Cycle-Accurate and Cycle-Reproducible Debugging of Embedded Designs using Artificial Intelligence

Habib ul Hasan Khan, Diana Göhringer
Technische Universitaet Dresden (TUD)
Dresden, Germany
{habib.khan, diana.goehringer}@tu-dresden.de

Abstract— This research work presents an intrusive methodology for debugging of embedded designs by using artificial intelligence technique. In this methodology, a cycle-accurate lossless debugging system with unlimited trace window can be used for debugging. Visibility of the embedded hardware is enhanced by an access network which also eliminates the need for frequent re-synthesis due to change in signal set. The controlling processor can configure the required network through software. A connectivity tool is developed which permits error-free connection of the DUT with the debugging system by using IP-XACT files. A correspondence analysis between the debugging results of the implemented hardware and its simulation results can be performed to speed up the debugging process. The debugging system can be partially reconfigured to any embedded design at runtime which can not only reduce the time spent on iterative place and route process of traditional debugging solutions but also makes the FPGA resources available to the user when debugging is not required.

Keywords—Cycle-accurate and cycle reproducible debugging, Device Start and Stop, Rule-based inference system.

1. INTRODUCTION

Testing and debugging an embedded design is a practical necessity. However, the available solutions have different limitations. Software simulation suffers due to low frequency. Virtual prototyping provides a speed-up for system validation but it may have timing-related issues due to a higher design abstraction level [1].

Speed limitation of the available solutions forced to shift the focus towards hardware prototyping. However, hardware suffers from inherent invisibility. Hence, scan and trace buffer based techniques like Integrated Logic Analyzer (ILA) were devised to increase the hardware visibility. However, such techniques suffer because of limited FPGA resources and data transfer rate limitation of the communication channel.

This research work presents an intrusive debugging technique which permits a cycle-accurate and cycle-reproducible lossless debugging by managing the clock of the Device Under Test (DUT). Cycle-accurate simulation results can be used to perform the correspondence analysis between the simulation and actual results and draw conclusions.

2. RELATED WORK

Cycle-accurate debugging through clock management is not new. In [2] a methodology is discussed which generates continuous stream of lossless data by automatically stopping the clock based upon the occupancy of the trace buffer. However, manual analysis of this enormous amount of data becomes a bottleneck. Although debugging through golden reference has been proposed [3], data analysis is still performed manually. In this paper artificial intelligence techniques like an inference system are suggested to avoid the manual analysis process by automating the data analysis.

3. DESIGN METHODOLOGY

In ILA based debugging methodologies, the debugging data is limited to the capacity of the trace buffers. Once they are full, data is sent to the terminal for analysis. However, debugging data generation is faster than the data transmission resulting in data loss. This loss forbids the cycle-accurate debugging.
In our proposed methodology, we suggested to stop the clock for DUT during the data transmission phase. The data transmission phase does not contribute towards DUT activity hence producing a cycle-accurate and cycle-reproducible debugging data as shown in Figure 1.

Trace-based cycle-accurate debugging is important because it provides an accurate description of the state of the DUT on each clock cycle. Cycle-reproduction makes it possible to replay the original faulting execution cycle-by-cycle. Hence post-silicon, complete cycle-accurate details can be reconstructed.

The debugging system can only monitor limited number of signals due to the scarce hardware resources hence restricting the visibility. The visibility of the embedded hardware is increased by adopting an access network. Access networks eliminate the need for frequent re-synthesis of the design due to a change in the observed signal set. Using supervisory control by a processor, required networks can be configured through software. The access network can be connected to a large number of nodes. Error free connection of such nodes is also a complicated issue. Utilizing IP-XACT files of the DUT for automatic connectivity generation can solve such problems. A Tcl file can then be used which can perform automatic network generation.

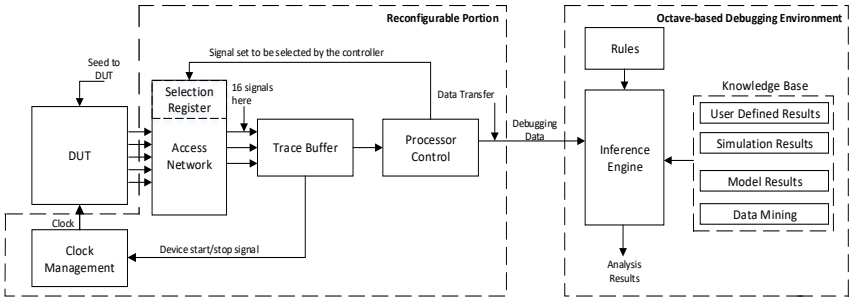
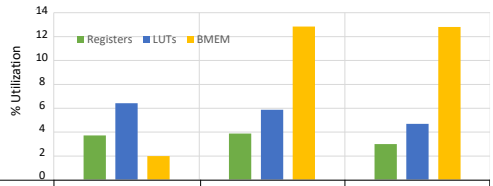


Figure 1: Trace-based debugging



Utilization Qty. (%)	Window 64	Window 1024	DSAS[2]
Registers	3968 (3.72%)	4144 (3.89%)	3192 (3%)
LUT	3418 (6.42%)	3123 (5.87%)	2511 (4.7%)
BRAM	2 (2%)	18 (12.85%)	17 (12.8%)

Figure 2: Resource utilization

Moreover, debugging systems are instantiated after observing the problem at post-silicon stage. Hence, the complete Place-and-Route (PAR) process is repeated. Furthermore, the debugging system is removed after the completion of the debugging process resulting in the repetition of the PAR process. Re-evaluation of the PAR process may introduce some routing or timing errors. This issue was resolved by using Dynamic Partial Reconfiguration (DPR). The DUT is present in the static region and the debugging system is placed in the reconfigurable region. Once required, the debugging system can be dynamically configured to any embedded design at runtime. This can not only reduce the time spent on an iterative PAR process of traditional debugging solutions but also makes the FPGA resources available to the user when debugging is not required. Furthermore, the debugging system can be replaced with a blank bitstream upon completion of debugging process. This removes the possibility of routing errors since the PAR process will not be repeated.

Furthermore, manual analysis of the test results constitutes a bottleneck for cycle-accurate and cycle-reproducible debugging systems due to the enormous lossless stream of data. A rule-based inference system was utilized in this research work which can perform correspondence analysis and draw conclusions. The debugging data of the implemented hardware and its either behavioral or post-synthesis/post-implementation simulation results are used as input to speed up the debugging process. The rule-based inference system is used for decision making without the need to go through all the values manually hence providing a time-efficient solution.

An inference system (Fig.1) incorporates a knowledge base containing accumulated experience and an inference engine which applies the knowledge base to each particular situation described to the program based upon certain rules. The system's capabilities can be enhanced by additions to the knowledge base or to the rule set. The rules set comprises of either the auto correlation, cross correlation or regression. The knowledge base can have one of the four types of datasets. The priority lies with the user-defined data set. It is also possible to use the HDL simulation results or the high-level modeling results. If either the user-defined datasets or the simulation results are not available, the debugging system can mine for one in the database (if a similar system was debugged in the past). The inference system then performs a correspondence analysis between the datasets to identify the suitable rule.

4. Results

The rule-based inference system carries out the cross correlation (depends upon the rule selection) between the two input datasets and displays the results. If a disparity between the results exists, the cross-correlation coefficient will have a value less than 1. The inference system is still helpful because it can point towards the lag at which maximum cross-correlation was found hence providing a starting point for further investigation.

4.1 Resource Utilization

Resource utilization of the presented debugging methodology is shown in Fig. 2. The debugging system is synthesized with a trace window of 64 and 1024. 16 signals are monitored with a maximum data width of 32 bits. We have also compared our resource utilization with our previous work (DSAS with a sample window of 64) [2]. At the expense of a minor increase in LUTs and registers utilization, we have been able to reduce the BRAM utilization significantly.

4.2 Power Utilization

Another benefit of the technique is the low power consumption. Managing the clock for data dispensation imitates clock gating which is known to reduce the power consumption [2].

4.3 Deployment Time

Using DPR technique to employ our debugging system took 260ms, this is the time needed to reconfigure the FPGA at runtime using the JTAG mode of configuration. In contrast the traditional flow requires minutes to hours depending upon the complexity of the DUT.

Acknowledgements

This work is funded by German Research Foundation (DFG) via project SFB/TRR 196 “MARIE” through project S05.

References

[1] Huang, C., Yin, Y., Hsu, C., Huang, T. B., and Chang, T. "SoC HW/SW verification and validation." In Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2011.

[2] Khan, H. H., Kamal, A., and Göhringer, D. " An Intrusive Dynamic Reconfigurable Cycle-Accurate Debugging System for Embedded Processors." In Proceedings of the International Symposium on Applied Reconfigurable Computing, pp. 433-445. Springer, Cham, 2018.

[3] Katrowitz, Michael, and Lisa M. Noack. "I'm done simulating; now what? Verification coverage analysis and correctness checking of the DEC chip 21164 Alpha microprocessor." Proceedings of the 33rd annual Design Automation Conference. ACM, 1996.