International Journal of Electrical and Electronic Engineering & Telecommunications

ISSN 2319 – 2518 www.ijeetc.com
Vol. 6, No. 2, April 2017

Research Paper

A SURVEY ON VVM BASED TEST PROGRAMS FOR HYBRID RISC CONTROLLER

M Kamaraju[1] and Y Divyasree[2]*

Corresponding Author:* ***Y Divyasree,* ✉ *yamarthidivyasree@gmail.com*

Hybrid RISC Controllers are used in present day embedded systems in order to increase the flexibility and performance. These processors requires efficient test patterns to detect faults.In present days there are number of techniques are developed to detect the permanent faults in different processors Software-Based Self-Test (SBST) methods are used for automatic generation of test programs (SBST) programs. But by using these techniques test duration is high and even if every line of code has been executed the Device Under Test (DUT) may not be correct. VHDL Verification Methodology (VVM) provides a way of collecting the values of nodes in the verification environment and helps to decide when verification is completed. The the code size and test duration are measured.

Keywords: Hybrid RISC Controllers, VHDL Verification Methodology (VVM), Device Under Test (DUT), Software-Based Self-Test (SBST)

INTRODUCTION

Mainly due to the continuous scaling in the manufacturing process of the integrated circuits the testability problems are occurred in processor chips. Even during the operational phase permanent faults may occur due to the metal migration phenomena or aging of the circuit.

Most of the System-on-Chip (SoC) designs consists of one or more embedded processor cores .It also consists of other cores which are used to store and process the data and code for execution i.e., embedded RAM or ROM cores and to communicate with other peripherals etc (Ulbricht *et al.*, 2011). Testing of the processor cores become very difficult process. In addition to this, if the gap between the operating frequencies of Automatic Test Equipment (ATE) and operating frequencies of SoC increases, the failures which appear only testing is performed at the actual speed of the IC (at-speed testing) will not detect.So

[1] Prof & HoD, Department of E.C.E, Gudlavalleru Engineering College, Gudlavalleru , India.

[2] PG Student, Department of E.C.E ,Gudlavalleru Engineering College, Gudlavalleru , India.

that it is necessary to develop test programs to provide high fault coverage and also to reduce the test duration.

RELATED WORK

Many of the methodologies has been implemented to improve the fault coverage for processor cores and processor based System-on-Chip (SoCs) like Built-In-Self-Test (BIST) and Software-Based-Self-Test (SBST) (Sabena *et al.*, 2012).

The methodologies which requires external hardware to perform test are infeasible this is because of increasing gap between Automatic Test Equipment (ATE) and SoC operating frequencies. It will make external at-speed testing problematic and expensive.

A. Built-In-Self-Test (BIST)

Modern day ICs may develop faults even after manufacturing test. So additional testing is required for assuring quality of service. Built-In-Self-Test (BIST) is such a test procedure which facilitates testing of circuits before every time they start their operations. BIST is basically same as off-line testing using ATE where the test pattern generator and the test response analyzer are on-chip circuitry (instead of equipments).

BIST moves the testing task from external resources (ATE) to internal hardware: additional hardware and software are integrated into the circuit to allow it to perform self-testing. The use of this technique leads to lower cost of test and shorter tests time, maintaining or improving the fault coverage, at the cost of additional silicon area. But, such hardware-based self-test mechanisms that seriously impact performance, design time, cost and power consumption due to random patterns causing high switching activity, can be considered of limited practical value.Once BIST finds a fault, the readjustment can done by replacing the faulty part with a fault free one.

B. Software-Based-Self-Test (SBST)

The basic idea of Software-Based-Self-Test is to generate test programs to be executed by the processor and able to fully exercise the processor itself or other components in the system, and to detect possible faults by looking at the produced results. One of the main advantages of SBST lies in the fact that it does not require any extra hardware; therefore, the test cost is reduced and any performance or area penalty is avoided. Moreover, the SBST approach allows at-speed testing, and can be easily used even for on-line testing. For these reasons, SBST is increasingly applied for processors and SoC testing, often in combination with other approaches (Sabena *et al*., 2014).

The previous approaches apply functional self-testing to processor cores and rely on the use of pseudorandom instruction sequences and operations/operands. The functional-based strategies can be divided into two subclasses. The first corresponds to methods that rely mainly on code randomizers (possibly oriented with suitable constraints) to obtain test programs. The second consists of methods that adopt a feedback based strategy, meaning they evaluate generated test programs according to suitable metrics (often computed through simulation) and try to progressively improve them (Psarakis *et al*., 2010). Due to the high level of abstraction of the approaches and their pseudorandom

nature , structural fault coverage is usually low, although test programs with excessively large execution time are used.

The other approach for processor cores is structural testing methodology *structural* testing methodology for processor cores .In this approach at the first stage, the *test preparation stage*, pseudorandom pattern sequences are developed for each processor component in an iterative method taking into consideration the constraints imposed by its instruction set. Subsequently, test sequences are encapsulated into self-test signatures that characterize each component and consist of the seed and the configuration of the pseudorandom TPG, along with the number of test patterns. At the second stage, the *test application stage*, the component self-test signatures are first expanded on-chip by a software emulated LFSR (test generation program) into pseudorandom test patterns, then stored in embedded memory and finally applied to the component by software test application programs (Koal *et al*., 2005).

The structural methods are subdivided into two major subcategories. Methods in the first group are called hierarchical because they adopt a hierarchical approach. Such methods focus on a processor's modules one at a time, generating stimuli for each module and then extending those stimuli to the processor level. The second group, called RTL, includes methods in which the test program generation process exploits structural RTL information along with ISA information to generate instruction sequence templates for justifying and propagating faults of the module under test. These templates are then adjusted according to the module's testability requirements.

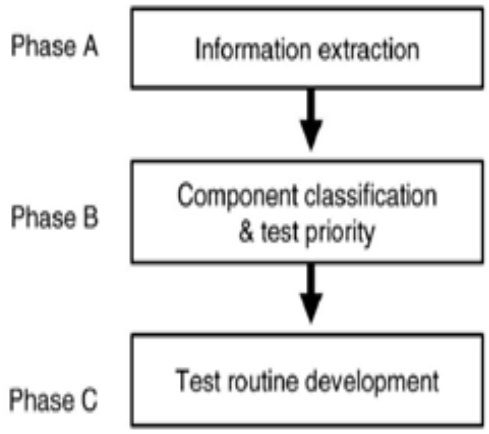
The SBST methodology for complex embedded processors previously proposed is consists of the three phases shown in Figure 1

Phase A: Identification of processor components and component operations, as well as instructions that excite component operations and instructions (or instruction sequences) for controlling or observing processor registers.

Phase B: Categorization of processor components in classes with the same properties and component prioritization for test development.

Phase C: Development of self-test routines emphasizing using compact loops of instructions, based on reusing a library of test algorithms for generic functional components in pseudocode tailored to the processor under test ISA that generate small precomputed test sets. These algorithms provide very high fault coverage for most types and architectures of the processor components independently of word length (Sabena *et al*., 2012).

Figure 1: The Phases in SBST Methodology



In this paper we proposed the development of test programs for Hybrid RISC controllers.

PROPOSED METHODOLOGY

The Hybrid RISC Controllers are used in embedded systems to improve the performance and flexibility. RISC (reduced instruction set computer) is a microprocessor that is designed to perform a smaller number of types of computer instructions so that it can operate at a higher speed (perform more millions of instructions per second, or MIPS). Since each instruction type that a computer must perform requires additional transistors and circuitry, a larger list or set of computer instructions tends to make the microprocessor more complicated and slower in operation. RISC processor, computer arithmetic-logic unit that uses a minimal instruction set, emphasizing the instructions used most often and optimizing them for the fastest possible execution. Software for RISC processors must handle more operations than traditional CISC processors, but RISC processors have advantages in applications that benefit from faster instruction execution, such as engineering and graphics workstations and parallel-processing systems.

Besides performance improvement, some advantages of RISC and related design improvements are:

- A new microprocessor can be developed and tested more quickly if one of its aims is to be less complicated.
- Operating system and application programmers who use the microprocessor's instructions will find it easier to develop code with a smaller instruction set.
- The simplicity of RISC allows more freedom to choose how to use the space on a microprocessor.
- Higher-level language compilers produce more efficient code than formerly because they have always tended to use the smaller set of instructions to be found in a RISC computer.

Standard VHDL has all the features necessary to code randomization of stimulus and functional coverage – both very important while verifying larger, system-level designs. The problem is that those features are quite advanced and require high coding skills. So that VHDL Verification Methodology is used for generating test programs for Hybrid RISC Controllers. It creates a couple of easily accessible VHDL packages that hide quite arcane implementation details from the average user, making generation of random stimulus and intelligent functional coverage.

CONCLUSION

In this paper different existing methods are explained along with their advantages and disadvantages but by using those methods the fault coverage is not up to the extent. For this proposed VVM based test programs for Hybrid RISC Controllers. And by using this the test program generation time is reduced which reduces the overall testing time.

REFERENCES

1. Koal T and Vierhaus H T, Kranitis N, Paschalis A, Gizopoulos D and Xenoulis G (2005), "Softwarebased Self-Testing of Embedded Processors", *IEEE Trans. Comput.*, Vol. 54, No. 4, April, pp. 461-475.

2. Psarakis M, Gizopoulos D, Sanchez E and Sonza Reorda M (2010), "Microprocessor Software-Based Self-Testing," *IEEE Design Test Com-put.*, Vol. 2, No. 3, pp. 4-19, May-June.
3. Sabena D, Sonza Reorda M and Sterpone L (2012), "A New SBST Algorithm for Testing the Registernle of VLIW Processors", in Proc. IEEE Int. Conf. Design, Autom. Test Eur., March, pp.412-417.
4. Sabena D, Sonza Reorda M and Sterpone L (2012), "On the Development of Software-Based Self-Test Methods for VLIW Processors", in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, October, pp. 25-30.
5. Sabena D, Sonza Reorda M and Sterpone L (2012), "On the Optimized Generation of Software-Based Self-Test Programs for VLIW Processors", in Proc. IFIP/IEEE 20th Int. Conf. Very Large Integr. Syst. Chip, October, pp. 129-134.
6. Sabena D, Sonza Reorda M and Sterpone L (2014), "On the Automatic Generation of Optimized Software-Based Self-Test Programs for VLIW Processors," in Proc. IEEE *Trans.VLSI*. Vol. 22, April.
7. Ulbricht M, Scholzel M, Koal T and Vierhaus H T (2011), "A New Hierarchical Built-in Self-Test with On-Chip Diagnosis for VLIW Processors", in *Proc. IEEE Symp. Design Diag. Electron. Circuits Syst.*, April, pp. 143-146.
8. Wong S, Anjam F and Nadeem F (2010), "Dynamically Recongurable Register File for a Softcore VLIW Processor", in Proc. IEEE Int. Conf. Design, Autom. Test Eur., Marc, pp. 962-972.