A Case Study for Using Dynamic Partitioning Based Solution in Volume Diagnosis

Tao Wang[1], Zhangchun Shi[1], Junlin Huang[1], Huaxing Tang[2], Wu Yang[2], Junna Zhong[3]

[1]Hisilicon Tech. Co., Shenzhen, Guangdong, P. R. China

[2]Mentor, a Siemens Business, Wilsonville, Oregon, USA

[3]Mentor, a Siemens Business, Shanghai, P. R. China

***Abstract* —** Diagnosis driven yield analysis (DDYA) has been widely adopted for advanced technology node product yield ramp [1]. However gigantic design size and high pattern count demand intense computation resources to diagnose volume failure data, and the diagnosis throughput becomes the bottleneck for the DDYA flow. This paper presents a case study which uses the fully automated dynamic partitioning based diagnosis solution to dramatically improve the throughput. Experimental results based on real silicon manufactured by a 16nm FinFET technology show more than 3X reduction for memory footprint and more than 4X improvement for runtime, which eliminates the throughput bottleneck.

I. INTRODUCTION

As the semiconductor industry quickly moves into FinFET era [2-4] for higher density, lower power and better performance, "myriads of systematic defects" have been observed for these extremely sophisticated processes [3]. DDYA has been widely adopted to quickly identify such systematic yield limiters by data mining volume diagnosis results, and it was estimated that more than half a million failure files were processed daily by end of 2016 [1].

High diagnosis throughput is critical to the success of DDYA flow. The continuously increasing size of modern designs poses two primary challenges to achieving the throughput goal. The first one is that the time *t* for diagnosing a single failing die keeps increasing for a larger design because longer time is needed to simulate more gates. Therefore the number of failures (*T/t*) diagnosed on a single processor within the time slot *T* is reduced. The second one is the extremely high memory required for performing the diagnosis job. For example, hundreds of giga-bytes (GBs) of memory are required for diagnosing modern designs with hundreds of millions of gates, which makes lots of CPU resources with limited memory unusable. Even for workstations with a large amount of memory and tens of CPUs, the number of concurrently running diagnosis programs will be very limited as only a few diagnosis programs will use up all the memory, and most of the CPUs will simply stay idle.

Many works have been published on improving the performance for diagnosis using various techniques, such as pattern sampling [5], fault dictionary [6], machine learning [7] and GPU-based simulation [8]. However, the high memory requirement for big design diagnosis remains.

Because a defect in a circuit will typically affect a small portion of the entire design, a static design partitioning algorithm [9] was proposed to partition the design into several nearly equal-size partitions while trying to minimize the negative impact on diagnosis results. Since the diagnosis is conducted on the partitions, both the memory and runtime can be dramatically improved. However the diagnosis quality may be impacted if the defect affects multiple partitions. To address this issue, [10] proposed to dynamically determine a partition for a given failure file and then efficiently diagnose the failure on this partition. More than an order of magnitude throughput improvement was reported with a minimal negative impact on diagnosis quality. This idea was extended for chain diagnosis in [11] and similar diagnosis throughput improvement was achieved. Compared to traditional diagnosis, dynamic partitioning (DP) based diagnosis is much more complex and needs multiple steps for diagnosing each failure. A fully automated flow is necessary for production volume diagnosis.

Section II of this paper describes the automated dynamic partitioning based diagnosis flow using diagnosis server. The case study for this flow based on real silicon failures is presented in Section III. Section IV concludes this paper.

II. DYNAMIC PARTITIONING BASED DIAGNOSIS FLOW

As shown in [10][11], the DP based diagnosis is quite complicated. For a given failure file *f*, three steps are needed to accomplish the dynamic partitioning based diagnosis. The first step is to create a partition by a partitioner. Then an analyzer diagnoses *f* on this partition using much less memory and shorter CPU time, and generates a temporary report. Last, a partitioner translates this temporary report into the final diagnosis report. The flow becomes even more challenging when dealing with a huge number of failure files in production volume diagnosis. In order to solve all challenges of DP based diagnosis, diagnosis server is used to orchestrate all tasks and automate the DP based volume diagnosis flow.

Server process is the "brain" of this flow. It is responsible for ensuring three steps, including generating partition, diagnosing and translating report, are executed orderly to generate the final diagnosis report for each failure file. It also tries to keep all partitioners and analyzers busy to minimize the overall time for diagnosing all failure files. In order to achieve these goals, server process maintains three queues for failure files: partition queue, diagnosis queue and translation queue. All failure files are first lined up in partition queue after all diagnosis collaterals and failure files are set up. Once the partition is generated for a failure file *f*, *f* is moved from partition queue into diagnosis queue. Later, it is moved into translation queue after *f* is diagnosed by an analyzer. After the final report is generated, *f* is removed from translation queue. When a partitioner becomes ready for next task, server picks one failure file from translation queue if that queue is not empty, and asks that partitioner to translate its temporary report. If the translation queue is empty, server picks one

978-3-9819263-0-9/DATE18/©2018 EDAA

failure file from partition queue instead for this partitioner. When an analyzer becomes idle, server assigns it with the next failure file from diagnosis queue. Server process continues doing this till all three queues become empty.

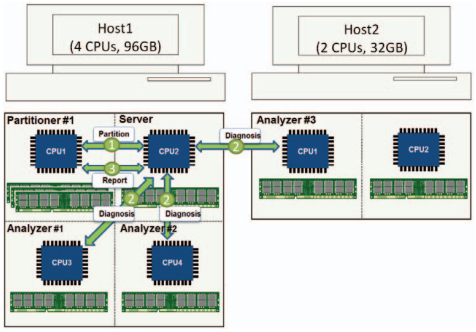

Figure 1. Dynamic partitioning based diagnosis server flow.

This automated flow is illustrated in Figure 1. In this example, let us assume that a batch of failure files for a design with 50 million gates needs to be diagnosed by using two workstations: *Host1* has 4 CPUs and 96GB RAM, and *Host2* has 2 CPUs and 32GB RAM. The server process is first started on *Host1* to orchestrate all tasks. Then one partitioner is added on *Host1*. Last three analyzers are added: two analyzers on the same machine *Host1* and the third analyzer a remote machine *Host2*. The three steps required for a failure file are marked by numbers. The only partitioner does all partition generation and reports translation jobs. Three analyzers cooperate on all diagnosis jobs. All processes work cooperatively to diagnose all failure files. Extra partitioners and analyzers can be added to further improve the diagnosis throughput. Another advantage for the DP flow is that machines with less memory like *Host2*, which is unusable for traditional diagnosis flow, can be used as analyzers due to reduced memory requirement.

III. Experimental Results

We picked one block of a large SoC design manufactured by 16nm FinFET process to evaluate the throughput improvement for this flow. This whole design has 240 million logic gates and this block has 53 million logic gates. Total 264 failure files were collected, including 99 chain failures and 165 logic failures. We used diagnosis server to automate diagnosing these failure files using our computer grid. Two runs were executed to compare the regular server flow against the dynamic partitioning based server flow. Both runs used 6 diagnosis processes plus the server process. For regular flow, all 6 processes performed diagnosis. For DP flow, 1 partitioner and 5 analyzers were used.

The diagnosis time results are reported in Table 1. The average diagnosis time in seconds for chain, logic and all failures is reported in column 2 to 4. The speedup is reported in the last row. On the average, diagnosis time is improved by 4.2X. DP flow achieves slightly higher speedup for chain failures diagnosis than that for logic failures diagnosis.

The memory footprint data is reported in Table 2 in GBs. Server process consumes little resource and is thus skipped. The memory footprint for each process is reported in column 2 to 7, and the last column shows the average memory footprint. In DP flow, the partitioner has a small memory overhead, but the analyzers use much less memory. Overall the DP flow reduces the average memory usage by 3.0X.

Another important metric for volume diagnosis is the turn-around time (TAT). For this experiment, the DP flow reduced the TAT from more than 4 days down to less than 1 day. In addition, extra partitioners and analyzers can be used to further reduce the TAT. For example, with 2 more partitioners and 10 more analyzers, DP flow may be able to reduce the TAT for this batch of failure files down below 8 hours. This can translate into a throughput improvement of ~12X.

Table 1. Average diagnosis time per failure file

	Chain (sec)	Logic (sec)	All (sec)
Regular	6696	6688	6691
DP	1508	1637	1589
Speedup	4.4	4.1	4.2

Table 2. Memory footprints for processes in GBs

	P1	P2	P3	P4	P5	P6	Avg.
Regular	66.2	63.8	68.2	56.5	63.7	63.6	63.67
DP	83.6	7.5	7.9	8.1	10	9.2	21.05

IV. Conclusion

Volume diagnosis driven yield analysis has become an organic component for production yield ramp for advanced technology nodes. However diagnosis throughput faces many challenges due to increasing design size and test pattern count. Dynamic partitioning based diagnosis solves this bottleneck by dramatically reducing both the CPU time and the memory footprint. A fully automated flow based on diagnosis server is suitable for production volume diagnosis. Experimental results on real silicon failures show that more than an order of magnitude of throughput improvement can be achieved.

References

[1] W. Rhine, Plenary Keynote, *ITC*, 2016

[2] https://staticwww.asml.com/doclib/investor/asml_3_Investor_Day-Many_ways_to_shrink_MvdBrink1.pdf

[3] S. Natarajan, et al., "A 14nm Logic Technology Featuring 2nd-Generation FinFET, Air-Gapped Interconnects, Self-Aligned Double Patterning and a 0.0588 µm^2 SRAM Cell Size", in *Proc. of IEEE Intl. Electron Devices Meeting*, Dec. 2014

[4] H.-J. Cho, et al., "Si FinFET based 10nm technology with multi Vt gate stack for low power and high performance applications", in *Proc. of IEEE Symp. on VLSI Technology*, Jun. 2016

[5] A. Leininger, et al., "Compression Mode Diagnosis Enables High Volume Monitoring Diagnosis Flow," In *Proc. of ITC*, pp. 7.3, 2005

[6] H. Tang, et al., "Improving Performance of Effect_Cause Diagnosis with Minimal Memory Overhead", in *Proc. of ATS*, pp. 281-287, 2007.

[7] S. Wang and W. Wei, "Machine Learning-based Volume Diagnosis," in *Proc. of DATE*, pp. 902, 2009

[8] H. Li, et al. "nGFSIM: A GPU-based Fault Simulator For 1-to-n Detection And Its Appications", in *Proc. of ITC*, pp. 1, 2010

[9] X. Fan, et al., "On Using Design Partitioning To Reduce Diagnosis Memory Footprint", in *Proc. of ATS*, pp. 219-225, 2011.

[10] X. Fan, et al., "Improved Volume Diagnosis Throughput Using Dynamic Design Partitioning", in *Proc. of ITC*, pp. 1-10, 2012

[11] Y. Huang, et al., "Distributed Dynamic Partitioning Based Diagnosis of Scan Chain", in *Proc. of VTS*, pp. 1-6, 2013