HITEC: A Test Generation Package For Sequential Circuits

Authors: Thomas Niermann and Janak H. Patel

In this paper, the authors present HITEC, a sequential circuit test generation package to generate test patterns for sequential circuits, without assuming the use of scan techniques or a reset state. Different techniques are used to improve performance of test generation. Most importantly, a targeted D element technique is presented, which greatly increases the number of mandatory assignments and reduces the over-specification of state variables, which sometimes result when using a standard PODEM algorithm. Technique to use state knowledge of previously generated vectors for state justification, without memory overhead of a state transition diagram is presented. Knowledge about fault propagation, by the fault simulator, is used for faults that were aborted during the standard test generation phase. This paper presented best results for ISCA89 sequential benchmark, of its time.

This paper presented several techniques to improve performance of sequential test generation. The new concept presented in this paper is the use of fault simulation knowledge to improve fault coverage, and a targeted D element technique for D propagation.

The test detector efficiency that is the ratio of detected faults plus proven untestable faults over the total number of faults is close to 99% for HITEC.

Although HITEC is a successful test generator, it is deterministic and thus slow on large circuits. Compared with Spectral Test, HITEC is harder to implement and requires changes to existing designs. The time of HITEC cannot be compared with Spectral Test algorithms because they were run on different machines. HITEC is most useful in provability of untestable faults than as a test generator. Randomized algorithms are faster at detecting faults in a short time.

Automatic Test Generation Using Genetically-Engineered Distinguishing Sequences

Authors: Michael S. Hsiao, Elizabeth M. Rudnick, and Janak H. Patel

The authors present a fault-oriented sequential circuit test generator, in which various types of distinguishing sequences are derived, both statically and dynamically, to aid the test generation process. A two-phase algorithm is used during test generation. The first phase activates the target fault and the second phase propagates the fault effects from the flip-flops with assistance from distinguishing sequences. This strategy improves the propagation of FE to the primary outputs and the overall fault coverage is greatly increased. The test generator DIGATE, uses genetic algorithms to derive both activating and distinguishing sequences about test generation. They report high fault coverages for ISCAS89 sequential benchmark circuits and several synthesized circuits.

Various types of distinguishing sequences are seeded in the GA to evolve valid distinguishing sequences for the target fault in the fault propagation phase. Pruning of distinguishing sequences is done adaptively during test generation to improve their effectiveness, quantified by the distinguishing power index. The GA uses fitness functions that maximize the number of detected faults, propagation of FE's to flip-flops with powerful distinguishing sequences, fault circuit events and reachable state-space. With the aid of distinguishing sequences, DIGATE achieves high fault-coverage in short execution times and the overall fault coverage is improved significantly compared to results obtained from other test generators.

DIGATE, although produces higher fault coverages, its produces larger sequence lengths. It is more complicated than spectral test methods to implement. Spectral methods have an advantage over fitness function. Methods using fitness function minimize just one function, however spectral methods minimize functions of each PI sequence. Although DIGATE is reported to be faster, the improvements do not rationalize the length of vector sequences.

Sequential Circuit Test Generation Using Dynamic State Traversal

The authors describe a new method of state justification for sequential circuit test generation. A linear list of states dynamically obtained during the derivation of test vectors is used to guide the search during state justification. State-transfer sequences that drive the circuit from current state to the target state is used, when known. Otherwise, genetic-algorithm-based techniques are used to generate valid state justifications sequences for the circuit in the presence of the target fault. This approach achieves extremely high fault coverage and outperforms previously reported techniques.

A test generation framework which utilizes dynamic state traversal for targeting hard-to-test faults were presented. Test generation for a targeted fault is carried in two phases. The first phase excites a fault and propagates its effects the flip-flops; single-time-frame fault activation and state justification using dynamic statement traversal are performed for hard to test faults. The second phase drives the fault effects from the flip-flops to the PO with the aid of distinguishing sequences. The dynamic state-transfer and distinguishing sequences are seeded in a GA to evolve valid state justification and fault propagation sequences, respectively, for the target fault. The GA combines various sequences, to engineer complete solution. Very high fault coverage is obtained in shorter execution times. Significant improvements were observed over previous GA-based approaches. More than 30% improvements in fault coverage, were obtained in some circuits.

STRATEGATE suffers same limitations as DIGATE. It produces long sequences compared with GATEST or HITEC. It does produce higher fault coverage. It is harder to implement than Spectral Test and may not be the best general approach. Sequences produced by STRATEGATE can be compacted by static compactors.

Partitioning and Reordering Technique for Static Test Sequence Compaction of Sequential Circuits

Authors: Micheal S. Hsiao and Srimat T. Chakradhar

They propose new static test set compaction method based on careful examination of attributes of fault coverage curves. They use 2 key ideas: (1) fault-list and test partitioning and (2) vector reordering. Typically, the first few vectors of the test set detect a large number of faults. The remaining vectors usually constitute a large fraction of the test set, but these vectors are included to detect relatively few hard faults. Signification compaction is achieved by partitioning faults into hard and easy faults. This significantly reduces the computational costs for static test set compaction without affecting quality of compaction. The second technique reorders the vectors in a test set by moving sequences that detect hard faults to the beginning of the test set. Fault simulation of the newly concatenated re-ordered test set results in omission of several vectors so that the compact test set is smaller than the original test set. Experiments on several ISCAS89 sequential benchmark circuits and large production circuits show that their compaction procedure yields significant test set reductions in low execution times.

Significant reductions in test set sizes have been obtained using this technique. Furthermore, the partitioning technique rapidly accelerates the compaction process and can easily be used to accelerate existing static compaction algorithms without compromising on the quality of the compaction. Compaction algorithms based on extensive fault simulations can benefit from the partitioning technique. The proposed technique is viable for large circuits with large test sets.

Although this method is reported to be fast, it is not as good as methods by Plomeranz and Reddy. Plomeranz and Reddy's method are slowed, but produce smaller test sequences. Hsiao and Chakradhar's method can be improved significantly. This method should be seen as a basic building block for complicated methods.

Novel Spectral Methods for Built-In Self-Test in a System-on-a-Chip Environment

Authors: Ashish Giani, Shou Sheng, Micheal S. Hsiao, and Vishwani D. Agrawal

The authors present new method of built-in self test(BIST) for sequential cores on a system-on-a-chip(SOC) that generates test patterns using a real-time program that runs on an embedded processor. Alternatively, this same program can be run on an external low-cost tester. This program generates patterns using circuit-specific spectral information in the form of one or more Hadamard coefficients. The coefficients are extracted from high fault-coverage compacted pattern sets. When an embedded processor is available on SOC, the overhead is negligible. Also sequential cores are tested in functional mode, avoiding activation of non-functional timing paths. They present experimental results to show that for hard to test circuits, with any given test time, spectral patterns provide significantly higher fault coverage than weighted random patterns.

The core provider first computes/analyzes the spectral characteristics for the core using the Hadamard transforms. Vector sequences, then are represented as a linear combination of basis vectors. These hadamard coefficients are then used by the embedded processor of the SOC to generate new BIST vectors for the cores/peripherals. Experiments conducted using the technique showed that higher fault coverage can be obtained when compared with a LFSR-based weighted random pattern generation technique, without incurring the overhead of additional test hardware.

The main criticism of this method is that requires an embedded microprocessor. This techniques uses a lot of hardware resources and may not be best suited for applications with small processors or hardware without processors. This technique adds a lot of overhead compared with simple LSFR methods.

State and Fault Information for Compaction-Based Test Generation

Authors: Ashish Giani, Shuo Sheng, Micheal Hsiao, Vishwani D. Agarwal

The author present test generation procedure for sequential circuits using newly travesed state and newly detected fault information, obtained between successive iterations of vector compaction. Two types of techniques are considered. One in based on the new states a sequential circuit is driven into, and the other is based on the new faults that are detected between consecutive iterations of vector compaction. These data modify an otherwise random selection of vectors, to bias vector sequences that cause the circuit to reach new states, and cause previously undetected faults to be detected. The biased vectors, when used to extend the compacted test set, provide an intelligent selection of vectors. The extended test set is then compacted. Repeated applications of state and fault analysis, vector generation and compaction produce significantly high fault coverage using relatively small computing resources. They obtained improvements in terms of higher fault coverage, fewer vectors for the same coverage, or smaller number of iterations and time required, consistently for several benchmark circuits.

They have proposed a new and efficient technique for compaction-based test generation where the information about state reachability and fault detectability is incorporated. The results show that high fault coverages are achieved. In some cases, they detected more faults than have been previously reported. A significant improvement in fault coverage for the b12 and b21 was observed using the both the NSPR/NSPO technique and the NF/TF technique. The number of vectors required to achieve a high level of fault coverage was smaller then other techniques. The CPU time was reported to be lower.

This method preformed better than many others methods and is very simple to implement. However, this method may not be optimal because it uses random perturbation. Random perturbation algorithms are as good as number of samples they deal with. The higher the number of samples, the better they do.

Static Test Compaction for Synchronous Sequential Circuits Based on Vector Restoration

Authors: Irith Pomeranz, Sudhakar Reddy and Ruifeng Guo

The authors propose a new static test compaction procedure for synchronous sequential circuits. The procedure belongs to a class of procedures that omit test vectors from a given test sequence in order to reduce its length without reducing the fault coverage. The previous procedure that achieved high levels of compaction using this approach attempted to omit test vectors from a given test sequence one at a time, or in subsequences of consecutive vectors. The omission of each vector or subsequence required extensive simulation to determine the effects of each omission on fault coverage. The procedure proposed here first omits (almost) all the test vectors from the sequence, and then restores them as necessary to achieve the required fault coverage. The decision to restore a vector requires simulation of a single fault. Thus the overall computational effort of this procedure is relatively low. The loss of compaction compared to the scheme that omits the vectors one at a time or in subsequences is small in most cases. Techniques to speed up restoration process are also investigated, including consideration of several faults in parallel during restoration, and the use of a parallel fault simulator. Experimental results are presented to demonstrate the effectiveness of vector restoration, as a static compaction technique.

They presented a new procedure for static compaction based on omission of test vectors. The previous procedure that achieved high levels of compaction using this approach attempted to omit test vectors from a given test sequence one at a time, or in subsequences. Consequently, the omission of each vector required extensive simulation to determine the effects of each omission on the fault coverage. The procedure proposed here first omits all the test vectors from the sequence, except possibly the ones required synchronizing the fault free circuit. It then restores test vectors as necessary to restore the fault coverage of the original sequence, Restoring a vector is done by simulation of a single fault. Thus, the overall computational effort is significantly reduced compared to the previous procedure. Several improvements to the basic procedure were considered. The improvements help reduce the run time of the procedure, as well as achieve higher levels of compaction. The improvements described to speed up the procedure consisted of consideration of several faults in parallel during restoration. This was achieved by considering all the faults with the same detection times together. A parallel fault simulator was used as part of the compaction procedures. Two orders of processing detection times, as well as a combination of the orders, were considered.

This method is OK but it might not be the most efficient method.

Static compaction using overlapped restoration and segment pruning

The authors propose a new technique for static compaction of test sequences.
Their method is based on two key ideas. (1) Overlapped vector restoration, and (2) identification, pruning, and re-ordering of segments. Overlapped restoration provides a significant computational advantage for large circuits. Segments partitions the compaction problem into sub-problems. Segments are identified, dynamically pruned and reordered to achieve further compaction and speed up. When compared with the fastest method proposed, this method was 5 to 30 times faster on ISCAS circuits and 20 to 50 times faster on large, industrial designs. The new algorithm was able to successfully process large industrial designs that could not be handled by earlier techniques quickly.

The new static compaction technique is proposed that is significantly superior to existing techniques. The new technique is based on vector restoration idea, but the use of segment pruning and overlapped restoration results in a significant improvement. Several acceleration techniques were also discussed. Results from experiments on ISCAS designs and several large industrial designs confirm the practicality of the approach. Ideas of overlapped restoration and segment pruning can also be used in existing static compaction methods to further improve CPU times and compaction quality.

This method was much better than any other static compaction method discussed. This method did it faster and produced sequence lengths similar to other methods, esp methods by Irith Pomeranz and Reddy.

Fast Algorithms For Static Compaction of Sequential Circuit Test Vectors

Authors: Michael S. Hsiao, Elizabeth M.Rudnick and Janak H. Patel

Two fast algorithms for static test sequence compaction are proposed for sequential circuits. The algorithms were based on the observation that test sequences traverse through a small set of states, and some states are frequently re-visited throughout the application of a test set. Subsequences that start and end on the same states may be removed if necessary and sufficient conditions are met for them. The techniques require only two fault simulation passes and are applied to test sequences generated by various test generators, resulting in significant compactions very quickly for circuits that have many revisited states.

A very fast static compaction algorithm is presented. This technique is deterministic, test for specific conditions. Trial and retrial based approaches could take many hours for static compaction even for small circuits; only a few seconds or minutes are needed by this procedure. Inert and recurrence subsequence removal techniques are based on the observation that test sets traverse through many similar states, and compaction is based on eliminating the candidate state-recurrence and inert subsequences inside the test sets. Significant reductions in test set sizes were obtained for HITEC and DIGATE. Moderate results were obtained for GATEST.

This method is fast but not the most efficient when it comes to test sequence lengths. This method must be considered over vector restoration ideas of Irith Pomeranz and Reddy.

PROPTEST: A property-Based Test Generator for Synchronous Sequential Circuits.

Authors: Sudhakar Reddy and Irith Pomeranz

The authors describe a property-based test generator for synchronous circuits. Several techniques are used to generate test sequences that achieve high fault coverage at low computational complexity. These include the use of static test compaction, input vector holding with optimal number of hold cycles, input vector perturbation, and identification of subsequences that are useful in extending the test sequence. Experimental results presented demonstrate that the proposed procedure achieves fault coverages which are in all cases the same or higher than those achieved by existing procedure.

An ATPG procedure for sequential circuits is described. The proposed procedure uses a combination of static test sequence compaction and sequence extension techniques. Sequence extension is achieved by repeating perturbed input vectors or subsequences included in the compacted sequence. The authors experimentally analyzed the reason why holding a random vector will increase the fault coverage. By holding a vector for a several clock cycle, the test sequence can set more flip-flops to one or zero than a pseudorandom sequence of the same length. The improved the likelihood of activating te hard-to-activate faults in the circuit under test. Based on the above observation, the authors proposed a method to determine optimal number of hold cycles for a given circuit based on logic simulation. A “hold truncation on” technique is also proposed to improve efficiency of the hold method. Subsequence template to reach the rare flip-flop values was also proposed to improve the effectiveness of the test generation process. A new method to search for a synchronizing sequence was implemented in the ATPG program. Experimental results presented show that the proposed procedure achieves the highest overall fault coverage for the benchmark circuits studied, while requiring relatively short run times.

PROPTEST is easy to implement and works well. Giani’s method improves on PROPTEST and is able to do a better job than PROPTEST. PROPTEST’s vector holding technique gives good results.

On Static Compaction of Test Sequences for Synchronous Sequential Circuits

Authors: Irith Pomeranz and Sundhakar M. Reddy

The authors propose three static compaction techniques for test sequences of synchronous sequential circuits. They apply the proposd techniques to test sequences generated for benchmark circuits by various test generation procedures. The results show that the test sequences generated by all the test generation procedures considered can be significantly compacted. The compacted sequences thus have shorter test application times and smaller memory requirements. As a by-product, the fault coverage is sometimes increased as well. More importantly, the ability to significantly reduce the length of the test sequences indicates that it may be possible to reduce test generation time if superfluous input vectors are not generated.

All three techniques yield high compaction. Significant reductions in test length are obtained, even in cases, which use memory-intensive dynamic compaction to produce test sequences that are already very short. In many cases, an increase in fault coverage is also obtained. In most cases, one of the procedures finished after 1 or 2 iterations before no additional vectors could be removed.

This paper is well laid out. The methods proposed in this paper may not be most optimal. Segment pruning may be the better way of doing business.

On Random Pattern Generation with the Selfish Gene Algorithm for Testing Digital Sequential Circuits

Authors: Junwu Zhang, Michael Bushnell, Vishwani D. Agrawal

A selfish gene(SG) algorithm is presented. Unlike genetic algorithm(GA), it evolves genes(characteristics) that provide higher fitness rather than evolving individuals with higher fitness. They enhance the spectral method of sequential circuit test generation by using a SG algorithm. The objects of evolution are the Hadamard spectral matrix, non-linear digital signal processing(DSP) filtering cut off values, vector holding time, and relative input phase shifts, which are all modeled as genes. These characteristics, extracted from compacted test vectors, are used to create new vector sequences to be further compacted with high fault coverage. Alternatively, new vectors are generated by holding randomly selected vectors and then randomly perturbing some bits in 8-bit chunks of bit streams, Both the SG algorithm and holding with bit-perturbation can outperform the previously published spectral method in either fault-coverage, or shorter vector length or both. The SG algorithm is often superior to random bit perturbation but it requires more CPU time.

The methods use longer CPU time than the Pure Hardamard method by Giani et al. They use omission-based vector compaction in the last iteration of each experiment and LROR in earlier iterations. The SG algorithm is slower than Giani and al's algorithm. For a large subset of the ISCAS '89 benchmaks, the SG spectral algorithm detected 3.44% more faults than the best of existing deterministic algorithms, using 74.2% fewer vectors. For 22 of 30 ISCAS '89 circuits, the SG spectral algorithm gave good results.

The method may not be feasible on large circuits, due to use of omission based compactor. The method is pretty random and difficult to implement. This method has too many details and results may not be relatively reproducible.

Efficient Spectral Techniques for Sequential ATPG

Authors: Ashish Giani, Shuo Sheng, Michael S.Hsiao and Vishwani D. Agrawal

The authors present a new test generation procedure for sequential circuits using spectral techniques. Iterations of filtering via compaction and spectral analysis of the filtered test set are performed for each primary input, extracting inherent spectral information embedded within the test sequence. This information when viewed in the frequency domain, reveals the characteristics of the input spectrum. These spectral characteristics are then used to generate future vectors. The authors develop a fault-dropping technique to speed up the process. We show that very high fault coverages and small vector sets are consistently obtained in short execution times for sequential benchmark circuits.

The authors have presented a novel spectral technique for test generation. Static compaction is first used to filter unwanted vectors. Then, Hadamard transforms is used to analyze input spectra. This technique identifies inherent periodicity of the input bits. Vector sequences, then can be represented as a linear combination of the basis vectors. Experiments conducted using this spectral technique showed that very high fault coverage with small test sets can be rapidly obtained in a few iterations. Together with fault dropping, a considerable computation cost reduction is achieved. In circuits such as b12, they achieved a noteworthy increase in the fault coverage when compared to any previously reported technique.

This method is good. It produces high fault coverage with smaller vector lengths. Filtering may not be the best use of Spectral methods.

Sequence Reordering to Improve the levels of Compaction Achievable by Static Compaction Procedure

Authors: Irith Pomeranz and Sudhakar Reddy

The authors describe a reordering procedure that changes the order of test vectors in a test sequence for a synchronous sequential circuit without reducing the fault coverage. They use this procedure to investigate the effects of reordering on the ability to compact the test sequence. Reordering is shown to have two effects on compaction. (1) The reordering process itself allows the authors to reduce the test sequence length. (2) Reordering can improve the effectiveness of an existing static compaction procedure. Reordering also provides insight into the detection by test generation procedures of fault that are detected by relatively long subsequences.

Test sequence reordering consists of changing the order of test vectors in a test sequence for a synchronous sequential circuit without reducing fault coverage. They investigated the effects of reordering on the ability to compact test sequence. Reordering was shown to have two effects on compaction. The reordering process itself allowed the TG to reduce the test sequence length. Reordering was shown to improve the effectiveness of existing static compaction procedures, especially when applied as a preprocessing step to compaction. Reordering was done by arbitrarily partitioning the sequences into equal or almost equal subsequences. Combining some of these subsequences if necessary to detect certain faults, and then finding a permutation of subsequences that allows the original fault coverage to be maintained. They found that there were several permutations of the subsequences that result in the same fault coverage as the original sequence. Thus reordering also provided an insight into the detected by test generation procedure of faults that require relatively long subsequences in order to be detected.

This method is not fast compared with methods by Chakradhar and others. It could be significantly improved.

On Improving a Fault Simulation Based Tes Generator for Synchronous Sequential Circuits.

Authors: Ruifeng Guo, Sudhakar M. Reddy, Irith Pomeranz

The authors propose several techniques to improve as simulation based test pattern generation procedure for sequential circuits. The effectiveness of the proposed techniques is demonstrated through experimental results on a large set of benchmarks.

In this paper, the authors describe several techniques to improve the ATPG procedure PROPTEST. They experimentally analyze the reason why holding a random vector will increase the fault coverage. By holding a vector for several clock cycles, the test sequence can set more flip-flops to one or to than a pseudo-random sequence of the same length. This improves the likelihood of activating the hard-to-activate faults in the circuits under test. Based on the above observation, they propose a method to determine the number of hold cycles for a given circuit based on logic simulation. A "hold truncation" technique is proposed to improve effectiveness of the hold method. Subsequence template to reach the rare flip-flop values is also proposed to improve the effectiveness of the test generation process. A new method to search for a synchronizing sequence is implemented in the improved ATPG program. Experimental results using benchmark circuits and synthesized circuits demonstrate the effectiveness of the improved ATPG program

The methods are simple to implement and the coverages are comparable to other methods.