International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 1, No. 1
March 2015

National Conference on "Recent Prends in Communication & Information Technologies"NCRTCIT 2015

Organized by

Dept. of ECE, Indra Ganesan College of Engg., Trichy, Tamil Nadu, India.

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

Int. J. Engg. Res. & Sci. & Tech. 2015

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Special Issue, Vol. 1, No. 1, March 2015
National Conference on "Recent Prends in Communication & Information Technologies" NCRTCIT 2015
© 2015 IJERST. All Rights Reserved

Research Paper

TESTING OF VLIW PROCESSOR WITH TIELINE TECHNIQUE

Janani M[1]* and Kanimozhi V[1]

*Corresponding Author: Janani M

Software based self test has been recently materialized as an effective methodology for detecting faults in the processors. Very long instruction word processors are widely used in many embedded signal processing applications due to facility to provide an instruction level parallelism. The SBST method eliminates the need of high cost testers and provide high fault coverage. The existing SBST technique may lead to be ineffective which generates test programs automatically to all the units to detect the faults. The existing algorithm may lead to maximum test duration and large power consumption. The proposed methodology implements tiling technique to handle the permanent faults. If it detects any permanent faults, the testing control avoids the prohibited zone to apply the test patterns which are permanently faulty. It achieves minimum test duration and consumes less power to detect the faults. The proposed method is designed using Verilog HDL and synthesized using Xilinx Software

Keywords: Software-based self-test (SBST), test program generation, very long instruction word (VLIW) processors

INTRODUCTION

Traditional approaches to improving performance in processor architectures include breaking up instructions into sub-steps so that instructions can be executed partially at the same time known as pipelining, dispatching individual instructions to be executed completely independently in different parts of the processor superscalar architectures, and even executing instructions in an order different from the program out-of-order execution. These approaches all involve increased hardware complexity like higher cost, larger circuits, higher power consumption because the processor must intrinsically make all of the decisions internally for these approaches to work. The VLIW approach, by contrast, depends on the programs themselves providing all the decisions regarding which instructions are to be executed simultaneously and how conflicts are to be resolved. As a practical matter this means that the compiler which means that software used to create the final programs that becomes much more complex, but the hardware is simpler than many other approaches to parallelism.

[1] Department of Electronics and Communication Engineering, Mookambigai College Of Engineering.
[2] Department of Electronics and Communication Engineering, K.Ramakrishnan College of Technology.

A processor that executes every instruction one after the other i.e. a non pipelined scalar architecture may use processor resources inefficiently, potentially leading to poor performance. The performance can be improved by executing different sub-steps of sequential instructions simultaneously this is pipelining, or even executing multiple instructions entirely simultaneously as in superscalar architectures. Further improvement can be achieved by executing instructions in an order different from the order they appear in the program; this is called out-of-order execution.

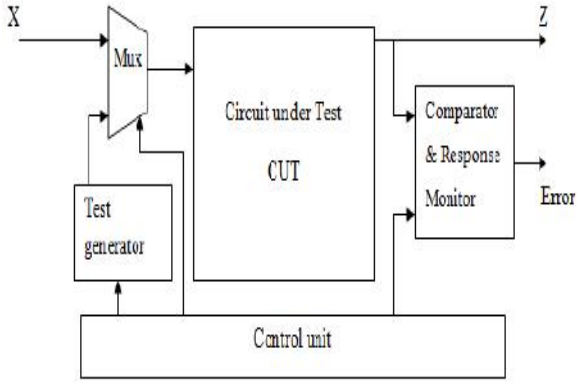
VLIW approach, on the other hand, executes operations in parallel based on a fixed schedule determined when programs are compiled. Since determining the order of execution of operations including which operations can execute simultaneously is handled by the compiler, the processor does not need the scheduling hardware that the three techniques described above require. As a result, VLIW CPUs offer significant computational power with less hardware complexity but greater compiler complexity than is associated with most superscalar CPUs.

TESTING

Since the electronics technology accomplished higher levels of integration into a single Silicon chip that led to Large Scale Integration (LSI), which preceded VLSI, the applications of electronic systems have experienced an almost unlimited expansion. However, despite the many advantages provided by VLSI, the inherent high integration level started to necessitate very sophisticated testing strategies in order to verify the correct device operation.As the electronics market stimulated the use of VLSI in a variety of tasks from critical military applications to consumer products, the reliability of the products' functioning gained an escalating importance. The expanding demand for ASIC applications led to development of more sophisticated Computer Aided Design (CAD) tools; which have shown most significant progress in layout and simulation, with yet more inferior improvement in testing.

Circuit manufacturers must thoroughly test their products before delivering them to customers. The causes of circuit failure can be divided into two main categories: design errors and manufacturing defects. Manufacturing defects are caused by random variations in the manufacturing process that can cause malfunctions in circuit components. These defects can occur even in circuits that have no design errors.

Figure 1: Circuit Under Test



VLIW BACKGROUND

Bolchini C. (2003), proposed "A software methodology for detecting hardware faults in VLIW data paths". In this paper the proposed methodology aims to achieve processor data paths for VLIW architectures able to autonomously detect transient and permanent hardware faults while executing their applications. The advantage of a Software approach to hardware fault detection

does not causing a delay in the execution of noncritical tasks. Here, use of Adopted redundancy Scheme achieves the desired hardware fault detection properties. Each operation concerning data path functional units is executed twice and compared in order to detect possible mismatches. This approach also target faults in the local memory units has limited code size. The drawbacks are performance degradation because that the number of instructions to be executed is doubled; yet the scheduling requires less than twice the execution time.

Gizopoulos D., Psarakis M., Hatzimihail M., Maniatakos M., Paschalis A., Raghunathan A., and Ravi S.(2008), proposed "Systematic software based self-test for pipelined processors". On-Chip processor executes SBST that eliminates the need for high-cost testers, and enables high quality at-speed testing. This paper avoids the problems of exiting SBST methodology i.e. address-related faults, hazard detection This paper propose a systematic SBST methodology to address the faults which improve the fault coverage of the control logic and the dataflow path of the pipeline structure. The proposed solution involves partitioning of a SBST program into multiple code segments that are virtually stored and executed from different memory regions. The drawback of this paper is that the required information cannot be automatically extracted from a high-level description of the processor, e.g., an instruction set description language (ISDL) model, it could be manually specified.

Koal T. and Vierhaus H. T. (2010), proposed "A software-based self-test and hardware reconfiguration solution for VLIW processors". Software-based self-testing is testing approach and provides at-speed testing capability without any hardware or performance overheads. Embedded processor testing techniques based on the execution of self-test programs have been proposed as an effective alternative to classic external tester-based testing. In this paper, we first present a high-level, functional component-oriented, software-based self-testing methodology for embedded processors aims at high structural fault coverage with low test development and test application cost. The proposed High-Level Component-Based SBST Methodology for complex embedded processor applied divide-and-conquer using component-based test development. Test development is based only on the Instruction Set Architecture (ISA) of the processor without the need of low gate-level fine tuning. Here the phases are information extraction from components, Component Classification & Test Priority and Test Routine Development. It achieves high fault coverage with respect to stuck at faults. In Logic Array testing, the proposed methodology is not adequate by targeting the data path functional components, test development should proceed with the other classes of components. In that case, verification based techniques are used test the control and hidden components which cannot guarantee acceptable fault coverage. The limitation of proposed methodology use verification-based functional testing techniques for the control and hidden subsystems has tradeoff between test cost and fault coverage.

Paschalis, Gizopoulos D., Kranitis N., Psarakis M. and Zorian.Y (2001), proposed "Deterministic software-based self-testing of embedded processor cores". SOC designs are based on embedded cores uses a reusable functional block called Intellectual Property. A Deterministic SBST methodology for processor cores is efficiently testing the processor data path modules without

any modification of the processor structure. The proposed methodology of this paper guaranteed to provide high fault coverage without repetitive fault simulation experiments. A processor with more than one functional modules of the same type (multiplier, adder, ALU, shifter) the self-test routines can be easily adopted so that they provide very high for coverage of all modules of the same type. Therefore, the proposed self-test routines can be easily executed by any processor architecture, utilizing the respective instructions of its instruction set. The drawback of this paper is not to evaluate the other parts of rocessor like register file, control parts.

Sabena, Sonza Reorda M. and Sterpone L. (2012), proposed "A new SBST algorithm for testing the register file of VLIW processors". This paper we propose a novel SBST algorithm specifically oriented to test the register files of VLIW processors. This algorithm addresses the cross-bar switch architecture of the VLIW register file by completely covering the intrinsic faults generated between the multiple computational domains. The key characteristics of novel SBST algorithm for VLIW processors are essentially two: the first is that in order to develop the test routine, it considers each component of the processor as a single independent unit, the second feature is that the test development is based only on the Instruction Set

ARCHITECTURE (ISA) OF THE VLIW PROCESSOR

The algorithm essentially focuses on the generation of test instructions for addressing the register file crossbar Switches which are generally implemented using multiplexers anddemultiplexer modules, it is important to analyze the physicalimplementation of this component in order to deal with the internalstuck-at faults. Fault simulation is done by injecting stuck at faultsinto the r-VEX VLIW model. Here first analyzed the structure ofthe register file in order to identify the instructions that properlyexcite this component operation and the instructions for controllingand observing the registers. The drawback is to when analyzing thefaults that remain untested, it is important that the large part ofthem are located on the reset signal of the 2,272 Flip-Flopscomposing the register file, and this signal cannot be controlled viasoftware access, resulting in a 0.87% of faults that are untestable.

VLIW PROCESSOR

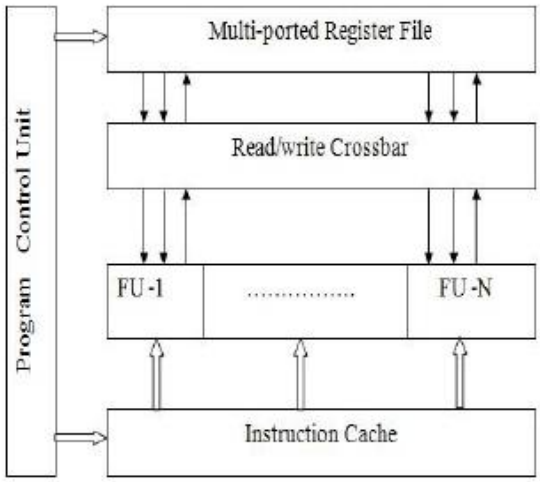
Introduction

Very long instruction word (VLIW) processors areincreasingly employed in a large range of embedded signalprocessing applications, mainly due to their ability to provide highperformances with reduced clock rate and power consumption. Atthe same time, there is an increasing request for efficient andoptimal test techniques able to detect permanent faults in VLIWprocessors. Software-based self-test (SBST) methods are aconsolidated and effective solution to detect faults in a processorboth at the end of the production phase or during the operationallife; however, when traditional SBST techniques are applied to VLIW processors, they may prove to be ineffective (especially interms of size and duration), due to their inability to exploit theparallelism intrinsic in these architectures.

VLIW architectures can be easily customized toefficiently perform any given application. In fact, a generic VLIWprocessor parametric architecture may have a variable number ofCDs and FUs, so that different options, such as the number andtype of FUs, the number of multiport registers (i.e.,

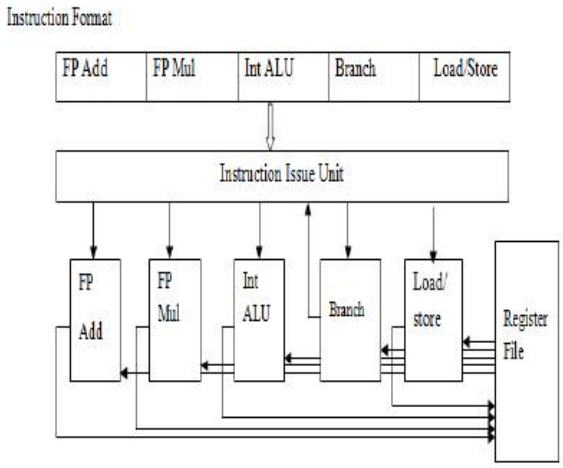
the size of theregister file), the width of the memory buses, and the type ofdifferent accessible FUs, can be modified to best fit the application requirements. All the features of a VLIW processor are grouped together and VLIW processor, several solutions were proposedin order to produce the assembly code suitable for this kind ofprocessors. When generating code for a VLIW processor, theprogrammer or the compiler is faced with the issue of extractingparallelism from a sequential code and scheduling independentoperations, concurrently, to the embedded FUs. For this reason,scheduling algorithms are critical to the performance of a VLIWprocessor.

Figure 2: VLIW Architecture



Many compilers for the first-generation of VLIWprocessors used a three phases method to generate code: first of allthey generate a sequential program, then they analyze each basicblock (a basic block is a sequence of instructions with a singleentry point and a single exit point) in the sequential programlooking for independent operations, and finally they scheduleindependent operations within the same block in parallel.

Figure 3: Instruction Set Architecture



EXECUTION FLOW DIAGRAM

The flow has three initial inputs, which include twoglobal requirements (the VLIW manifest and the library of genericSBST programs), and a specific input (the SBST program fortesting the VLIW register file).The execution flow diagram consists of four steps.

They are

· Fragmentation

· Customization

· Selection

· Scheduling

The VLIW manifest contains all the features of theprocessor under test, while the SBST library contains a set ofprogrammable to test the different modules within the processoritself.

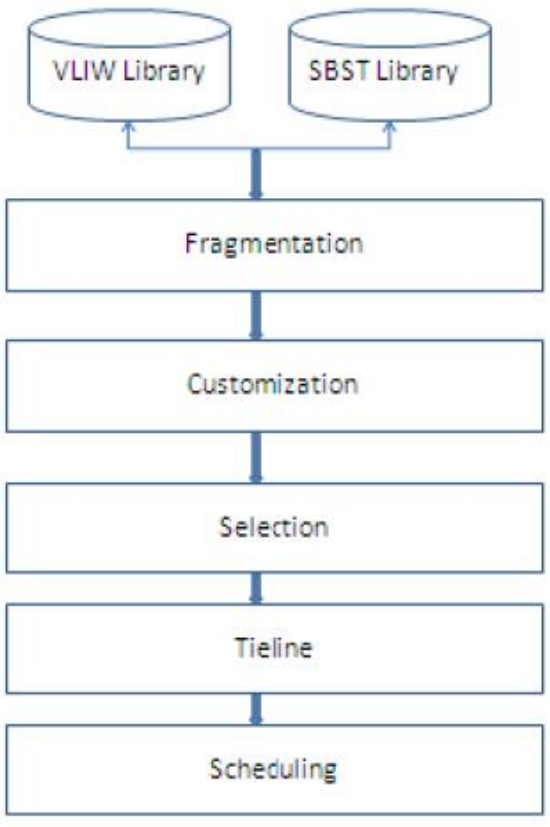
These two requirements are defined as global sincethey are configurable depending on the characteristics of theaddressed VLIW processor. In particular, the library is a collectionof generic SBST programs based on literature test

algorithms, itcontains the functional test code able to test the most relevant FUsof a generic VLIW processor. The test codes stored into the libraryare purely functional and are completely independent on anyphysical implementation of the FU they refer to. These codes maybe based on the techniques used to test the same FUs when used inconventional processors

Fragmentation

The purpose of the fragmentation phase is tominimize the number of test operations in order to create efficientand optimized test programs. The fragmentation phase, illustratedStep A, performs two main tasks. The first is the selection from thelibrary of the test programs needed to test the VLIW processorunder test, ignoring those which refer to FUs that are not part ofthe processor itself. The second task performed by this step is tofragment each selected test program into a set of small pieces ofcode, called fragments, containing few test operations and theother instructions needed to perform an independent test.

Figure 4: Execution Flow Diagram



Customization

The customization step, illustrated in Step B, isresponsible for the translation of the generic architectureindependent test programs into the VLIW code, exploiting the ISAof the considered processor. In particular, starting from thefragments library and from the VLIW manifest, the methodtranslates each generic fragment in a custom fragment that can beexecuted by the processor under test. A custom fragment is definedas a set of instructions belonging to the ISA of the processor undertest, which perform several operations in order to test theaddressed FU.

Selection

The selection of the custom fragments, illustratedin Step C, consists in the choice of the test fragments that optimizea set of rules dependent on the requirements desired for the finalSBST program. The optimization is performed by the execution ofthe algorithm described; the algorithm is able to implement twoalternative rules. The former aims at selecting the minimumnumber of custom fragments that allow reaching the maximumfault coverage with respect to all the resources of the processorunder test.

During this phase, all the fragments are filtereddepending on their fault coverage on the full VLIW processor. Thefiltering of the fragments

is performed by the execution of multiplealgorithm iterations. At each iteration, the algorithm adds to theselected fragment list the fragments that maximize the faultcoverage with respect to all the resources of the processor undertest. In this way, at the end of the execution, several customfragments are not selected, since the faults covered by thesefragments are already covered by the fragments chosen by thealgorithm.

Scheduling

In computing, scheduling is the method by whichthreads, processes or data flows are given access to systemresources (e.g. processor time, communications bandwidth). Thisis usually done to load balance and share system resources effectively or achieve a target quality of service. In real-timeenvironments, such as embedded systems for automatic control inindustry (for example robotics), the scheduler also must ensure thatprocesses can meet deadlines; this is crucial for keeping the systemstable. Scheduled tasks can also be distributed to remote devicesacross a network and managed through an administrative back end

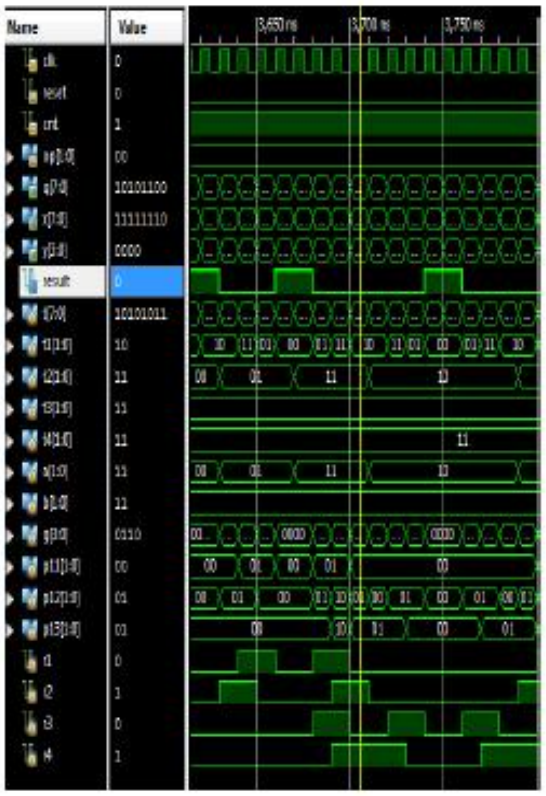
RESULTS AND DISCUSSION

IC View

SIMULATION RESULTS

The Xilinx® ISE Simulator (ISim) is a HardwareDescription Language (HDL) simulator that enables you toperform functional and timing simulations for VHDL, Verilog andmixed language designs.

Figure 5: Simulation Output



Synthesis Results

The Synthesis and Simulation Design provides ageneral overview of designing Field Programmable Gate Array(FPGA) devices using a Hardware Description Language (HDL). Itincludes design hints for the novice HDL user, as well as for theexperienced user who is designing FPGA devices for the first time.The Synthesis and Simulation Design Guide does not addresscertain topics that are important when creating HDL designs, suchas

- Design environment

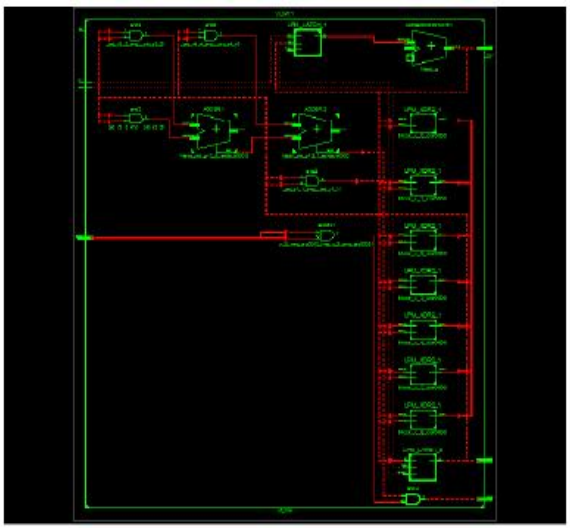
- Verification techniques
- Constraining in the synthesis tool
- Test considerations
- System verification

After the HDL synthesis phase of the synthesisprocess, you can display a schematic representation of yoursynthesized source file. This schematic shows a representation ofthe pre-optimized design in terms of generic symbols, such asadders, multipliers, counters, AND gates, and OR gates, that areindependent of the targeted Xilinx® device. Viewing thisschematic may help you discover design issues early in the designprocess.

After the optimization and technology targetingphase of the synthesis process, you can display a schematic representation of your synthesized source file. This schematicshows a representation of the design in terms of logic elementsoptimized to the target Xilinx® device or "technology,"

RTL Schematic View

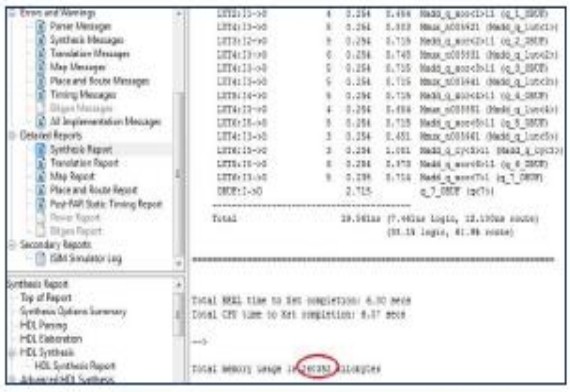
Figure 6: RTL Schematic View



Memory Usage

The total memory usage is 260352 kilobytes

Figure 7: Memory Usage



Power Consumption

Figure 8: Power Consumption

CONCLUSION

The software based self test program is to generateoptimized SBST programs for VLIW processors by exploiting theintrinsic parallelism e. It is fully automatic and allows todrastically reducing the effort of generating test programs forVLIW processors**.** This methodology may lead maximum testduration and power consumption even though it achieves high faultcoverage. The proposed methodology uses the tiling technique inorder to reduce test duration and power

consumption to detect thefaults in the VLIW processor.As future work, we plan to betterevaluate the performances of the proposed solution with the use ofother VLIW models with different FUs and detect more faults.

REFERENCES

1. Bolchini C (2003), "A software methodology for detecting hardware faults in VLIW data paths", *IEEE Trans. Rel.*, ITR.
2. Gizopoulos D, Psarakis M, Hatzimihail M, Maniatakos M, Paschalis A, Raghunathan A and Ravi S (2008), "Systematicsoftware based self-test for pipelined processors", *IEEE Trans.*
3. Koal T and Vierhaus H T (2010), "A software-based selftestand hardware reconfiguration solution for VLIW processors", *in Proc. IEEE Symp. Design Diag. Electron. Circuits Syst.*
4. Kranitis N, Paschalis A, Gizopoulos D and Xenoulis G (2005), "Software based self-testing of embedded processors", *IEEE Trans. Comput., ITC.*
5. Paschalis, Gizopoulos D, Kranitis N, Psarakis M and ZorianY (2001), "Deterministic software-based self-testing of embedded processor cores", *in Proc. IEEE Int. Conf. Design*.
6. Psarakis M, Gizopoulos D, Sanchez E and Sonza Reorda M (Jun 2010), "Microprocessor software-based self-testing", *IEEE Design Test Comput.*
7. Sabena Sonza Reorda M and Sterpone L (2012), "Anew SBST algorithm for testing the register file of VLIWprocessors", *in Proc. IEEE Int. Conf. Design, Autom. Test Eur.*

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com