GATTO*: a PVM-based Distributed Test Pattern Generation Algorithm for Large Sequential Circuits

Fulvio CORNO, Paolo PRINETTO,
Maurizio REBAUDENGO, Matteo SONZA REORDA

Politecnico di Torino
Dipartimento di Automatica e Informatica

Enzo VEILUVA

Centro Supercalcolo Piemonte

Due to the continuous increase in the size and complexity of VLSI circuits, Automated Test Pattern Generation (ATPG) [ABFr90] is now a major problem from the industrial and economic point of view. As far as the single stuck-at fault model is considered, efficient algorithms have been devised for combinational and small sequential networks. Very large sequential circuits, however, still constitute a problem. None of the methods proposed so far can fully handle all the real-world circuits test engineers have to deal with, since they either definitely get off or generate sequences with very low fault coverages.

Previous attempts of resorting to *parallel* and/or *distributed systems* to enhance capabilities in terms of CPU time required and maximum circuit size are reported in the literature: [RaBa92] describes a parallel/distributed version of a topological ATPG for sequential circuits; a distributed version of a commercial ATPG tool is reported in [AAVi93]. However, we believe that both proposals do not correctly address the portability issue, as they either do not guarantee any code portability, or rely on an ad hoc software platform to solve this problem.

A new mono-processor approach to the ATPG problem, GATTO, has been presented in [PRSR94]: the method is based on Genetic Algorithms [Holl75] and it is organized in three phases: in the first one a target fault is chosen; in the second a test sequence is generated for the target fault using a Genetic Algorithm; in the last phase the generated sequence is fault simulated in order to detect additional faults. Experimental results show that the method compares well with the previously poposed ones, and that it provides good results with reduced CPU times, even for the largest benchmark circuits.

This paper describes a PVM-based distributed version of the same algorithm, named GATTO*, and discusses the advantages of the approach.

The heaviest task in the mono-processor algorithm is the computation of the evaluation function for all the generated sequences. Therefore, the distributed algorithm is based on dynamically assigning to each processing node the computation of the evaluation function of a sequence. This corresponds to running a modified version of a fault simulator; we exploit the efficiency of the state-of-the-art algorithm described in [NTPa92].

A prototypical tool implementing the algorithm has been developed using PVM; the tool, amounting to about 3,500 lines of C code, has been experimented on a network composed of 16 DEC Alpha AXP 3000/500 workstations; the standard set of benchmark sequential circuits [BBKo89] has been used.

Being based on the PVM library, GATTO* has the following characteristics:

- it can run on any workstation network;
- it is portable over a wide range of architectures, including both distributed systems and several parallel machines (it is currently being experimented on a *CM-5*);
- it will benefit from further developments in workstation architecture.

Thanks to the versatility of the underlying Genetic Algorithm, the computational power of the network can be exploited in two directions:

- to reduce the time, while maintaining the same values for the parameters, and thus the same results in terms of fault coverage and sequence length;
- to improve the result quality with virtually the same time requirement.

When the former approach is followed, we obtained a speed-up with respect to the mono-processor version which is graphically drawn in Fig. 1: thanks to the good load balancing and the reduced communications, the speed-up always ranges from 8 to 13.

On the other side, when the latter approach was followed, GATTO* was able to provide significant improvements in result quality. For most circuits, these are the best results that have so far appeared in the literature.

We are currently working towards running the same code on a *CM-5* system, and preliminary data show that the results are comparable with the ones obtained on a workstation network, provided that the differences in the processor power are taken into account.

As a conclusion, this work shows how a state-of-the-art algorithm for ATPG can be ported to a workstation network, and demonstrates that the distributed approach is effective for reducing the CPU time required and improving the result quality. The adoption of Genetic Algorithms allows the user to easily trade-off CPU time and result quality, and makes the algorithm particularly suitable for parallelization. The use of the PVM library guarantees the real portability of the code without any significant cost in terms of additional code and speed overhead. Work is currently done to evaluate the portability and effectiveness of the same code on the CM-5 system.

References

[AAVi93] P. Agrawal, V.D. Agrawal, J. Villoldo, "Sequential Circuit Test Generation on a Distributed System," *Proc. 30th Design Automation Conference*, 1993, pp. 107-111

[ABFr90] M. Abramovici, M. A. Breuer, A. D. Friedman, "*Digital systems testing and testable design*," Computer Science Press, New York, NY (USA), 1990

[BBKo89] F. Brglez, D. Bryant, K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Proc. Int. Symp. on Circuits And Systems*, 1989, pp. 1929-1934

[Holl75] J.H. Holland, "Adaption in Natural and Artificial Systems," University of Michigan Press, Ann Arbor (USA), 1975

[NTPa92] T.M. Niermann, W.-T. Cheng, J.H. Patel, "PROOFS: A Fast, Memory-Efficient Sequential Circuit Fault Simulator," *IEEE Transactions on Computer-Aided Design*, Vol. 11, No. 2, pp. 198-207, February 1992

[PRSR94] P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms," *Proc. International Test Conference*, 1994

[RaBa92] B. Ramkumar, P. Banerjee, "Portable Parallel Test Generation for Sequential Circuits," *Proc. Int. Conf. on Computer Aided Design*, 1992, pp. 220-223

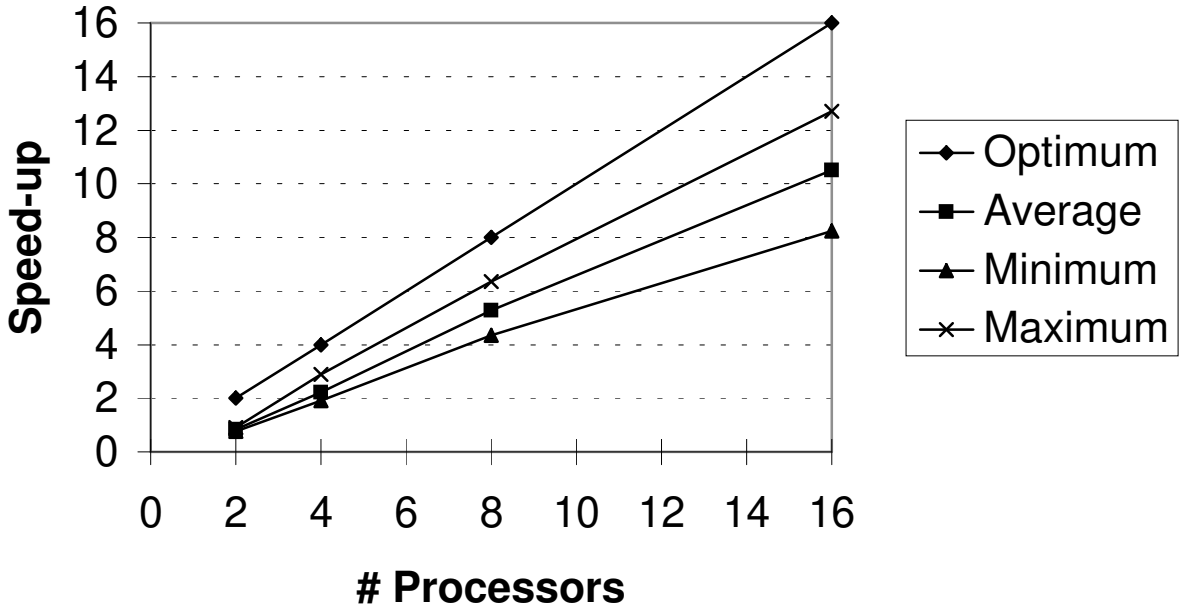

Fig. 1: Speed-up.