MEMORY SHARING & UTILIZATION AMONG MULTIPLE PROCESSING UNITS TO IMPROVE THE PERFORMANCE OF ATPG PROCESS THROUGH HDL

Mr. A.CHAKRADHAR[1] , Mrs.A.KARUNA SRI[2]
1. Associate Professor, Department of ECE ,Jayamukhi Institute of Technogical Sciences , Warangal, India
2. Department of ECE, Jayamukhi Institute of Technological Sciences, Warangal, India.

Abstract:A new test generation methodology is proposedthat takes advantage of shared memory multi-core systems. Appropriateparallelization of the main steps of ATPG allocatesresources in order to minimize workload duplication and multithreadingrace contention, often encountered in parallel implementations.Recent works on ATPG parallelization for on-chip multicoreenvironments exploit a variety and, often mixture, of parallelism dimensions such as fault parallelism, structural (circuit) parallelism, and algorithmic (including search-space) parallelism. Moreover, the goal of utilizing parallelism often varies.The proposed approach ensures that the obtainedacceleration grows linearly with the number of processing coresand, at the same time, keeps the test set size close to that obtainedby serial ATPG. The experimental results demonstrate that theproposed methodology achieves higher degree of speed-up thancomparable state-of-the-art multi-core based tools, while maintainssimilar test set sizes.

Keywords—ATPG; Parallel Processing; Multi-core systems

Introduction

Generation of test patterns for combinational logic is a search through the set of all input values to find one that causes the output of a good circuit to differ from that of one containing a fault. Much research has gone into increasing the efficiency of algorithms for ATPG. However, the overall gains achieved through these improvements have not kept pace with increasing circuit size, and computation times are still excessive. This report surveys techniques now being explored to map the ATPG to parallel processing machines. As the size and complexity of IC's continue to grow, the need for fast and effective testing methods for these devices becomes even more important. A significant portion of design time for IC's and digital systems in general, is spent in generating test patterns that distinguish a faulty IC from a fault free one. In order to keep defective products from reaching the market, manufacturers must be able to test their product in an efficient and cost effective manner.

Technology shrinking in the integrated circuit manufacturing process allowed the implementation of multiple processing units (cores) on a single chip as well as large amounts of on chip memory.

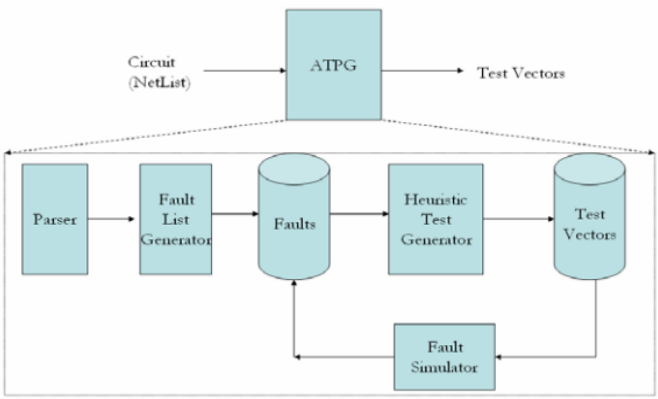

Figure 1: Components of ATPG.

These developments offer extensive processing power that can be used in various computationally intensive problems including popular electronic design automation processes. However, the distributed fashion of this processing power guides towards the development of parallel methodologies that scale well as the number of cores per chip are expected to increase beyond two dozens to hundreds. Automatic Test Pattern Generation

(ATPG), a well-known NP-hard problem, becomes more demanding as devices under test are becoming larger and more complicated and as emerging defects require new fault models of higher complexity.

As a means to increase the testability of the circuits and also to reduce the Automatic Test Pattern Generation (ATPG) complexity, Design-For-Test (DFT) methods are employed. Two main parameters that determine the testability of a circuit are the controllability and observability of its signals. Controllability of a signal refers to its ability or ease to be set to a particular logic value from the primary inputs of the circuit. Observability of a signal refers to its ability or ease to be observed at one of the primary outputs of the circuit.

Automatic Test Pattern Generation (ATPG), a well-knownNP-hard problem, becomes more demanding as devices undertest are becoming larger and more complicated and as emerging defects require new fault models of higher complexity.

While previously proposed procedures are very effective, among many others, they are inherently non-parallel and thus, cannot rely on automatic parallelization using sophisticated compilers. Proper problem decomposition, workload distribution and final test set re-composition are essential to guarantee the quality of the results while maintaining fault coverage. Since, typically, each core does not consider the entire search space, parallel approach tend to choose local optimal solutions resulting in test set Increase, known as the test inflation problem. Parallel ATPG has been studied before the on-chip multi core era, by either applying bit level parallelism or distributing ATPG components among multiple processing units, not physically on the same chip. These approaches were designed to avoid/minimize communication overhead and were constrained by the machine's word size.

Parallelization speed-up rates and test set inflation are investigated in the recent work of which also considers ashared memory architecture model. Shared memory is utilizedas an extremely low latency communication mean with highcapacity to leverage synchronization and communication ofthe process.

Test generation process

Testing digital circuits must include the two classes of digital circuits: combinational and sequential. For combinational logic circuits, only one test vector sequence is required for stuck-at fault detection. Sequential circuits inherently require the application of a series of test vector sequences for the detection of a fault. Hence, combinational testing is a subset of the sequential test problem. Most sequential test algorithms map the generation of test sequences to iterative combinational test methods. Some techniques allow for the conversion of sequential circuits to combinational circuits for the purpose of testing. This conversion reduces the complexity of test generation for a sequential circuit to that of combinational logic. Therefore, efficient combinational test algorithms are needed to reduce the time spent in test.

Test generation can be achieved either by deterministic test pattern generation or by statistical test pattern generation. Deterministic test pattern generation uses a specific algorithm to generate a test for every fault in a circuit, if a test exists. Statistical test pattern generation randomly selects test vectors, and using fault simulation, determines which faults are detected.

This statistical method can quickly find tests for the easy-to-detect faults, but becomes significantly less efficient when only the hard-to-detect faults remain. Deterministic test pattern generation uses one of numerous Automatic Test Pattern Generation (ATPG) algorithms. ATPG algorithms provide a mechanism to generate a test vector for a specific fault, and fault simulation algorithms are available which can determine if any additional faults are covered by a given vector. As a result, it is now possible to test large circuits within a reasonable period of time

In addition to using algorithmic techniques to improve the efficiency of ATPG, parallel processing environments can be utilized to reduce computation time. There are several methods available to parallelize ATPG. These methods include fault partitioning, heuristic parallelization, search space partitioning, algorithmic partitioning, and topological partitioning of these methods, the simplest toimplement is fault partitioning, which divides the fault list across various processors. It is this method of parallelization that is the basis of this investigation.

A common parallelization procedure consists of three steps:(i) decomposition (domain or

functional), (ii) parallel execution,and (iii) final result assembly. Step (ii) can result in significantcompromise of the quality of the obtained results and,at the same time, not offer the expected speed-up. An efficientparallel algorithm should effectively overcome challengessuch as memory contention and imbalanced workload distribution. The proposed ATPG method appropriately designs allthree steps to ensure that these challenges are treated efficiently.

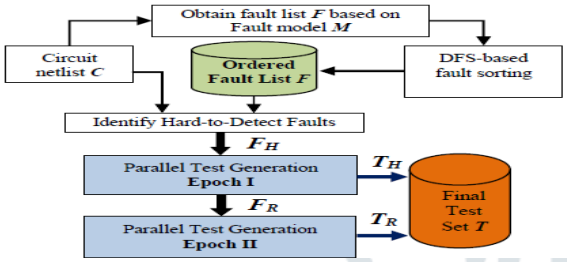

Fig.2.Test generation process

The proposed ATPG method appropriately designs allthree steps to ensure that these challenges are treated efficiently.Specifically, two conceptual approaches are adopted: (i)problem partitioning to avoid executing the same work concurrentlyin different cores and (ii) fine-grained granularity ofeach step to provide dynamic distribution of work. Variousparallel optimization heuristics based on this concept.

The methodology takes advantageof fast and low cost shared memory communication inherentin the underlying architecture in order to coordinate the mainsteps of the ATPG to avoid redundant work and dynamicallyallocate the workload while minimizing memory contentioncaused by multiple cores (threads) when accessing shared data.A test generation flow is proposed in which hard-to-detectfaults are targeted first, followed by a parallel fault simulationbasedmerging process to maximize fault coverage. This processemploys a series of newly proposed parallelization heuristicsto explicitly avoid simultaneous consideration of thesame faults by two or more cores, in order to minimize extrawork and thread idle time. Any remaining undetected faultsare targeted during a following phase, in a similar manner.The obtained experimental results demonstrate the effectivenessof the proposed approach in speeding-up the ATPG process.

Shared Memory Architecture

There are two types of parallel processing architectures.

- Shared Memory Architecture
- Message Passing Architecture.

These two differ in their memory organization, resulting in different speed and communication. Programs written for one type of architecture might not perform well when executed on the other architecture.

Shared Memory Systems: Shared memory systems have single global memory which can be accessed by all processors. Processors have their own caches but the address space is the same. A major characteristic of most shared memory systems is that access to data is independent of the processor making the request and is relatively fast, almost as fast as typical memory access times in a uniprocessor system. However, when many processors are making simultaneous requests to a single memory location or bank, and memory access becomes a bottleneck, access times can increase greatly. For this reason, physical memory layout and data organization within the memory are critical to ensure that the memory system can handle as many simultaneous requests as possible.

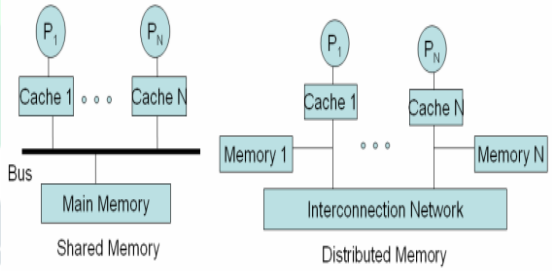

Figure 3:Shared Memory andDistributedMemory system architectures.

Parallelization methodology and optimizations:

A. Test-Epoch Parallelization:

Fig.4 presents a flowchart illustrating the basic steps of the parallel Test Generation (TG) methodology followed during a test epoch, namely *seed-based TG* and *dynamic test mergingand restricted TG*. An epoch explicitly targets on a fault-byfaultbasis, only a small subset of the fault list F (FH forEpoch I and FR for Epoch II). Note that $FC = F - (FH \cup FR)$typically constitutes the overwhelming majority of the faultswhich are easily detectable in an implicit manner (i.e., viafault

simulation). The faults in a fault list are sorted based onstructural similarities of fault locations (net list), in order toincrease the probability of proximate faults to be detected bythe same test.

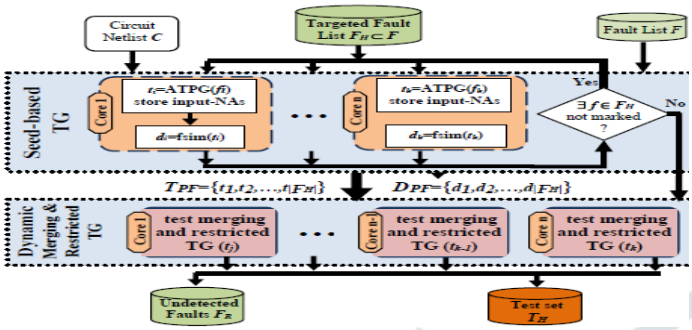

Fig.4. A test epoch targeting hard-to-detect faults (Epoch I).

B. Parallelization Optimizations

Detection-based Primary Test Selection. In the merging step of Fig.5, test selection is very important for the efficientevolution of merging since it sets the constraints and outcomesof consequent merging iterations, restricted TG, and faultsimulation. Practice in ATPG suggests that early faultdropping plays a more important role than having fewerconstraints (more unspecified bits) in the test seed. For thisreason, the *primary test* ***ti*** during dynamic merging (mergingseed) is selected based on its number of detected faults in ***di***.Recall that the fault simulation process performed at the endof the first step of the test epoch.

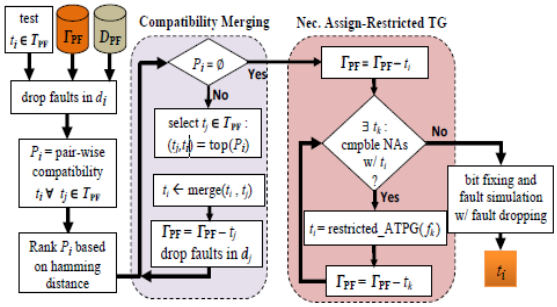

Fig.5. Dynamic Test merging and Restricted TG processes per core

Test Set Private Consideration.

The search for the best candidate tests to be merged (either the primary or the ones tofollow) involves high interaction of each core with the shared memory. Specifically, selecting the primary test, as well as computing the pair-wise compatibilities with the remaining tests in **TPF, inherently** involves memory contention since all cores are searching **TPF.** This issue is addressed by dynamically partitioning **TPF** in **n** private subsets (**n** being then number of available cores), one for each core. Each core can only select tests from its own private subset of **TPF** (and the corresponding **DPF**) which can be safely moved to its own private cache. This implicitly minimizes concurrent memoryaccess requests from different cores that can result ininefficient memory utilization due to memory contention. Moreover, it implicitly minimizes duplication of work as each core considers a distinct part in **TPF**. When a core finishes with themerging process within its private part of **TPF**, it is allowed to work on the entire set in order to ensure workloadbalancing by avoiding idle periods in cores. At this point, concurrent memory accesses can occur, however, their impact is minimal as the bulk of the merging process has alreadyoccurred during the private consideration, and, hence, the size of **TPF** is by this point significantly reduced.

Test Provisional Marking

During compatibility merging, the list **Pi** which holds pair-wise compatibilities between testsrequires updating after each merging. This updating is highly demanding in processing resources as it is of cubic complexity in the worst case. To avoid this issue the proposed methodology calculates and ranks compatibilities only once for each test **ti**. If a test **tj** is selected to be merged with **ti**, it is provisionally marked in **TPF** so that it is not merged in anothercore, explicitlyavoiding imposing unnecessary constraints inanother thread that performsmerging. If compatibility between **ti** and a test **tj** in **Pi** is invalidated by a previous merging, merging between **ti** and **tj** is not completed and theprovisional marking is cleared. Otherwise, provisional marking indicates permanent discarding of **tj** from **TPF**.

Balanced Workload Distribution:

Distribution of workload to the available cores can significantly impact the speed-up of a parallel methodology. Test generation and fault simulation processes have unpredictable execution times due to the nature of the problems and fault

dropping. Core idle time is minimized by dynamically selecting:'

(i)The next fault to be targeted in seed-based test generation in each epoch.

(ii) The next test to be used as primary in test merging.

(iii) The tests to be merged with the primary test seed, and

(iv) The next fault to be targeted in restricted TG based on necessary assignments. Since, data is stored in shared memory (fault list and test seeds), and thus, is easily accessible by all cores, provides a punctual way ofdetermining how the workload will be selected at each stepand by each optimization mechanism of the approximation.

Simulation results

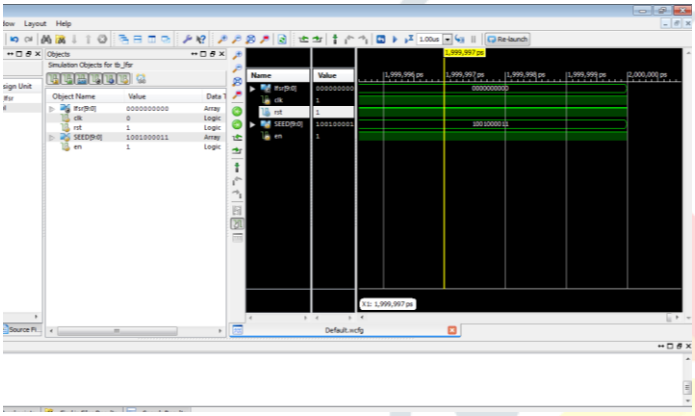

Fig.6.Simulated output

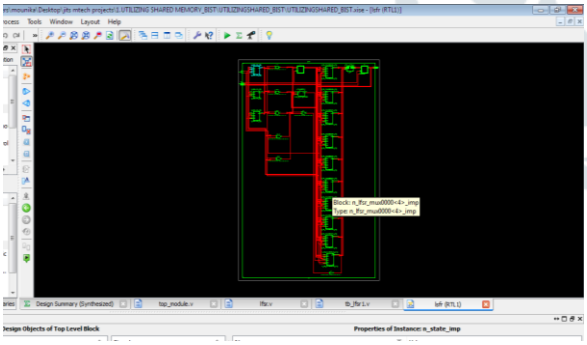

Fig.7.RTL schematic of proposed method

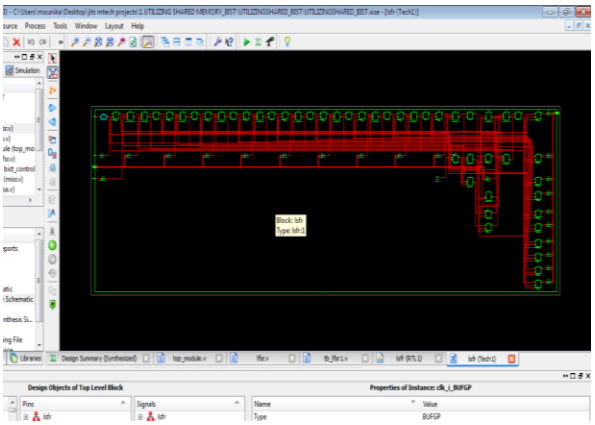

Fig.8.Technology schematic of proposed method

CONCLUSION

Proposed a parallel test pattern methodology for shared memory multi-core environments. A number of newly proposed heuristics attempt to avoid assigning the same workload to multiple cores, while the distribution of work in the available resources to minimize core idle time. Experimental results demonstrate high speed-up rates that keep increasing as the number of the available cores increases. Test set size increase is limited and comparable to other state-of-the-art parallel approaches.

REFERENCES

[1] K. Scheibler, D. Erb and B. Becker, "Improving test patterngeneration in presence of unknown values beyond restricted symbolic logic," in Proc. of *ETS*, pp. 1-6, 2015

[2] S. Eggersglub, K. Schmitz, R. Krenz-Baath and R. Drechsler, "Optimization-based multiple target test generation for highly compacted test sets," in Proc. of *ETS*, pp. 1-6, 2014.

[3] I. Pomeranz, "Generation of compact multi-cycle diagnostic test sets," in Proc. of *ETS*, pp. 1-1, 2013.

[4] S.Patil and P.Banerjee, "Fault partitioning issues in an integrated parallel test generation/fault simulation environment," in Proc. of *ITC*, pp. 718–726, 1989.

[5] J. Wolf, L. Kaufman, R. Klenke, J. H. Aylor, and R. Waxman, "An analysis of fault partitioned parallel test generation," *IEEETrans. on CAD*, vol. 15, no. 5, pp. 517–534, 1996.

[6] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. M. Reddy and B. Becker, "Thread-parallel integrated test pattern generator utilizing satisfiability analysis," *International Journal*

ofParallel Programming, vol. 38, pp. 185-202, 2010.
[7] K-Y. Liao, C-Y. Chang and JC-M Li "A parallel test pattern generation algorithm to meet multiple quality objectives," *IEEETrans. on CAD*, vol.30, no. 11, pp. 1767-1772, 2011.
[8] K.-W. Yeh, M.-F. Wu and J.-L. Huang, "A low communication overhead and load balanced parallel ATPG with improved static fault partition method," in Proc. of *Intl. Conf. on Algorithmsand Architectures for Parallel Processing*, pp. 362-371, 2009.
[9] JC-Y. Ku, RH-M. Huang, LY-Z. Lin and CH-P. Wen,"Suppressing test inflation in shared-memory parallel Automatic Test Pattern Generation," in Proc. of *ASP-DAC*, pp.664-669, 2014.
[10] K-W. Yeh, J-L. Huang, H-J. Chao and L-T. Wang "A circular pipeline processing based deterministic parallel test patterngenerator," in Proc. of *ITC*, pp. 1-8, 2013.
[11] X. Cai, P. Wohl, J.A. Waicukauski and P. Notiyath "Highly efficient parallel ATPG based on shared memory," in Proc. OfITC, pp. 1-7, 2010.