Database Analysis and Data Mining Methods To Improve the Quality and Efficiency of Coverage-Driven Functional Verification

Amir Hekmatpour
IBM Microelectronics
Research Triangle Park
North Carolina, USA
Rima@us.ibm.com

Azadeh Salehi
IBM Microelectronics
Research Triangle Park
North Carolina, USA
asalehi@us.ibm.com

Abstract

Functional verification is the bottleneck of the microprocessor design process. There is no practical method to test and verify all functional states of a complex microprocessor. In this paper, we introduce an integrated analysis environment utilizing database mining and data analysis techniques to improve the overall verification quality. The resulting system turns raw verification and coverage data into verification intelligence that are converted to test generation directives, efficient simulation biases, and improved coverage analysis models. A web-based interface provides verification progress reports and database management utilities. A prototype of the system has been utilized in the verification of an advanced PowerPC 4XX embedded processor.

1. Introduction

Functional verification is a major component of and the bottleneck of the microprocessor design process. It is estimated that over 70% of design resources are consumed by the verification [1][2]. However, the majority of the design re-spins are due to functional bugs [3][4]. Moreover, there is no practical method to fully test and verify the immense test space of a complex microprocessor [5]. Simulation is the predominant technique for functional verification. Its primary advantages are scalability and speed. Its primary disadvantage is limited coverage of design behavior - it is simply infeasible to exhaustively test every possible combination of inputs and states. To address the above limitations, a variety of techniques have been developed.

Automatic test generators running on a large network of computers can generate hundreds of thousands of tests each day resulting in gigabytes of test data that must be filtered and analyzed. In addition, coverage models and analysis filters applied to the above tests and simulation sessions generate additional gigabytes of functional coverage data each day. In many cases, the generated simulation and coverage data cannot be fully analyzed as fast as they are generated. Consequently, the verification bottleneck has now spread to test and coverage analysis as well. Not only is the verification an incomplete task, but the coverage analysis is also incomplete and requires additional resources of its own.

New techniques have been developed in response to the challenges facing the dynamic verification (simulation-based) of microprocessors. The new techniques include formal static verification methods, semi-formal methods, and assertion-based methods. Formal and semi-formal methods face design size limitations, require specialized skills, and can't guarantee completeness of the rules and models [6][7]. The assertion-based methods can be applied to both dynamic and static verification.

The coverage-driven verification methods described in this paper support assertion-based verification and utilize database analysis and data mining (as shown in Figure 1) to address some of the shortcomings of dynamic verification outlined above.

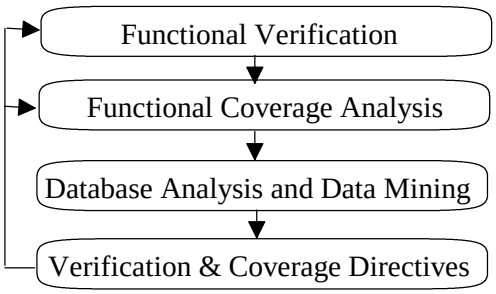

Figure 1: System Functional Diagram

In the remainder of this paper we first give an overview of the evolution of the dynamic verification from manual test generation, to automatic test generation, to fully random test generation, and finally to the coverage-driven directed random verification. We then introduce the database analysis and data mining methods and the ways in which these

techniques are integrated into a highly automated and advanced verification environment to improve the overall verification quality and efficiency.

2. Processor Functional Verification

Microprocessor verification is an important part and the bottleneck of the microprocessor design process. There is no practical method to test and verify all functional states of a complex microprocessor. A variety of techniques are used to generate test programs (i.e. tests) to verify the design's intended functionality. These include deterministic tests manually generated by engineers, tests generated by automatic test generators, standard test benches or a combination of all techniques. In all cases, test programs are simulated against the design model to identify faulty functions. Figure 2 shows a typical dynamic (simulation-based) functional verification environment utilizing manual and automatic test generation.

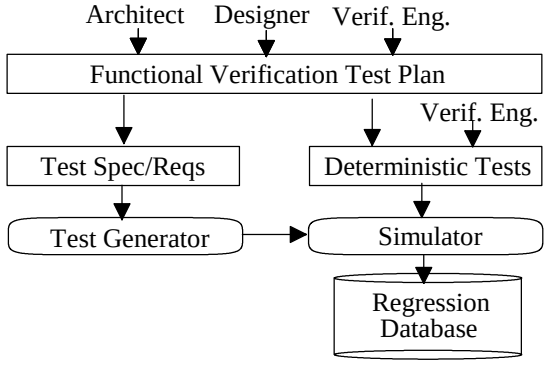

Fig. 2. A typical Processor Verification Environment

As the complexity of integrated circuit designs increase, so do the number and complexity of test programs required to verify them. This in turn requires more resources to simulate and debug tests. Various techniques and systems have been developed for automatic generation of directed random and highly specialized targeted random tests [8]. Figure 3 shows an example of an advanced functional verification environment where in addition to manual tests and standard test benches, directed random and targeted random test generators are also utilized. A large number of tests are generated and simulated targeting all design features, corner case functional scenarios, and CUSP floating point operations [4]. To take full advantage of the test generator capabilities and to better manage the verification resources, management and optimization techniques are devised [4].

A highly automated and distributed test generation and verification environment can be developed based on the methodology shown in Figure 3. Such a verification environment generates a large number of tests and a large quantity of simulation data and test fails which need to be continuously reviewed and processed (24/7). In addition, such automatic random test generators running on a large network of computers can generate hundreds of thousands of tests each day resulting in several gigabytes of test data that must be filtered and analyzed every day.

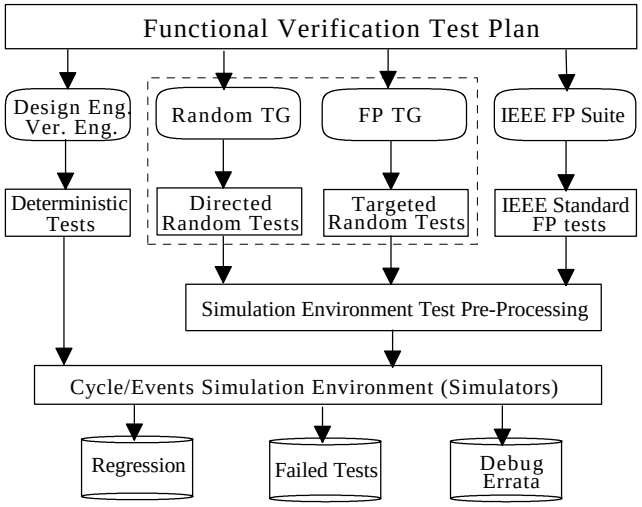

Figure 3. Automatic Random Functional Verification

3. Coverage Analysis

One major issue that must be addressed is whether all functions of the design under test have been tested and verified; if not, which functions or design features have not been fully verified. This information should be available in a timely manner so resources can be directed to verify functions that have been missed.

It is also very important to know promptly which functions have already been verified so that duplications are reduced and resources are targeted towards untested functions. Verification resources could include verification engineers, test generators, simulation computers' capacity (i.e. simulation cycles) and debugging resources, etc. In order to decide whether the design has been fully verified, the following questions need to be answered:

1. Has everything detailed in the test plan been verified; if not, what are the remaining functions to be verified (functional coverage holes)?
2. Have all the new scenarios/states not included in the test plan been verified?
3. Where should the attention/resources be focused to ensure the best coverage of verification?

To answer these questions, functional coverage tools and methodologies have been developed that monitor the simulation processes and analyze the test programs, thereby generating a vast quantity of data on what has been tested so far (Figure 4). Also, some of the coverage analysis tools/methodologies allow the verification procedures to define models of classes of scenarios that should be verified and enumerate a number of design states to be verified. These in turn result in "coverage hole" reports which list scenarios that have not yet been simulated or verified. Verification resources can then focus on the remaining unverified design features (i.e. coverage holes).

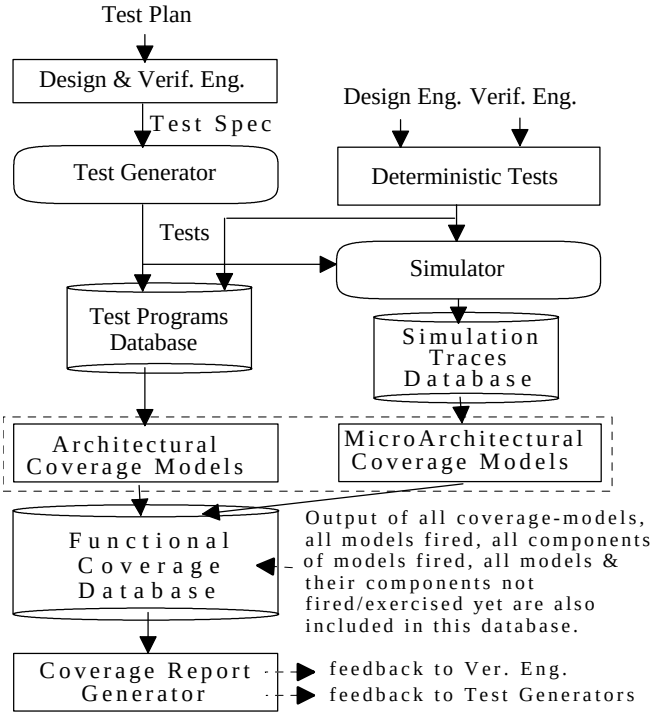

Figure 4. A Coverage-Driven Functional Verification

When coverage analysis models and filters are applied to tests generated and simulated in a 24/7 verification environment, several gigabytes of functional coverage data could be generated each day. In many cases, the generated simulation data and its corresponding coverage data cannot be fully analyzed as fast as they are generated, resulting in the following deficiencies:

1) A failure to extract all of the important coverage data from simulation database, thereby having an incomplete view of what has been simulated
2) A failure to properly interpret the accumulated data to provide timely feedback to verification process based on accurate coverage data
3) Shallow analysis of simulated tests because coverage models/filters are not extensive enough and they do not deal with the tiered and temporal functional features and design constraints
4) Many interesting functional scenarios and patterns go undetected because they are not reflected in the coverage models and are hence not detected
5) No data is generated on correlation between models, since models are developed independently and coverage analysis tools and methods work with the output of models and no cross-model analysis capability is usually provided
6) No new discovery is made based on the actual data because the generated tests or coverage data are not analyzed as a whole from a temporal point of view or from an overall design progress point of view but rather in isolation one test at a time and one model at a time, and
7) Not all design changes are reflected in coverage models, since models are defined based on the test plans and design specs rather than the actual design itself.

Furthermore, focused coverage analysis requires dynamic adjustment of the analysis process based on a deep knowledge of test features and coverage analysis data. It requires taking into consideration the trends and relevant patterns of the actual verification progress, as well as any known and expected patterns missing in the collected data. Today's standard coverage analysis, as well as advanced coverage-driven verification such as the one shown in Figure 4, generate reams of data and many coverage reports which may not fully reflect the actual attributes of the verification process. Expertise for development of coverage analysis models and effective utilization of coverage tools are scarce. Designers and verification engineers are the best candidates to write effective coverage models and analyze coverage data. They have a good knowledge of the design's internal workings and details, but are usually too busy with their own traditional design and verification tasks. Therefore, new and dynamic analysis techniques are required.

4. Verification Intelligence

Coverage analysis tools generate a variety of reports presenting predefined coverage information, which usually show accumulated statistics rather than shifts in the trends or new behavior. Design and verification engineers are already overwhelmed with data. They need useful hints and specific directives which would provide insight into the actual progress of the test

generation and simulation process. They need intelligent analysis and assessment of the data collection process and the accumulated data. They need verification intelligence (VI) rather than more data and more reports.

Intelligent and methodical analysis based on deep knowledge of the verification process is required to turn continuous flow of test and coverage analysis data to readily usable and relevant information. New methods beyond simulation-based verification, assertion-based techniques, and coverage analysis are needed to convert gigabytes of verification data that are collected each day to usable knowledge. Furthermore, hidden verification patterns, attributes, trends, associations, and relationships in the collected verification and coverage analysis data that may shed light on the actual behavior and progress of the verification process should be extracted and analyzed.

The functional verification environments we have described in the previous sections and the class of problems they face are similar in many ways to those in the fields of finance and science - where database mining techniques have proved effective in analyzing large volumes of data on a continuous basis [9]. Figure 5 shows how database mining and data analysis can be utilized to provide corrective feedback and automatic calibration of the utilized tools and resources.

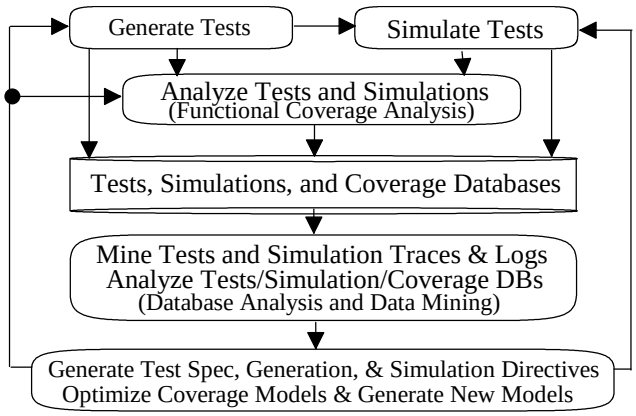

Figure 5. System Functional Diagram

Figure 5 shows the functional flow diagram of the proposed system. The system performs comprehensive analysis of the verification environment in order to gain insight into the simulation, coverage analysis, and the data collection processes. It then generates feedback for dynamic adjustment of the environment variables and its utilities based on the discovered missing test features, redundant simulation patterns, coverage model deficiencies, etc.

5. Database Analysis and Data Mining

Data Mining is an analytic process designed to explore large amounts of data in search of consistent patterns and/or systematic relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data [9]. The ultimate goal of data mining is prediction - and predictive data mining is the most common type of data mining and has the most direct business applications [10]. The process of data mining consists of three stages: (1) the initial exploration, (2) model building or pattern identification with validation/verification, and (3) deployment (i.e., the application of the model to new data in order to generate predictions).

To identify any possible relationships between tests and any corresponding simulation patterns, the test regression database and its corresponding simulation database are analyzed to identify the following:

- Over/under-simulated design features
- Over/under-generated design features present in the test database
- Patterns, relations and design features present in the database but not in design/test spec
- Relations between class of generated tests and deterministic tests
- Sequential and temporal trends and relations

Similarly, coverage database and coverage models are also analyzed to identify the following:

- Underutilized (i.e. under-simulated) patterns, relationships and design features
- Patterns, relationships and design features that are present in the database but not modeled
- Meta relations, relationships and trends common between models
- Over-utilized (highly and frequently simulated) patterns, relations, design features and models
- Temporal trends, seemingly independent patterns that emerge after a period of time

6. System Architecture

Figure 6 shows the architecture, associated processes, and functional features of the proposed system for functional verification of advanced and complex integrated circuits such as microprocessors, Applications Specific Integrated Circuits (ASIC), and System-on-a-Chip (SoC). The system is a simulation-based verification environment which incorporates database analysis and data mining techniques to generate directives for fine tuning and

optimization of test specifications, test generation, simulation, and coverage analysis. The feedback analysis and fine tuning enables the verification process to adapt to the actual characteristics of the tools, models, and the design attributes. The following sections give an overview of the system architecture and functional features as shown in Figure 6.

6.1 Database Mining Methods

The following sections describe how database analysis and data mining techniques are applied to verification processes and databases. The goal is to identify and extract new relations, rare associations, dependent patterns, and unexpected sequences in order to identify the following verification attributes:

- Underutilized and under-simulated patterns, relationships and design features from the analysis of simulation database and traces
- Patterns, relationships and design features present in the test and simulation database but not reflected in the coverage models
- Meta relations, relationships and trends common between coverage models or their reports
- Overutilized (highly and frequently simulated) patterns, relations, design features and models
- Temporal trends, seemingly independent patterns that emerge periodically and how they correlate to tests or simulation events
- Relations and operations found in the simulation database but not in the design spec or test plan
- Relations between automatically generated tests and manually generated tests that may reveal redundancy, and
- Temporal trends and relations that emerge in sequence, perhaps reflecting repetitious biases in simulation or verification environment.

6.2 Test Generation Optimization

The proposed system includes all the traditional sources for test specification such as a test plan and inputs from the architect, designers and verification engineers. In addition, feature extraction data mining techniques are applied to the design HDL to classify rare and unique HDL constructs. The identified constructs are then mapped to microarchitectural and architectural test specifications. All specifications are

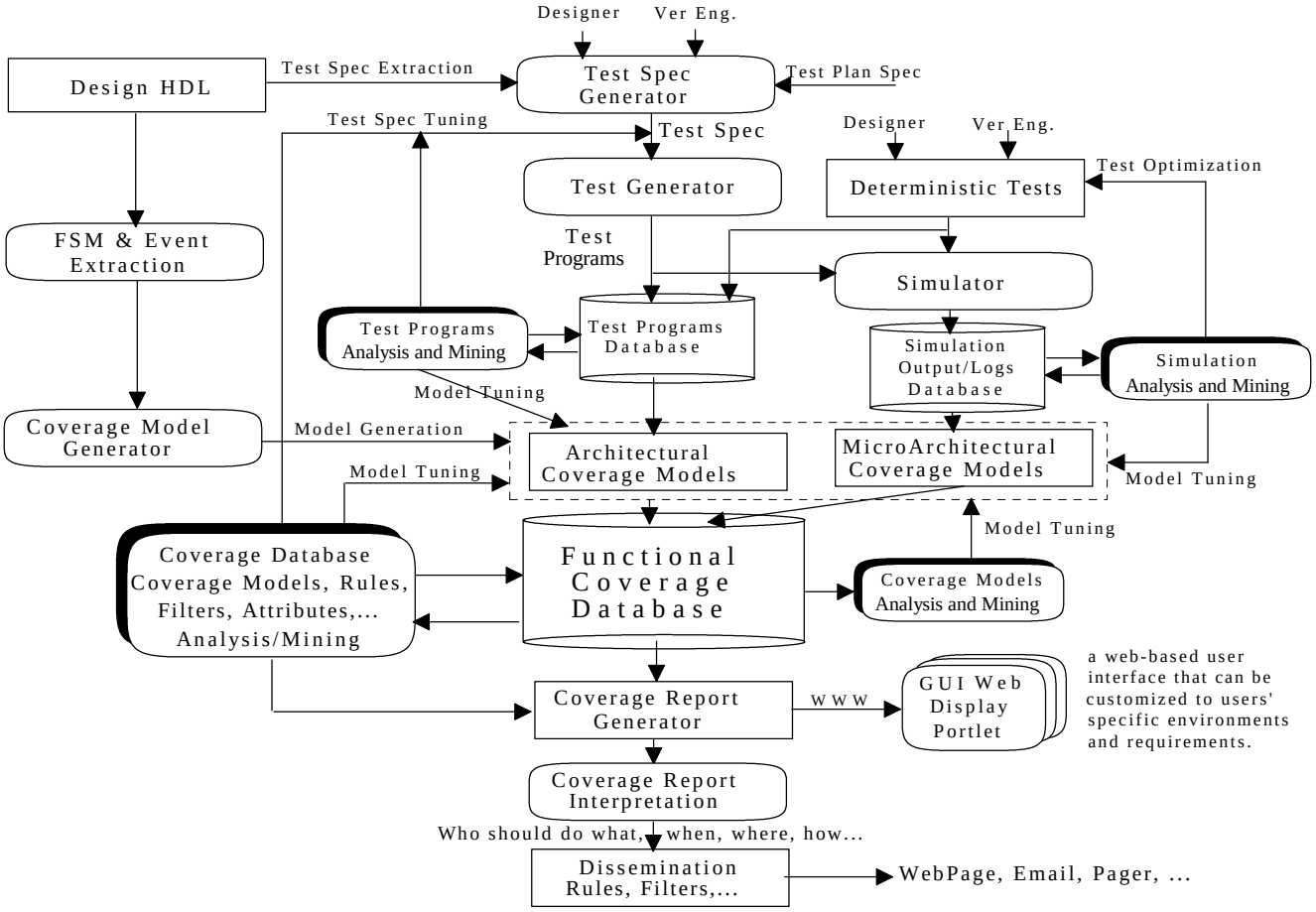

Figure 6. System Architecture and Functional Organization of the Coverage-Driven Verification System

then customized for the deployed generator(s). The test spec generation process is further optimized by the feedback provided from the analysis of the test and coverage databases. These directives compensate for the test generator biases, coverage model deficiencies, coverage analysis shortcomings, and missing tests (coverage holes). Redundant patterns and saturation trends discovered in the test and coverage databases are converted to directives for test spec optimization as well as test generator fine tuning and customization.

6.3 Coverage Analysis

Functional coverage consists of architectural and microarchitectural analysis of the verification process (tests and simulation traces, logs and outputs) based on the models developed manually by the engineers. In our proposed system additional coverage models are generated automatically by extracting Finite State Machines (FSM) from the design HDL source. The extracted FSM and functional events are mapped to mathematical (Boolean) representations which are then converted to microarchitectural coverage models. In addition, any discovered operational sequences or data pattern sequences in coverage database that evolve over time are also converted to coverage models.

6.4 Simulation Analysis

Data mining classification and clustering techniques are applied to the simulation database to identify redundant tests and tests with similar operation sequences. This is very helpful when utilizing random test generators which may produce similar tests from different test specifications. The actual patterns and temporal sequences discovered in the simulation database are used to optimize the test generation process and the corresponding coverage models. In addition, simulation and coverage databases are analyzed to identify unique and underutilized complex operation sequences. New directives to test generators request more tests in those categories. Neural networks methods can be applied to the simulation patterns that have contributed to design bug discovery. Tracing backward through their corresponding simulation trace trees may identify the dominant attributes of tests that found the design bugs. This information is the type of feedback the test specification generator and test generators need to optimize their operations.

6.5 Directives and Progress Reports

Additional data mining techniques and knowledge discovery tools [10] can be applied to the functional coverage database and the coverage models database. Discovered trends and associations are then used to generate interesting coverage report schemas, modify existing coverage reports, or generate action directives. Coverage reports and directives are available from the system's web-based interface or can automatically be disseminated via wired and wireless communication media.

7. Summary

We described the evolution of functional verification technologies and how the verification bottleneck has shifted from test coding, to test generation, to simulation, to coverage analysis. To better understand and manage the actual progress of the verification, large volumes of test, simulation, and coverage data have to be analyzed. We then described how database analysis and data mining techniques are utilized to enhance coverage analysis, as well as optimizing test generation and simulation process.

References

[1] Denali, "Comprehensive Verification Suite," www.denali.com/news_pr20040510.html, 2004.

[2] 0-In Inc., "Verification reuse," www.0-in.com/resources-tech-articles.html.

[3] H. Foster, "FV of Block-Level Assertions," Proc. Of DesignCon East, April 2003, Boston.

[4] A. Hekmatpour, J. Coulter, Coverage-Directed Random Test Generation Management and Optimization, " Proc. Of International Test Conference, Sept. 2003, Charlotte, NC.

[5] Todd Austin, et. Al, "Building Buggy Chips - That Work!," The DIVA Project, University of Michigan, www.cs.utexas.edu/users/cart/slides/austin/pdf

[6] R. Jones, et. Al, "Practical formal verification in microprocessor design," IEEE Design and Test, 18(4):18-25, July/August 2001.

[7] R. Jones, "Application of formal mathematics to the specification, design, and verification of high-performance system," sprout.stanford.edu/rjones/research.html.

[8] L. Fournier, Y. Arbetman, M. Levinger, "Functional Verification Methodology for Microprocessors Using the Genesys Test-Program Generator," Proc. Of DTE99, Munich, Germany 1999.

[9] StatSoft, Electronic Textbooks, "Data Mining Techniques," www.statsoftinc.com/textbook/stdatmin.html

[10] P. Wright, "Knowledge Discovery In Databases: Tools and Techniques," www.acm.org/crossroads/xrds5-2/kdd.html

Acknowledgments

Authors are grateful to Dave Roberts and James Coulter for their review and comments on the early drafts of this paper.