# MONTE CARLO BASED SSTA ON GRAPHIC PROCESSING UNITS ON FPGAS

Srinivas
Assistant Professor
Murthi College of Engineering

***ABSTRACT:*** **A novel acceleration scheme for Monte Carlo based statistical static timing analysis. The Monte Carlo technique is a well known solution for statistical analysis. We drew upon advancements in improving the efficiency of Monte Carlo based static timing analysis using technique to reduce the sample size. This is implemented on GPU based on NVIDIA CUDA architecture. In our approach timing graph of a target net list will be translated into RTL description that can be mapped into FPGA dedicated STA engine. We show that SH-QMC implemented on a multi GPU is twice as fast as a single sta o a cpu for bench mark circuits considered.**

## I INTRODUCTION

Increasing effects of process variations have strengthened the case for a move from static timing analysis (STA) to statistical static timing analysis (SSTA). SSTA aims to use statistical techniques to analyze the timing performance of a chip in the presence of statistical variations in underlying parameters. Various approaches for SSTA have been suggested. These are recent surveys of the field. We defer a discussion of related work to Section V. Analysis of timing performance boils down to a study of the distribution of the worst case circuit delay. An important feature of the delay distribution estimation problem is that we are typically interested in the right-hand tail of the distribution, i.e., the tail corresponding to large delay values (see Figure 1). Given a model, we seek statistical information about the 'bad' (timing violating) region in the parameter space. Recent years have seen the rapid scaling of throughput-optimized processors, such as Graphics Processing Units (GPUs). Modern GPUs deliver over 1 Tera Flops of computational power with more than 100 GB/second of memory bandwidth while conventional processors face difficulties with frequency scaling and are increasingly incorporating multiple cores on a chip to keep up with Moore's law. Throughput processors recognize two crucial aspects of machine organization which are parallel execution and hierarchical memory organization. To increase performance in throughput processors, applications will need to expose parallelism while finding locality in their computations to overcome restrictions arising from communication bandwidth bottlenecks. In this work we show the importance of these two aspects for improving performance and efficiency in the context of statistical timing analysis by drawing inferences from the implementation on a specific GPU architecture. in this paper a novel scheme for accelerating MC-SSTA is proposed. in this approach we utilize a generalized STA engine called an STA processing element (STA-PE) which can calculate the latest or earliest arrival time of one logic gate with in a clock period. In the case of MC-SSTA, however, since each Monte Carlo runs can be executed independently, it is possible to accelerate it dramatically with dedicated delay-sample generator for each logic gate. Motivated by this, we propose an acceleration of MC-SSTA using pipeline scheme. Using the proposed scheme, a target net list to be analyzed will be translated into synthesizable RTL description and then mapped into FPGAs. Figures 1 and 2 show the proposed analysis flow and an example of the transformation from a target net list into a MC-SSTA implementation on FPGAs, respectively. Using our proposed implementation, fully pipelined MC-SSTA dedicated to a given net list is realized. The SSTA problem is (increasingly) high dimensional. Moreover, depending on the variation model, it can have a large number of 'bad' regions. For example, consider a model with independent gate delay variations. The number of important dimensions scales almost linearly with circuit size (one dimension for each gate that lies on *some* critical path). Further, there are a huge number of 'bad' regions, corresponding to different paths being critical, that are not connected (for disjoint paths) or very weakly connected (for overlapping paths) to each other. In worst case, this number of 'bad' regions can grow exponentially corresponding to the explosion of critical paths. Thus, the problem is 'hard', making it difficult to obtain large improvements over simple Monte Carlo sampling. Techniques such as Statistical Blockade (SB) [5] and importance sampling (e.g. [6]) used for other rare event problems, notably SRAM failure analysis, are not useful for SSTA as discussed in Section V. Markov chain Monte Carlo is a popular approach for estimating distributions that are hard to compute exactly. In particular, it can be used effectively for solving a variety of rare event problems, including those with high dimensionality, and a multitude of 'bad' regions. We demonstrate the power of MCMC in this work by applying it to the SSTA problem. Our algorithm retains the well-documented advantages of Monte Carlo over other approaches for SSTA, while exploiting the large deviation nature of problem.

## II STATIC TIMING ANALYSIS (STA)

Static timing analysis is one of the most widely used timing analysis methods for designing synchronous digital circuits. In STA critical paths in a given chip design are identified by propagating the latest arrival time over all logic gates in the circuit. one of the biggest advantages of STA is its speed; computational complexity is O(N) when the number of simple graph tracing. conservative timing is calculated without any input vectors. Conservatism ,however, is one major drawback of STA.with modern process technologies, the delay variation of each logic gate due to process, voltage, and temperature variations has to be properly considered. a worst case analysis has traditionally been employed to cope with his problem.

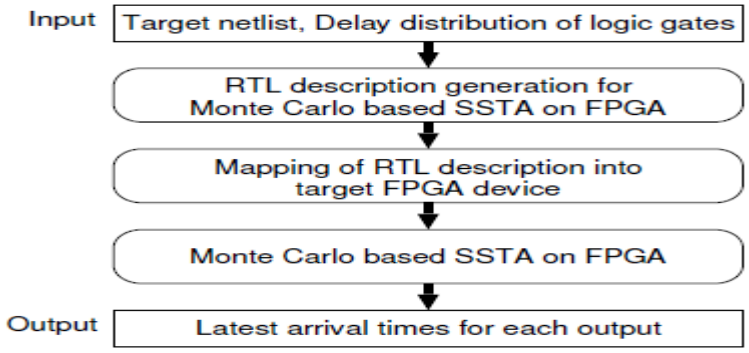


Fig 2.0 MC_SSTA Flow

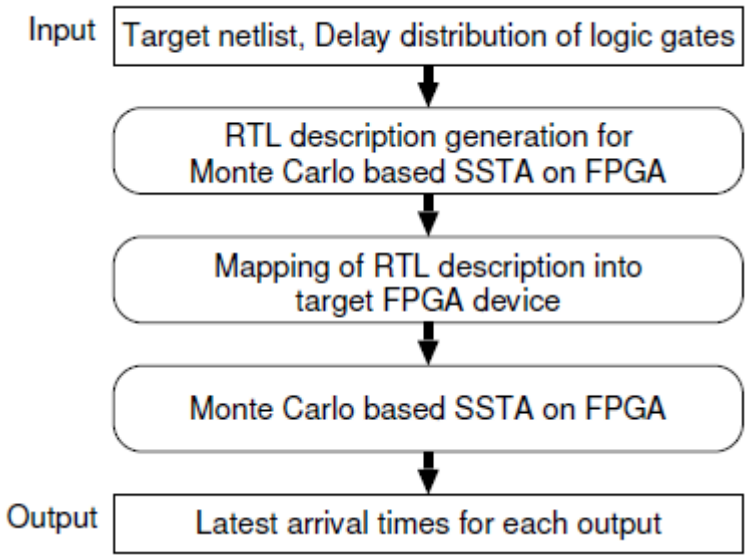


Fig 2.1 Proposed MC_SSTA Flow

## III Sampling Based MC SSTA: SH-QMC

We propose to implement a smart sampling based MC SSTA approach called SH-QMC (Stratification + Hybrid Quasi Monte Carlo) [11] on GPUs. This algorithm significantly speeds MC based SSTA using sample size reduction. In this technique, circuit timing criticality information is used for intelligent selection of samples. It is shown that 100-200 samples are sufficient for accurate statistical timing analysis. The process variation model is based on [2], which considers intra-die spatially correlated variation by partitioning the die into *n* * *n* grids and assuming identical parameter variations within a grid [11]. SH-QMC uses a combination of standard techniques and and circuit timing criticality information to reduce sample size for MC based analysis (variance reduction techniques). The variance reduction techniques employed are Quasi Monte Carlo (QMC), stratified sampling, and Latin Hypercube Sampling (LHS) [6]. These techniques are employed on variables based on their convergence properties and the ability to handle multiple variables. A detailed analysis of the algorithm is presented by the authors in [11].

## IV Monte Carlo Based Statistical Static Timing Analysis On GPUs

This section describes techniques for efficient implementation of MC SSTA on GPUs. In a random sampling based MC approach, samples of the chip are generated using process variation information. These samples have no data dependence on each other and therefore are directly amenable to parallelism. This is referred to as sample parallelism. Each thread is dedicated to the computation of one sample (representing one virtually fabricated die) of the circuit. Gates are visited in the topological order in the circuit by a thread and delay computations are performed. For the computation of process variation samples we use a Mersenne Twister based random number generator [14]. A detailed discussion is omitted for brevity.

## V. Fully Pipelined Implementation Of Monte Carlo Based SSTA

### *A. Overall flow*

Figure 1 shows the overall flow of the proposed MCSSTA. The inputs of the proposed flow include a target net list to be analyzed and delay description of each logic gate. It is currently assumed that the gate-delay distributions are represented by normal distributions. More specifically, two delay parameters, i.e. mean and standard deviation of each delay arc, are given as floating numbers. From the given input data, a synthesizable RTL description which realizes an MCSSTA corresponding to the target circuit is generated. The generated RTL description is mapped into a target FPGA, so that the MC-SSTA can be executed stand-alone. In the proposed implementation, STA operations are realized by a logic gate

basis to fully pursue parallelism. Each logic gate in the target circuit is replaced by a module called *delay-sample generator and LAT calculator* (DGLC), in which delay samples of delay arcs in the gate are generated and basic STA operations are executed. As shown in Fig. 2, a DGLC is connected cascade delay, i.e. the output of each DGLC is connected to the inputs of other DGLCs according to the wire connection in the original net list. We call the connection between DGLCs as *link*. Different circuits yield different constructions of DGLCs, thus FPGAs are our target implementation device. It is capable of changing hardware implementations according to the changes of the target circuit.
A DGLC includes normal distribution random number generators (NDRNG), adders and a comparator. Here, the comparator provides max (min) function according to the longest (shortest) path analysis. The NDRNG generates normally Distributed random numbers from uniform random numbers. A linear feedback shift register (LFSR) is used to generate uniform random numbers. Hereafter, uniform random number generators by the LFSR will be denoted as RNGs. In the following, the RTL description generation and implementation detail of the DGLC will be described. After that, we discuss the hardware cost of the proposed scheme and quality of the normally distributed random numbers.

*B. RTL description generation*

The net list of a target circuit is first translated into a corresponding synthesizable RTL description. For this purpose, we have developed a conversion program. The inputs of the program are a Verilog-RTL net list and delay parameters for each logic gate instances. First, a directed acyclic graph is constructed by parsing the input net list. Each node of the graph corresponds to logic gate in the target net list. For each input of the node, two delay parameters, which are mean and standard deviation, are associated. Arcs of the graph simply correspond to the connections between logic gates. Next, an instance of a DGLC is generated for each node. Each instance has specific delay parameters and initial values for uniform random number generators. We tested our approach on two circuits, the 'maccontrol' circuit from the Open cores Ethernet MAC design, and the 'fbfly' circuit which is a router design developed in the Concurrent VLSI Architecture Group at Stanford University [9]. Each of our circuits was synthesized using only three kinds of gates – NAND, NOR and INV from a 90 nm commercial standard cell library (see Section V for a description of less favorable performance seen in a different setting). Synthesis was carried out using the Synopsys Design Compiler tool. The sizes of the synthesized circuits were:

| Circuit | #Gates | # Links | Depth |
|---|---|---|---|
| Maccontrol | 1666 | 2507 | 18 |
| Fbfly | 165602 | 248586 | 49 |

On each circuit, the performance of our algorithm (which we call MCMC-PT) was compared to the performance of simple MC. The core deterministic STA engine used was essentially the same for both algorithms, enabling a fair comparison. No pruning of the circuits was done. The rationale for this is that any pruning should yield the same improvement factor for both simple Monte Carlo (MC) and MCMCPT, hence our comparative analysis will remain valid. The MCMC-PT algorithm parameters were tuned to enable fast computation of the yield at confidence points over 99.9%. For each circuit, simple MC was run for 17000 steps (41s for 'maccontrol' and 72 mins for 'fbfly' using one core of an Intel Core 2 Duo E6850 3.00 GHz processor). This leads to an estimation error of nearly 25% at 99.9% yield, for example. Then MCMC-PT was run for *exactly the same CPU time* for each circuit. As explained above, the same core procedures were used for each algorithm. The results obtained are shown in Figure 3.

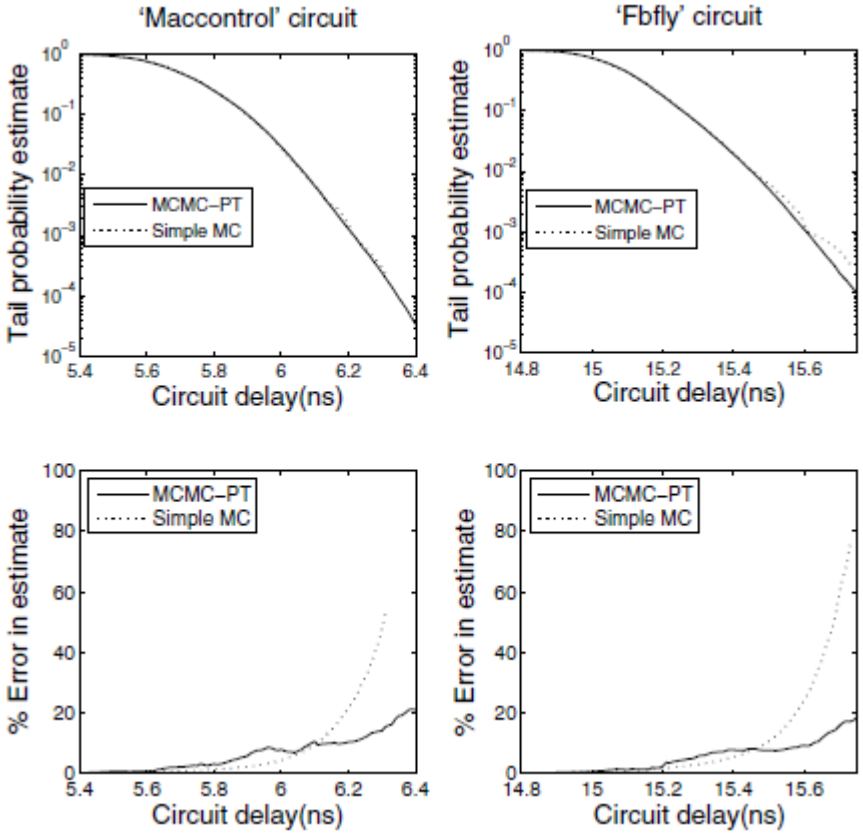


Fig 3.0 Mcmc-Pt Delays

 

## V. CONCLUSIONS

We proposed a fully pipelined implementation of MC-SSTA on FPGAs. According to the experimental result, 87 times Acceleration can be achieved compared to the software implementation in the case of a 6 bit multiplier. The analysis accuracy comparable to the Mersenne Twister and the Box Muller methods, which are the well-known high quality normal distribution random number generator, has been also experimentally verified. We are planning to extend the proposed scheme considering the following aspects. The size of the input net list, in terms of gate count, is strictly limited in our scheme, since the hardware resource required to implement our MC-SSTA is in proportion to the size of the input net list.

## REFERENCES

[1] D. Blaauw et. al., "Statistical timing analysis: from basic principles to state-of-the-art," *IEEE Trans. Computer-Aided Design*, vol. 27, pp. 589–607, Apr. 2008.
[2] L. Scheffer, "The count of Monte Carlo," *ACM/IEEE TAU*, 2004.
[3] V. Veetil et. al., "Criticality aware Latin Hypercube Sampling for efficient statistical timing analysis," *ACM/IEEE TAU*, personal communication.
[4] H. Niederreier, *Random number generation and quasi-Monte Carlo methods*. CBMS-NSF Regional Conference Series in Applied Math. no.63, SIAM, 1992.
[5] S. M. Ross, *Simulation*. 4th edition, Academic Press, 2006.
[6] M. D. McKay et. al., "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, pp. 239–245, 1979.
[7] J. F. Traub and A. G. Werschulz, *Complexity and information*. Cambridge University Press, 1998.
[8] S. R. Nassif, A. J. Strojwas, and S. W. Director, "A methodology for worst-case analysis of integrated circuits," *IEEE Trans. CAD*, vol. 5, no. 1, pp. 104–113, Jan. 1986.
[9] Y. Zhang, A. J. Strojwas, M. Sharma, and D. Newmark, "Statistical critical path analysis considering correlations," in *Proc. ICCAD*, Nov. 2005, pp. 699–704.
[10] X. Li, J. Le, M. Celik, and L. T. Pileggi, "Defining statistical sensitivity for timing optimization of logic circuits with large-scale process and environmental variations," in *Proc. ICCAD*, Nov. 2005, pp. 844–851.
[11] M. Pan, C. C.-N. Chu, and H. Zhou, "Timing yield estimation using statistical static timing analysis," in *Proc. ISCAS*, vol. 3, May 2005, pp. 2461–2464.
[12] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proc. ICCAD*, Nov. 2003, pp. 621–625.

 