© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

Analysis of Microprocessor Validation Processes

KARAN LEHKRA ,VIKAS, ANIL

DRONACHARYA COLLEGE OF ENGINEERING

Abstract **To improve the efficiency of microprocessor validation processes it is necessary to quantitatively analyze these processes with respect to key metrics such as: design quality and process quality, and relative debug effectiveness and relative debug efficiency. In this paper, we present a simple but novel process modeling technique, that can be used for analyzing, comparing and assessing validation processes, as well as for developing practical design quality and schedule management techniques.**

We apply this method to two robust industrial case studies, with two unique strategic choices representing simulation-centric and isomorphism-centric methodologies, respectively. We show that these two unique processes can be effectively and efficiently modeled, and their performances characterized and quantitatively assessed. We also show that the proposed method provides process comparisons with enough engineering details, and at the same time it is simple and insightful – useful features for process management, as well as potential improvements.

I. INTRODUCTION

Microprocessor validation processes can be characterized as either simulation–centric or isomorphism–centric, according to which technology, simulation or formal verification, detects most design errors. Today's microprocessor validation processes are largely simulation-centric, using formal methods [1] to detect a few but very difficult corner-case design errors and large simulation compute-farms [2, 3] to detect most of the specification and design errors. However, its future success faces, in addition to Moore's Law, at least two formidable challenges: Amdahl's Law [4] and Pareto's Law (the final 20% of the design errors consume 80% of the schedule). Thus, because the impact of the latter occurs late in the pre–silicon validation cycle, the efficiency of simulation–centric validation processes is at best problematic.

An isomorphism-centric validation process reverses this strategy. Starting at the functional block level with two independent models [5], it uses isomorphism-checking (an indirect method of formal verification [6] that's analogous to Kurshan's Boolean homomorphism– checking [7]) to more efficiently detect 100% of the unique design errors earlier in the validation cycle as they occur. It then focuses the power of the simulation compute-farm on the relative few specification errors and remaining common design errors.

The main objective of this paper is to quantitatively and rigorously analyze, compare and assess these two unique strategies. To accomplish this, we developed a simple, but novel electronic design verification process (EDVP) model, called MPROV. We apply this unified quantitative analysis model to two case studies typifying each of the two unique validation processes, respectively, and demonstrate the effectiveness of MPROV.

II. THE MPROV MODEL

To construct MPROV, we follow Foster [8] and Rosenthal [9], and leverage useful software engineering and development process concepts. MPROV is composed of four major components. The first is the input data profile (IDP) which contains the results from a specific validation project, as illustrated in Tables 2 and 5. Process data are hard to obtain. When they are presented to the public sanctioned by respective companies, these data are often presented in different formats. Therefore, for the time being when there is no standard governing this, IDP often requires reverse engineering and decipher from the original published data. Many of the data contained in Tables 2 and 5 are therefore not necessarily categorized the same in the original referenced papers.

The second is metrics. According to the well-known management guru, Peter Drucker, "If you cannot measure it, you cannot manage it." The significant role played by metrics in software and hardware development is well known [10, 11]. Our metrics are shown in Table 1, with phase-based measurements as well as processbased measurements.

The third is decomposition which dissects a validation process into four distinct phases called design review (DR), design integration verification (DIV), implementation verification (IV) and hardware
integration test (HIT). Their relationship to the phases of electronic design process (EDP) [8] is quite clear: DR and DIV focus on the detection of specification errors in an original "paper" specification, and the design errors introduced by designers. IV focuses on the implementation errors injected post-synthesis during physical design, and HIT on the detection of errors escaping the three preceding phases. We use " bug" to mean any pre-silicon error and "defect" solely in reference to post-silicon errors.

Lastly, with the practical ancient wisdom that "To judge a thing one must first know the standard" [12] in mind, MPROV's fourth component is the Sigma learning curve, which is widely used in improving business processes [13], and was considered early on by IBM EDA for software quality improvement [14]. It aligns electronic system design validation process improvement with manufacturing process improvement [15], where the gold-standard of performance excellence is often called Six-Sigma (or 3.4 DPM). Sigma-based evaluation allows us to assess the effectiveness and efficiency of validation processes with different setups and managements.

III. SIMULATION-CENTRIC VALIDATION PROCESSES

In this section, using MPROV, we analyze the Pentium® 4 case study [2]. The resulting IDP and MPROV model are shown in Tables 2 and 3, respectively. From the last column in Table 3 we obtain a data point, corresponding to the HIT/System Design Quality value of 134 DPM, on the Sigma curve as shown in Figure 1. To place the result in perspective, we have added the values attained by NASA and HP-Y to the curve.

The Pentium® 4 MPROV model results in Table 3 make several conclusions readily apparent. First, note that a constant bug-density, a constant 48-month development schedule and Moore's Law, imply the bug-removal rates of successive microprocessor projects must quadruple! Second, our Pareto analysis of the bug-exposure results as shown in Table 4, indicates that 20% of the total bug exposure was discovered during the pre-validation design review phase; in other words, at the earliest possible opportunity. Clearly, the post-mortem rootcause analyzes [2, 18] " closed the loop" and made future reviews even more effective

IV. ISOMORPHISM-CENTRIC VALIDATION PROCESSES

IBM first used LSI technology to lower costs and increase reliability, as its 3033 processor clearly demonstrated. However, its use in the 3081 processor, with increased performance and functionality, entailed enormous challenges in establishing its functional correctness, which resulted in a paradigm shift from the conventional hardware-centric, to an isomorphismcentric validation process. The core capability was novel methods of formal verification, which were supported by DFV rules (though not explicitly labeled as such) [16], as well as design-for- testability (DFT) rules. These methods were referred to as isomorphism-checking [6] and formal equivalence-checking [16], respectively. In this section and using MPROV, we analyze this shift and quantify the benefits, which include dramatic improvements in design quality, and in bug-removal rates and validation schedules [6].

The IBM 3081 case study [6] was analyzed and the resulting MPROV IDP is shown in Table 5 by reverse engineering. The MPROV model, as shown in Table 6, makes a number of conclusions apparent. First, our Pareto analysis of the bug-exposure results in Table 7 shows that 15% of the total bug exposure was discovered during pre-validation design review, the earliest possible opportunity. Second, the value of

bug-density achieved matches the best of NASA's long-term effort. But this was due to a clever methodology improvisation that resulted from the 3081's use of the two-model paradigm discussed earlier. Third, the bug-exposure obtained during the DIV/Block phase shows that detection of 100% of the unique design bugs consumed about 50% of the design verification schedule. This explains the dramatic 66% schedule reduction [6] that resulted from an earlier, more effective and efficient bug detection. Fourth, the bug-exposure obtained during HIT phase reflects the 3081's critical dependence on hardwareprototyping greatly enhanced by employing isomorphism-checking, since it had removed most bugs earlier upstream. From the last column in Table 6 we obtain a data point, corresponding to the HIT/System Design Quality value of 313 DPM, on the Sigma curve as shown in Figure 1.

V. COMPARISONS AND ANALYSIS

To compare the Pentium® 4 and IBM 3081 validation processes, we use the MPROV model data in Tables 3 and 6. Comparing the Relative Debug Effectiveness during the DR phase in Tables 4 and 6, it shows that the Bug Exposure number with Pentium® 4 is 3x larger than

3081's, while Pentium® 4 DR size is 25% larger. This indicates that Pentium® 4 had a much effective and efficient DR capability to discover bugs at the earliest possible moment. Note that with Pentium® 4 most of the design bugs were identified through simulation, while with 3081 most of the design bugs were discovered by isomorphic-checking formal methods.

Comparing the Relative Debug Effectiveness during the
DIV/BLOCK phase in Tables 4 and 6, it shows that the 3081's isomorphism-centric validation captures 78% of the non-implementation bugs (96% of those escaped the DR phase), equally as effective as Pentium® 4 simulation-centric validation process, but is more efficient with less time (31% vs. 44% schedule impact).
Comparing total Debug Rates (281 vs. 165) of the DIV phase shown in Tables 3 and 6, it suggests that a simulation-centric strategy is more efficient. However, the fact that 8 Pentium® 4 microprocessors were simulated and debugged concurrently [2] implies that its effective Debug Rate is about 34. Thus, relative to a single microprocessor, an isomorphism-centric strategy is about 3 times more efficient.

Comparing the System Design Quality during the HIT phase (134 DPM of Pentium® 4 vs. 313 DPM of 3081) shows the dramatic impact that concurrent simulation had on discovering more bugs earlier. This is most vividly illustrated in Figure 1, with the two respective data points on the Sigma curve.

VI..CONCLUSION

In this paper we have shown that the proposed MPROV model provides a methodical approach to evaluate microprocessor validation processes which may use different methodologies. The MPROV metrics provides line-item measurements with enough details for analyzing process effectiveness and efficiency. We also demonstrated with two well-known case studies, IBM 3081 and Intel Pentium 4, the application of MPROV and the insights the methodology provides.

In analyzing the Pentium 4 validation process results, we found that it had one of the best design review methodologies and it made the use of compute-farm its center piece, relying on simulation to discover most of the design bugs. In analyzing the IBM 3081 validation process results, we found that its unique use of formal methods using isomorphic-checking was very effective and efficient in identifying design bugs early on in the process, when compute-farm technology was not invented yet.

Our analysis made it apparent that design quality and schedule impact can be further improved by combining formal methods as well as compute-farm simulation technologies.

REFERENCES

[1] R. Composano, "The Expanding Role of Formal Methods in Electronic Design" , Keynote at ISQED 2001.

[2] R. Bentley, " Validating the Intel® Pentium® 4 Microprocessor" , in the Proc. of the 38th Design Automation Conference, paper 16.1, June 2001.
[3] D. Lee, " Compute Ranch Takes over EDA", EEdesign of EE Times, August 1999. http://www.eedesign.com/editorial/1999/ranch9908. html.
[4] R. Bentley, "What's the Next 'Big Thing' in Simulation-Based Verification?" IEEE International High Level Design Verification and Test Workshop, 2003.
[5] E. Detjens, Editorial Comments in Computer Design, August 1996.
[6] M. Monachino, " Design Verification System for Large-Scale LSI Designs" , IBM J. of Res. and Dev., pages 89-99, Vol. 26, No. 1, January 1982.
[7] R. P. Kurshan, " Computer-Aided Verification of Coordinating Processes – The Automata-Theoretic Approach" , ©1994, Princeton University Press. ISBN 0-691-03436-2.
[8] H. D. Foster, A. C. Krolnik and David J. Lacey, "Assertion-Based Design" , ©2003, Kluwer Academic Publishers. ISBN 1-4020-7498-0.
[9] C. W. Rosenthal, "A Structured Method for Assessing the Maturity of an Electronic Design Process" , in the Proc. of the International Conference on Management of Engineering and Technology, October 27-31, 1991, pages 242-246.
[10] K. H. Mö ller and D. J. Paulish, " Software Metrics – A practitioner's Guide to Improved Product Development" , ©1993, IEEE Computer Society Press. ISBN 0-412-45900-0.
[11] E. J. Aas, " Design Quality and Design Efficiency: Definitions, Metrics and Relevant Design Experiments" , paper 4B.1 at ISQED 2003.
[12] Anonymous, an ancient Sanskrit saying.
[13] J. Welch and J. A. Byrne, " Straight from the Gut" , ©2001, Warner Books, Inc. ISBN 0-446-52838-2.
[14] Electronic News, March 18, 1991.
[15] N. R. Mann, "The Key to Excellence – The Story of the Deming Philosophy," 3 rd Edition, ©1989, Preswick Books, Los Angeles.
[16] G. L. Smith, R. J. Bahnsen and H. Halliwell, " Boolean Comparison of Hardware and Flowcharts" , IBM Journal of Research and Development, Vol. 26, No. 1, pages 106-116, January 1982.
[17] DAC-2004 Panel " Verification: What Works and What Doesn't" , the 41 st Design Automation Conference, ACM/IEEE. June, 2004. Also the Summary in EE Times, June 25, 2004.
[18] B. Colwell, " Design Reviews" , Computer, pages 813, V. 36, No. 10, October 2003, IEEE.

Figures

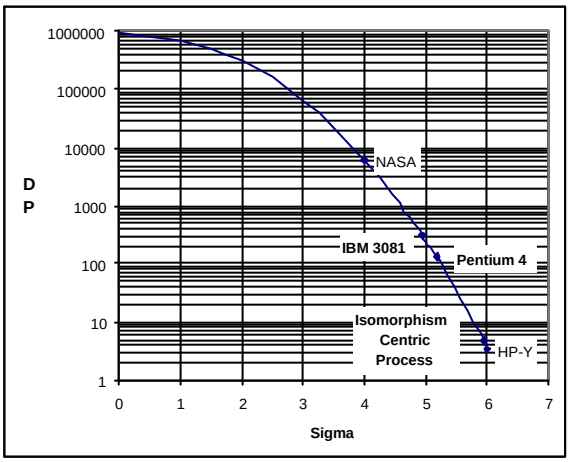

Figure1. Comparison of Processes

Tables

Table 1. Definition of Metrics Used by MPROV

METRIC NAME	DEFINITION
Validation Throughput	# correct gates per month
Debug Rate	# bugs removed per month
Bug Density	# bugs per KG (or KLOC)
Relative Debug Effectiveness / Efficiency	#bugs removed time of # bugs removed / #total bugs - # impl. bugs debug time - IV time
Design Quality / Process Quality	# defects per MG (or MLOC) / Sigma

Table 2. Pentium® 4 Validation Process IDP

	Pentium® 4 [1] (Bugs / Months)		
Total Functional Bugs* / Validation Schedule	7989 (7855+134) / 48		
Design Level	Functional Block	Subsystem (CTE)	System (Full-Chip)
Design Size	?	?	> 1 mil. SRTL LOC
Design Review			1554 / unknown
Design Verification	100 / ? (Model Checking)	3411 / 9 (Simulation)	2398 / 12 (Simulation)
Implementation Verification	392 / 8 (Equivalence Checking)		
Hardware Test	?	?	134 [17] / 9

Table 3. Pentium® 4 MPROV Model

			MPROV MODEL METRICS							
			Debug Cycle-Time (Months)	Bug Exposure (Bugs)	Design Size (KLOC)	Validation Through-put (KLOC / Month)	Debug Rate (Bugs/ Month)	Bug Density (Bugs / KLOC)	Relative Debug Effectiveness / Efficiency (%)	Design/Pro Quality (DPMLO / Sigma)
MPROV MODEL PHASES	DR			1554	1000			1.6	20 / ?	
	DIV	Block		100*						
		Subsystem	9	3411			379	3.4		
		System	12	2398	1000		200	2.4		
		Total	21	5909	1000		281	5.9	78 / 70	
	IV		18	392	1000		21.8	0.4		
	HIT	Block								
		Subsystem								
		System	9	134	1000		14.9		2 / 30	134 / 5.1
		Total	9	134	1000		14.9			
TOTAL			48	7989	1000	21	166	7.9	100 / 100	