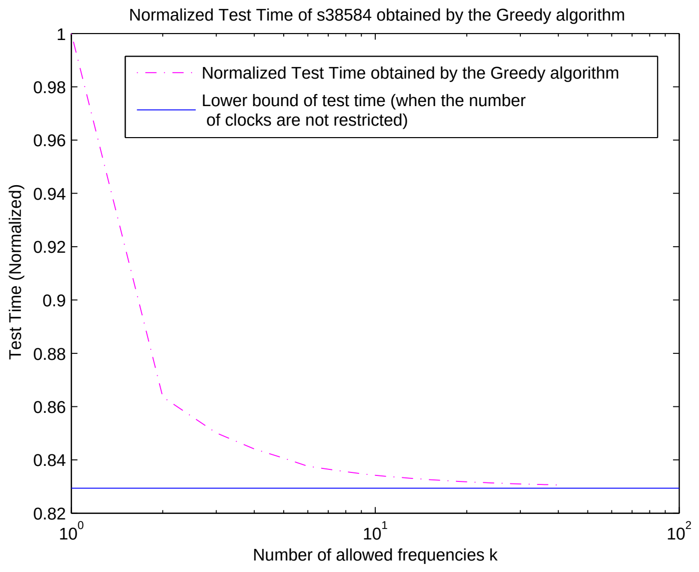


Figure 5.13: Normalized test time of s38584 ISCAS '89 benchmark circuit by the greedy algorithm for k optimum frequencies.

Table 5.4: Percentage of reduction in test time of s713 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)			
		Greedy Algorithm	Iterated Local Search	Direct Search	Simulated Annealing
s713	3	27.10	27.90	27.89	27.90
	4	30.20	30.27	30.25	30.27
	5	31.53	31.89	31.87	31.85
	6	32.47	32.49	32.38	32.37
	7	32.98	33.08	33.02	32.91
	8	33.40	33.43	33.50	33.45
	9	33.72	33.77	33.79	33.64
	10	34.04	34.06	34.05	33.76
	11	34.27	34.32	34.23	34.00
	12	34.48	34.51	34.49	34.24
	13	34.61	34.67	34.64	34.38
	14	34.73	34.78	34.73	34.35
	15	34.83	34.89	34.80	34.64

Table 5.5: Percentage of reduction in test time of s400 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)			
		Greedy Algorithm	Iterated Local Search	Direct Search	Simulated Annealing
s400	3	22.57	22.82	23.10	23.13
	4	25.62	25.66	25.62	25.65
	5	26.66	26.89	26.75	26.82
	6	27.52	27.61	27.51	27.58
	7	28.08	28.08	28.08	27.94
	8	28.43	28.50	28.52	28.35
	9	28.77	28.84	28.74	28.68
	10	29.06	29.13	29.01	28.87
	11	29.35	29.38	29.23	29.12
	12	29.48	29.56	29.42	29.22
	13	29.62	29.68	29.68	29.31
	14	29.74	29.75	29.78	29.27
	15	29.84	29.89	29.89	29.56

Table 5.6: Percentage of reduction in test time of s1238 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)			
		Greedy Algorithm	Iterated Local Search	Direct Search	Simulated Annealing
s1238	3	49.67	50.43	50.43	50.39
	4	51.85	52.37	52.37	52.34
	5	53.02	53.49	53.49	53.43
	6	54.11	54.11	54.22	54.01
	7	54.81	54.81	54.85	54.36
	8	55.14	55.23	55.23	55.23
	9	55.44	55.50	55.53	55.56
	10	55.67	55.75	55.75	55.73
	11	55.89	55.92	55.92	55.89
	12	56.07	56.16	56.13	56.18
	13	56.22	56.27	56.29	56.27
	14	56.35	56.36	56.38	56.37
	15	56.46	56.48	56.50	56.48

Table 5.7: Percentage of reduction in test time of s1423 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)			
		Greedy Algorithm	Iterated Local Search	Direct Search	Simulated Annealing
s1423	3	21.21	21.65	21.65	21.64
	4	23.36	23.51	23.51	23.51
	5	24.33	24.50	24.50	24.48
	6	25.08	25.10	25.10	25.08
	7	25.40	25.44	25.49	25.45
	8	25.69	25.80	25.81	25.75
	9	25.94	26.04	26.04	25.84
	10	26.13	26.23	26.22	26.16
	11	26.31	26.39	26.38	26.31
	12	26.48	26.50	26.52	26.50
	13	26.58	26.63	26.63	26.63
	14	26.68	26.71	26.70	26.71
	15	26.76	26.79	26.78	26.79

Table 5.8: Percentage of reduction in test time of s13207 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)			
		Greedy Algorithm	Iterated Local Search	Direct Search	Simulated Annealing
s13207	3	12.97	13.23	13.23	13.23
	4	13.81	13.94	13.94	13.94
	5	14.21	14.31	14.30	14.31
	6	14.54	14.54	14.54	14.54
	7	14.66	14.70	14.70	14.67
	8	14.78	14.82	14.82	14.82
	9	14.87	14.90	14.92	14.90
	10	14.94	14.98	14.99	14.98
	11	15.01	15.05	15.05	15.06
	12	15.07	15.08	15.10	15.10
	13	15.11	15.13	15.14	15.12
	14	15.15	15.15	15.17	15.16
	15	15.19	15.17	15.17	15.16

Table 5.9: Percentage of reduction in test time of s15850 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)			
		Greedy Algorithm	Iterated Local Search	Direct Search	Simulated Annealing
s15850	3	21.73	22.78	22.83	22.83
	4	24.32	24.32	24.33	24.33
	5	24.94	25.11	25.18	25.18
	6	25.45	25.45	25.71	25.71
	7	25.94	25.94	26.06	26.06
	8	26.26	26.31	26.32	26.31
	9	26.44	26.47	26.51	26.50
	10	26.61	26.70	26.81	26.70
	11	26.73	26.80	26.90	26.80
	12	26.84	26.85	27.00	26.95
	13	26.95	26.97	27.01	27.01
	14	27.04	27.07	27.09	27.07
	15	27.10	27.11	27.10	27.11

Table 5.10: Percentage of reduction in test time of s38584 benchmark circuit obtained by different algorithms.

Benchmark circuit	Number of optimum clocks k	Percentage of reduction in test time (%)		
		Greedy Algorithm	Iterated Local Search	Direct Search
s38584	3	15.02	15.19	14.90
	4	15.58	15.76	15.59
	5	15.92	16.07	15.93
	6	16.24	16.25	16.08
	7	16.35	16.38	16.26
	8	16.35	16.47	16.37
	9	16.44	16.54	16.47
	10	16.52	16.60	16.51
	11	16.58	16.64	16.58
	12	16.62	16.67	16.62
	13	16.66	16.71	16.66
	14	16.69	16.73	16.69
	15	16.71	16.75	16.70

Chapter 6

Conclusion

A greedy algorithm to find optimum clock frequencies for an aperiodic test was proposed and compared with other local search and global search algorithms and was proved to be as good as any of them. The greedy algorithm was found to be faster than all of the other algorithms. The greedy algorithm is also efficient in that it dynamically adds one frequency at each step and can be stopped whenever the required reduction in test time is obtained, or can be monitored for the test time to reach the lower bound. As few as four frequencies can give a test time reduction up to 52% in a few circuits, and the computation could be completed in a time of the order of *milliseconds*. Ten clock frequencies are optimum for test time reduction for most of the benchmark circuits simulated in this work, beyond which the test time reduction is negligible. The aperiodic test was also experimentally verified on the ATE.

6.1 Future Work

The high costs of automatic test equipment (ATE) and the growing clock frequencies bring about the need to study on-chip clock generation circuitry for generation of aperiodic clocks, which can be implemented in BIST circuits where the test patterns are generated on-chip.

Bibliography

[1] *Nanosim User Guide.* Synopsys, San Jose, CA, 2008.

[2] *ATPG and Failure Diagnosis Tools.* Mentor Graphics Corp., Wilsonville, OR, 2009.

[3] *EZWAVE User Guide.* Mentor Graphics Corp., Wilsonville, OR, 2011.

[4] *Leonardo Spectrum User Guide.* Mentor Graphics Corp, Wilsonville, OR, 2011.

[5] N. N. Abdelmalek, "An Efficient Method for the Discrete Linear Approximation Problem," *Mathematics of Computation*, vol. 29, no. 131, pp. 844–850, 1975.

[6] V. D. Agrawal, "Pre-Computed Asynchronous Scan (Invited Talk)," in *13th IEEE Latin American Test Workshop, Quito, Ecuador*, Apr. 2012.

[7] V. D. Agrawal, S. K. Jain, and D. M. Singer, "Automation in Design for Testability," in *Proc. Custom Integrated Circuits Conf.*, (Rochester, N.Y.), May 1984, pp. 159–163.

[8] N. Badereddine, P. Girard, S. Pravossoudovitch, C. Landrault, and A. Virazel, "Minimizing Peak Power Consumption during Scan Testing: Test Pattern Modification with X Filling Heuristics," in *Proc. Int. Conf. on Design and Test of Integrated Systems in Nanoscale Technology*, Sept. 2006, pp. 359–364.

[9] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipe.* lulu.com, first edition, 2012.

[10] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Springer, 2000.

[11] K. M. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis, and G. Hetherington, "Minimizing Power Consumption in Scan Testing: Pattern Generation and DFT Techniques," in *Proc. International Test Conference*, 2004, pp. 355–364.

[12] R. M. Chou, K. K. Saluja, and V. D. Agrawal, "Scheduling Tests for VLSI Systems Under Power Constraints," *IEEE Trans. VLSI Systems*, vol. 5, no. 2, pp. 175–185, June 1997.

[13] R. W. Eglese, "Simulated Annealing: A Tool for Operational Research," *European Journal of Operational Research*, vol. 46, no. 3, pp. 271–281, 1990.

[14] P. Girard, "Survey of Low-Power Testing of VLSI Circuits," in *IEEE Design and Test of Computers*, May-June 2002, pp. 80–90.

[15] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization.* SIAM, 2009.

[16] R. Hooke and T. A. Jeeves, ""Direct Search" Solution of Numerical and Statistical Problems," in *Journal of the Association for Computing Machinery*, volume 8, 1961, pp. 212–229.

[17] J. Kleinberg and É. Tardos, *Algorithm Design.* Pearson Education India, 2006.

[18] T. Kolda, R. Lewis, and V. Torczon, "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods," *SIAM Review*, vol. 45, no. 3, pp. 385–482, 2003.

[19] B. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial Optimization*, volume 1. Springer, 2002.

[20] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search.* Springer, 2003.

[21] S. Luke, *Essentials of Metaheuristics.* Lulu, second edition, 2013. Available at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[22] P. Mangilipally and V. P. Nelson, "Emulation of Slave Serial Mode to Configure the Xilinx Spartan 3 XC3S50 FPGA Using Advantest T2000 Tester," in *Technical report, Auburn University*, 2011.

[23] MATLAB, *version 7.14.0.739 (R2012a).* Natick, Massachusetts: The MathWorks Inc., 2012.

[24] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Courier Dover Publications, 1998.

[25] T. Pavlidis, "Waveform Segmentation Through Functional Approximation," *IEEE Transactions on Computers*, vol. 10, no. 7, pp. 689–697, 1973.

[26] J. R. Rice and K. H. Usow, "The Lawson Algorithm and Extensions," *Mathematics of Computation*, pp. 118–127, 1968.

[27] G. H. Sasaki and B. Hajek, "The Time Complexity of Maximum Matching by Simulated Annealing," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 387–403, 1988.

[28] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer, 2003.

[29] P. Shanmugasundaram, "Test time optimization in scan circuits," Master's thesis, Auburn University, Auburn, Alabama, USA, 2010.

[30] P. Shanmugasundaram and V. D. Agrawal, "Dynamic Scan Clock Control for Test Time Reduction Maintaining Peak Power Limit," in *Proc. 29th IEEE VLSI Test Symposium*, May 2011, pp. 248–253.

[31] P. Shanmugasundaram and V. D. Agrawal, "Externally Tested Scan Circuit with Built-In Activity Monitor and Adaptive Test Clock," in *Proc. 25th International Conf. VLSI Design*, Jan. 2012, pp. 448–453.

[32] V. Sheshadri, V. D. Agrawal, and P. Agrawal, "Optimal Power-Constrained SoC Test Schedules With Customizable Clock Rates," in *Proc. 25th IEEE International System-on-Chip Conf.*, Sept. 2012, pp. 271–276.

[33] V. Sheshadri, V. D. Agrawal, and P. Agrawal, "Optimum Test Schedule for SoC with Specified Clock Frequencies and Supply Voltages," in *Proc. 26th International Conf. VLSI Design*, (Pune), Jan. 2013, pp. 267–272.

[34] C. Stroud, *A Designer's Guide to Built-In Self-Test.* Springer, 2002.

[35] V. V. Vazirani, *Approximation Algorithms.* Springer, 2001.

[36] P. Venkataramani, *Reducing ATE Test Time by Voltage and Frequency Scaling.* PhD thesis, Auburn University, Auburn, Alabama, USA, May 2014.

[37] P. Venkataramani and V. D. Agrawal, "ATE Test Time Reduction Using Asynchronous Clocking," in *Proc. International Test Conf.*, Sept. 2013. Paper 15.3.

[38] P. Venkataramani and V. D. Agrawal, "Reducing Test Time of Power Constrained Test by Optimal Selecction of Supply Voltage," in *Proc. 26th International Conf. VLSI Design*, Jan. 2013, pp. 273–278.

[39] P. Venkataramani and V. D. Agrawal, "Test-Time Reduction in ATE Using Asynchronous Clocking," in *Proc. 6th IEEE International Workshop on Design for Manufacturability and Yield*, June 2013. Poster.

[40] P. Venkataramani, S. Sindia, and V. D. Agrawal, "A Test Time Theorem and Its Applications," in *Proc. 14th IEEE Latin-American Test Workshop*, Apr. 2013.

[41] P. Venkataramani, S. Sindia, and V. D. Agrawal, "Finding Best Voltage and Frequency to Shorten Power-Constrained Test Time," in *Proc. 31st IEEE VLSI Test Symp.*, Apr. 2013, pp. 19–24.

[42] T. Weise, *Global Optimization Algorithms - Theory and Application.* Self Published, 2009-06-26 edition, 2007.