The International Journal Of Engineering And Science (IJES)
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805
Pages || 37-41 || March - 2015 ||

Efficient Architecture for Proposed VLIW for High Performance in SBST Technology

K.santhana priya[1], (M.E AE), R.valarmathi[2], M.E.,
[1]*Student/Dept of ECE, Anna University Chennai,* [2]*Assistant Professor/Dept of ECE, Anna University Chennai*
Jayam College of Engineering and Technology, Dharmapuri DT, India

ABSTRACT

A promising approach for processors and processor-based systems (e.g., systems on a chip, or SoCs) corresponds to the so-called software-based self-test (SBST) the basic idea 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. To provide high performances with reduced clock rate and power consumption. At the same time, there is an increasing request for efficient and optimal test techniques able to detect permanent faults in VLIW processors. Software based self-test (SBST) methods are a consolidated and effective solution to detect faults in a processor both at the end of the production phase or during the operational life; however, when traditional SBST techniques are applied to VLIW processors, they may prove to be ineffective (especially in terms of size and duration), due to their inability to exploit the parallelism intrinsic in these architectures.

KEY WORDS — *Software-based self-test (SBST), test program generation, very long instruction word (VLIW) processors.*

I. INTRODUCTION

A typical VLIW compiler tries to optimize the parallelism of the instructions exploiting the VLIW resources without any external constraints; in our case, we are considering test programs and thus the instructions composing each piece of code have to be executed in a specific CD and cannot be moved from one to another without modifying the corresponding fault coverage. More in general, the test of a specific unit in a VLIW processor requires performing a well-defined sequence of instructions in a well defined CD. If we encode the test program in a high level language and then launch the compiler, we do not have any way for forcing it to generate a code, which executes the given sequence of instructions on the FUs of a given CD. Consequently, it is not possible to use a VLIW compiler to generate the machine code for testing the processor, nor to use it to optimize the test code. We focused on a specific issue which must be faced when testing a VLIW processor: the register file characteristics are different than in other processors, since it must be accessed from different domains. The generation of effective SBST test programs for the whole VLIW processor, characterized minimal duration, minimal size, and maximal fault coverage. The proposed method starts from existing functional test algorithms developed for each single FU type embedded into the processor (e.g., ALUs, adders, multipliers, and memory units).

Although the characteristic of the FUs used within a VLIW processor are similar to those used in traditional processors, generating optimized code to effectively test these N units is not a trivial task. In fact, by exploiting the intrinsic parallelism of VLIW processors, it is theoretically possible to reduce the increase in duration and size of the test programs when the VLIW processor size grows.

VLIW processors do not include any specially designed hardware module (as it happens for other processor types),but are rather based on a combination of common FUs. Exploiting this characteristic, our solution allows test program generation and optimization to be performed autonomously and automatically, without any manual effort. The test programs generated by the proposed method are highly optimized and exploit the VLIW processor features in order to minimize the test time and the test program size.

II. PREVIOUS METHOD

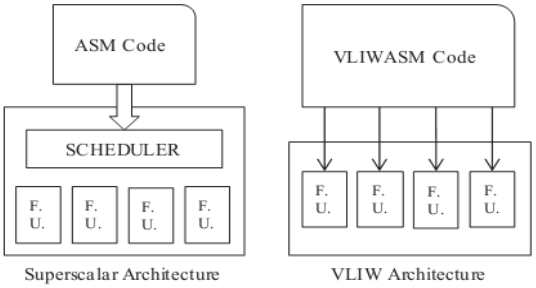
ATPG methods have several drawbacks. First of all, transforming the test patterns generated by the ATPG into test programs is not always straightforward. Secondly, the resulting test programs are far from being optimized, especially in terms of test length, and finally, the attainable fault coverage is not always as high as it may be required.

Unlike superscalar processors, VLIW processors do not include any significant control logic, since instruction scheduling is completely performed by the compiler. This implies that the hardware complexity is far lower for superscalar processors, while the compilation steps become more complicated.

Inconsistent visualizations are present SBST that may be difficult to interpret. SBST has less bandwidth adaption. Pipelined architecture is defined for limited applications only. Area may be high.

III. PROPOSED METHOD

To provide high performances with reduced clock rate and power consumption. We present a new method for the automatic generation of efficient test programs specifically oriented to VLIW processors. The test programs generated by the proposed method are highly optimized and exploit the VLIW processor features To proposed an effective solution to the test of VLIW register files. In the generation of effective SBST test programs for the whole VLIW processor, characterized by minimal duration, minimal size, and maximal fault coverage.



Fragmentation:

The purpose of the fragmentation phase is to minimize the number of test operations in order to create efficient and optimized test programs. The fragmentation phase, performs two main tasks. The first is the selection from the library of the test programs needed to test the VLIW processor under test, ignoring those which refer to FUs that are not part of the processor itself. The second task performed by this step is to fragment each selected test pro-gram into a set of small pieces of code, called fragments, containing few test operations and the other instructions needed to perform an independent test.

Customization:

The customization step, is responsible for the translation of the generic architecture-independent test programs into the VLIW code, exploiting the ISA of the considered processor. In particular, starting from the fragments library and from the VLIW manifest, the method translates each generic fragment in a custom fragment that can be executed by the processor under test.

Selection of the custom fragments:

The test fragments that optimize a set of rules dependent on the requirements desired for the final SBST program. The optimization is performed by the execution of the algorithm is to implement two alternative rules. The former aims at selecting the minimum number of custom fragments that allow to reach the maximum fault coverage with respect to all the resources of the processor under test.

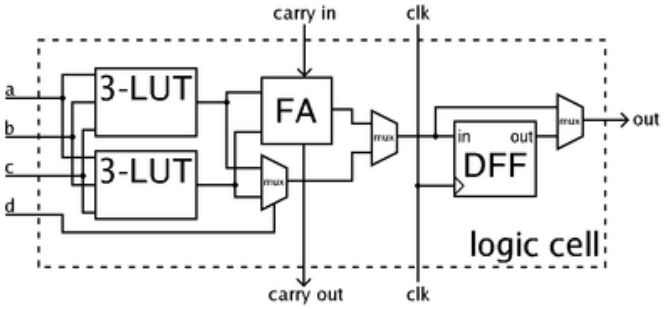
Scheduling Phase:

The last step of the proposed automatic test program generation flow is the scheduling, illustrated in Fig. 3 Step D. The scheduling phase first elaborates the selected custom fragments obtained from the selection phase. This process is responsible for the integration of the custom fragments in order to obtain an optimized and efficient final test program. In order to reach this goal, we developed a scheduler that optimizes and merges the codes contained into the custom fragments exploiting the VLIW features; in particular, it compacts the test programs trying to maximize the ILP of the VLIW processor by an optimal usage of the parallel CDs.

IV. ARCHITECTURE

The most common FPGA architecture [26] consists of an array of logic blocks (called Configurable Logic Block, CLB, or Logic Array Block, LAB, depending on vendor), I/O pads, and routing channels. Generally, all the routing channels have the same width (number of wires). Multiple I/O pads may fit into the height of one row or the width of one column in the array.

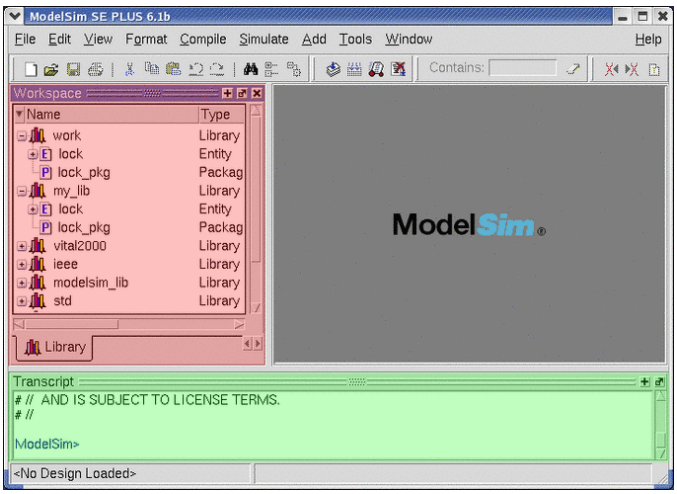
An application circuit must be mapped into an FPGA with adequate resources. While the number of CLBs/LABs and I/Os required is easily determined from the design, the number of routing tracks needed may vary considerably even among designs with the same amount of logicSince unused routing tracks increase the cost (and decrease the performance) of the part without providing any benefit, FPGA manufacturers try to provide just enough tracks so that most designs that will fit in terms of LUTs and IOs can be routed. This is determined by estimates such as those derived from Rent's rule or by experiments with existing designs.



V. EXPERIMENTAL RESULT

Simulation

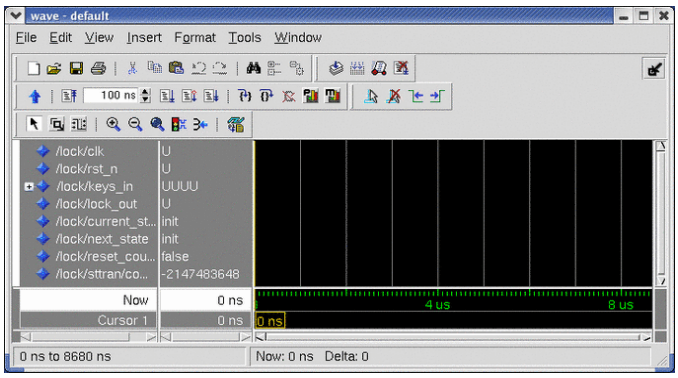
ModelSim simulator is invoked with command vsim. The default window will look something like the picture below. There might also be a welcome window when launched for the first time.



In this tutorial we will mostly write commands in Transcript pane as they will be recorded into the transcript file. Once that we have finished the simulation, we can use the transcript file to create a macro file that can be reused in subsequent simulations.

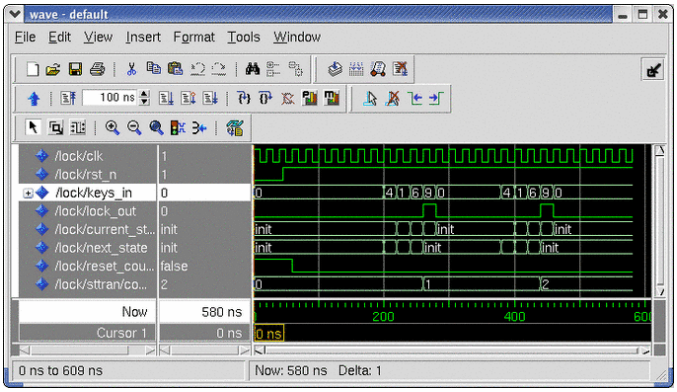
Prepare Simulation

ModelSim Main window showing Workspace, Objects (signals), Locals (variables) and Source panes. There are two processes in this design (sttran and streg), Workspace pane. Locals pane shows variables (count_v) in process currently selected in Workspace.



Run Simulation

All inputs have been set with initial values and everything is ready for a simulation. To run the simulation press the Run button. Run button can be found from both ModelSim Main and Wave window as well as from menubar (Simulate -> Run -> Run 100ns).



VI. CONCLUSION

The obtained results clearly demonstrate the efficiency of the method, that allows to reduce significantly both the number of clock cycles and the memory resources with respect to test programs generated by applying generic SBST methods to the specific case of the VLIW processors. More in particular, the method shows that it is possible to develop test programs whose duration and size grows less than linearly with the VLIW parallelism also compared the test program (denoted as TP) generated by our approach with a test program (denoted as Plain TP) consisting of several test programs developed using some algorithms taken from the literature for the FUs of traditional processors. In the plain TP, these test programs have simply been queued in a unique test program, without performing any selection or scheduling steps, therefore adopting a realistic test estimation of what can be achieved with previously developed test algorithms without any optimization method. This is the only possible approach to have a comparison for the proposed method. To the best of our knowledge in the literature, there is no method aimed at optimizing the SBST routines for VLIW processors exploiting the parallelism that characterizes these architectures. In order to fairly evaluate the two solutions, these test programs have been applied using the loop-unrolling technique, as it is common for any VLIW application.

ACKNOWLEDGMENT

I would like to take this opportunity to express my sincere gratitude to all my professors who have guided, inspired and motivated me for my project work. It gives me immense pleasure to acknowledge their co-operation.

REFERENCE

[1] M. Psarakis, D. Gizopoulos, E. Sanchez, and M. Sonza Reorda, "Microprocessor software-based self-testing," IEEE Design Test Com-put. , vol. 2, no. 3, pp. 4–19, May Jun. 2010.

[2] J. A. Fisher, P. Faraboschi, and C. Young, Embedded Computing: A Vliw Approach to Architecture, Compilers and Tools .Berkeley,CA,USA: Univ. California Press, Dec. 2004.

[3] M. Beardo, F. Bruschi, F. Ferrandi, and D. Sciuto, "An approach to functional testing of VLIW architectures," in Proc. IEEE IEEE Int. High-Level Design Validation Test Workshop, Jul. 2000, pp. 29–33.

[4] D. Sabena, M. Sonza Reorda, and L. Sterpone, "A new SBST algorithm for testing the register file of VLIW processors," in Proc. IEEE Int. Conf. Design, Autom. Test Eur. , Mar. 2012, pp. 412–417.

[5] S. Wong, F. Anjam, and F. Nadeem, "Dynamically reconfigurable register file for a softcore VLIW processor," in Proc. IEEE Int. Conf. Design, Autom. Test Eur., Mar. 2010, pp. 962–972.

[6] S. Wong, T. Van As, and G. Brown, " ρ -VEX: A reconfigurable and extensible softcore VLIW processor," in Proc. Int. Conf. ICECE Tech-nol. , Dec. 2010, pp. 369–372.

[7] Hewlett-Packard Laboratories. EX Toolchain [Online]. Available: http://www.hpl.hp.com/downloads/vex/

[8] J. Fischer, "Very long instruction work architectures and the ELI-512," IEEE Solid, State Circuits Mag., vol. 1, no. 2, pp. 23–33, Sep. 2009.

[9] J. A. Fischer, "Trace scheduling: A technique for global microcode compaction," IEEE Trans. Comput. , vol. 30, no. 7, pp. 478–490, Jul. 1981.

[10] N. Kranitis, A. Paschalis, D. Gizopoulos, and G. Xenoulis, "Software-based self-testing of embedded processors," IEEE Trans. Comput., vol. 54, no. 4, pp. 461–475, Apr. 2005.

[11] T. Koal and H. T. Vierhaus, "A software-based self-test and hardware reconfiguration solution for VLIW processors," in Proc. IEEE Symp. Design Diag. Electron. Circuits Syst., Apr. 2010, pp. 40–43.

[12] M. Ulbricht, M. Scholzel, T. Koa l, and H. T. Vierhaus, "A new hier-archical built-in self-test with on-chip diagnosis for VLIW processors," in Proc. IEEE Symp. Design Diag. Electron. Circuits Syst., Apr. 2011, pp. 143–146.