Proceedings of the XXV Conference on Design of Circuits and Integrated Systems

DCIS2010

Lanzarote, Spain – November 17-19, 2010

ISBN: 978-84-693-7393-4

Editors

José F. López, IUMA-U. de Las Palmas de Gran Canaria
Roberto Sarmiento, IUMA-U. de Las Palmas de Gran Canaria
Valentín de Armas, IUMA-U. de Las Palmas de Gran Canaria
Sebastián López, U. IUMA-U. de Las Palmas de Gran Canaria
Gustavo Marrero Callicó, IUMA-U. de Las Palmas de Gran Canaria
Félix Tobajas, IUMA-U. de Las Palmas de Gran Canaria

Session W4B. Digital IC Design

Session W4C. System Design Methodologies

Session W4D. RF IC Design II

Long-term on-chip verification of systems with logical events scattered in time

J. Viejo, J. I. Villar, J. Juan, A. Millan, E. Ostua, and J. Quiros
Grupo ID2 (Investigacion y Desarrollo Digital)
Departamento de Tecnologia Electronica-Universidad de Sevilla
E. T. S. Ing. Informatica, Campus Universitario Reina Mercedes
41012 Sevilla (SPAIN)
Email: {julian,jose,jjchico}@dte.us.es, amillan@us.es, {ostua,jquiros}@dte.us.es

***Abstract*—Traditionally, in the design and development of digital electronic systems, verification and debugging are the costliest tasks in time and money. To improve the efficiency of this process the market has provided several solutions to facilitate both off-chip and on-chip debugging of high speed systems. However, there are other types of designs, such as network synchronization systems, in which the verification of their correct operation requires the analysis of events scattered in time over a period of days, weeks or even months. To verify these systems, ad-hoc cores are developed for every specific design. To avoid this, a customizable on-chip verification core based on PicoBlaze, a data analysis software and a verification methodology have been developed. This paper presents the main features of the developed core comparing it in several aspects against already existing commercial alternatives.**

I. Introduction

Nowadays, digital electronic systems development has experienced a tremendous growth due mainly to the popularization and low costs of programmable devices such as FPGAs, which are now affordable for an increasing number of designers and companies. This popularity, coupled with the fact that the use of FPGAs beyond the prototype stage can lead to a shorter time-to-market and lower manufacturing cost for low and medium volume production, makes essential the adoption of efficient methods and tools for debugging and verification.

Traditionally the two main methods for FPGA design debugging and verification are simulation and hardware execution [1]–[3]. In the first stage of the design flow, simulation allows debugging and verification of the correct operation of the design. This is really true in systems whose proper operation can be determined over a short period. In certain types of designs, there are situations where simulation is not a feasible approach, mainly because it would need billions of simulation cycles to reach certain points of its operation, which would take days or even weeks to complete. On the other hand, hardware execution permits the observation of the design under study in real time during its operation. Such observations are taken in several ways including external logic analyzers and on-chip logic analyzers such as Chipscope [4]. Currently there are numerous commercial and non-commercial alternatives for on-chip logic analysis with a common feature, all of them enable the acquisition of data at very high speed with an horizon limited by the storage memory available in the chip [5], [6].

While the above mentioned limitation is not a serious problem for the verification of many types of systems, there are others where it is necessary to verify the correct long-term operation. In these systems, the verification process is the result of the analysis of a large number of events distributed over a long period of time. An example is a network synchronization system previously developed [7] where data collection is spaced in the order of seconds but spans over a period of several days or even months.

Although the market offers a wide spectrum of solutions for monitoring and debugging digital electronic systems, both on-chip and off-chip, none of them fits perfectly to the monitoring of distant events over a long period of time, making it necessary to develop custom tools and test logic for each design. In this contribution we present the development of a tool and logic components that can be adapted to any system in which the verification is based on the capture of events scattered in long periods of time. The paper is organized as follows: in section two an outline of current verification tools is made, analyzing them from the point of view of their applicability to scattered event acquisition and analysis. Section three presents the architecture of the long-term on-chip data acquisition system that has been developed. Section four presents a real application where the proposed tool and the commercial ChipScope Pro test system are compared with respect to several parameters. Finally, section five outlines the most relevant conclusions derived from this experience.

II. Current solutions for system verification

Currently there are three main types of solutions when approaching digital system verification: standalone logic analyzers, on-chip logic analyzers and custom cores for specific purposes. In this section we will highlight the main features, advantages and disadvantages of each one from the perspective of their applicability to long-term verification.

Standalone Logic Analyzers (SLAs) are very powerful tools for debugging an already implemented design. This kind of equipment is able to acquire data at a very high frequency from any signal that can be accessed at the pins of the chip. Moreover, they may have a large number of channels (100 or more) that makes them a very useful tool for debugging

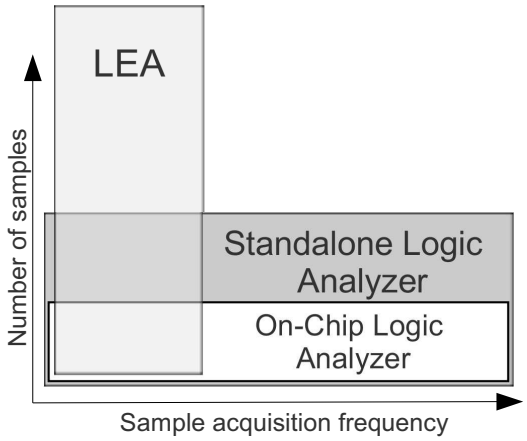

Fig. 1. Field of applicability of several verification solutions.

high speed buses and signals between components. The main disadvantage of SLAs is that they cannot reach signals inside the design. To overcome this issue, designs are modified in order to route the desired signals to external pins accessible by the SLA thus modifying the characeristics and timing parameters of the original design.

An evolution of SLAs are On-chip Logic Analyzers (OLAs) like Xilinx's ChipScope, that have become very popular in the field of programmable logic. This kind of analyzers are hardware components that connect to the desired signals inside the chip and communicate over a standard bus (usually RS232 or JTAG) with a computer that executes software for data analysis. These components are an intrusive solution since the verified design is different from the production design when analysis components are removed.

These two types of verification tools have a common denominator: the limit of the capture size is given by the size of the storage memory since they store a complete capture frame before sending it to the processing unit.

Some tests require to capture some kind of events from within the system continuously. To capture these events, developers usually create custom debugging cores (CDCs) for every specific purpose when SLAs or OLAs do not fulfill the verification requirements [8], [9].

III. Logical Event Analyzer

To overcome the cost of designing a CDC for every application, we propose a general purpose device, the Logical Event Analyzer (LEA), that can fill the gap between SLA and OLA and substitute CDC in several practical cases. As it can be observed in Fig. 1, SLAs and OLAs are used to verify high-speed systems where the number of samples is not a critical aspect. However, LEA can be used for debugging systems where it is neccesary to capture a large number of samples spaced in the time. The LEA also features a much lower footprint than an OLA.

The architecture of the proposed analyzer is based on the PicoBlaze microprocessor from Xilinx [10]. The Fig. 2 shows

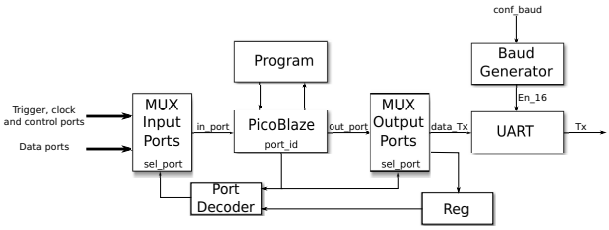

Fig. 2. Architecture of the designed analyzer.

TABLE I
Bit Rates calculated for some typical Baud Rates.

Baud Rate	Bit Rate (bps)
4800	3840
9600	7680
19200	15360
38400	30720
57600	46080
115200	92160

the block diagram of the designed analyzer. As it can be observed from the diagram, a set of input ports of PicoBlaze are reserved for trigger, clock and communication control signals. The remaining ports are dedicated to capturing data signals. PicoBlaze allows for the addressing of 256 8-bits ports. Thus, using a single PicoBlaze module and dedicating N ports to trigger, clock and control signals the analyzer can capture $(256 - N) * 8$ bit signals. If it is necessary to capture more signals a simple solution is to add an external n bits register to extend the port selection signal *port_id*. By using this alternative, the maximum number of signals that can be sampled would be $(256 - N) * 2^n * 8$. The number of data signals that can be acquired is also limited by the rate at which captured data can be transmitted out of chip as we will discuss later.

PicoBlaze uses one of their output ports to communicate with an UART that is in charge of transmitting the captured data via serial. The UART has a *half_full_buffer* signal which indicates that the FIFO is half full, and its baud rate can be set as needed. PicoBlaze will use the *half_full_buffer* signal to control the data transmission to the UART. Assuming the following UART configuration: fixed data format "8N1" (8 data bits, 1 stop bit and no parity), communication through the RX and TX signals (no other signals needed), and flow control disabled, the maximum bit rate (bps) that can be obtained is calculated according to (1).

$$BitRate = \frac{BaudRate * 8}{10} \quad (1)$$

Bit rates calculated for some typical baud rates are shown in Table I.

Finally, the program module corresponds to the program that will be run by PicoBlaze. This program performs the following tasks:

1) Trigger condition verification. PicoBlaze reads the ports

assigned to the trigger signals and applies the configured logical function. In the case this condition is verified, data acquisition begins.

2) Data acquisition according to the clock signal. The clock signal corresponds to an event of the designed system, so that whenever this event occurs PicoBlaze starts to read the data connected to the input ports.
3) Data processing and transmission. This processing consists of calculating a checksum of the transmitted data so that the receiver can verify the correct reception of information. Data will be sent to the UART.
4) Communication control. Periodically, the microprocessor will check the status of *half_full_buffer* signal. If it is active PicoBlaze will wait for the FIFO to becomes half empty before sending more data.

IV. Application Example and Results

In this section we describe the application of the LEA to the on-chip verification of a SNTP client and server fully implemented in hardware. SNTP is a simplified version of the more general Network Time Protocol (NTP) [11] that is commonly used for synchronizing the clocks of computer systems over data networks such as the Internet. The operation of this protocol is to send periodic time requests to a server synchronized with an accurate time source like a GPS receiver at request intervals that can vary from a few seconds to several minutes. When the server reply is received, the client uses a set of timestamps to calculate the round trip time and the time offset between the client's and server's clocks.

The client can then adjust its local clock based on these calculations. In a typical scenario, the client will be accurately synchronized to the server only after several request-response cycles. Since the time between requests can vary from a few seconds to several minutes the most important aspect in the testing analysis of these systems is not the speed at which samples are acquired but to capture a large number of system events, covering a wide time interval.

On-chip verification tools like ChipScope Pro feature high frequency sampling which allow the testing of high-speed buses and systems, but they face some limitations regarding the maximum number of samples that can be obtained from the system. This is mainly due to: 1) internal resources of the FPGA are used to store the samples, and 2) some of these resources, depending on the type of programmable device used, are often limited. The number of LUTs, FFs and BRAMs used by ChipScope depending on the number of signals and the number of samples are shown in Figures 3, 4, and 5, respectively. As it can be observed from Figures 3 and 4, LUTs and FFs depend mainly on the number of signals. However, Figure 5 shows that the main problem when performing on-chip verification using such tools is that the number of BRAM used is directly proportional the number of signals and the number of samples. Furthermore, an additional BRAM must be included for each added Integrated Logic Analyzer (ILA) since each ILA only can capture a maximun of 256 signals. The total number of BRAMs is calculated according

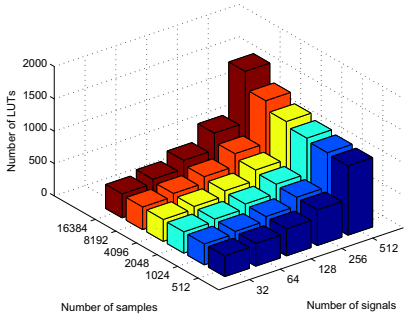

Fig. 3. Number of LUTs dependency of number of signal and samples using ChipScope.

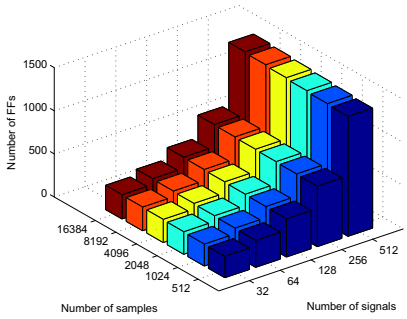

Fig. 4. Number of FFs dependency of number of signal and samples using ChipScope.

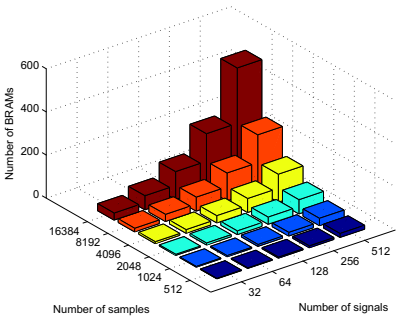

Fig. 5. Number of BRAMs dependency of number of signal and samples using ChipScope.

to (2). Thus, considering that BRAMs are the most limited resource and fixing a number of signals to capture, this type of simulation is unfeasible if the goal is to capture a large number of samples.

$$NumBRAMs = \left| \frac{NumSignals \times NumSamples}{BRAMsize(bits)} \right| + \\ + NumILAs \quad (2)$$

For the case under discussion, the SNTP client and

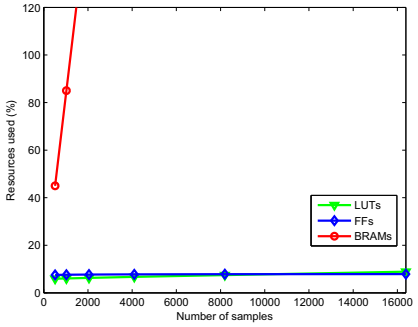

Fig. 6. Percentage of used resources dependency of the number of samples for 256 signals using ChipScope.

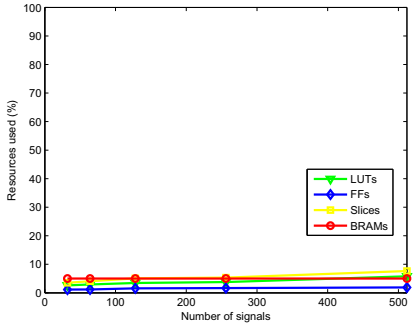

Fig. 7. Percentage of used resources dependency of the number of signals using LEA.

server have been implemented on a Spartan-3E FPGA device (xc3s500e). These FPGAs have a total of 20 BRAMs and each block contains $18,432$ bits of fast static RAM, with 16 Kbit allocated for data storage. For each design a total of 256 signals have been sampled: timestamps (least significant part), time offset, round trip time and adjustment parameter of the local clock. With this configuration, the percentage of used resources (LUTs, FFs and BRAMs) to verify the system in terms of number of samples is shown in Figure 6. As shown in that figure, the LUT and FF usage is not critical, since it is a 9% and 8% respectively of total resources in the worst case. However, there is an excessive use of BRAMs even for a small number of samples. A maximum of 1024 samples can be captured which is insufficient for the type of system that is intended to verify.

The developed LEA has the advantage that it does not store data in BRAM but transmits them via serial. Therefore the number of BRAMs used to verify the system does not dependent on the number of signals or the number of samples so the system can be tested indefinitely, only limited by external resources. For the rest of the FPGA resources, they become solely dependent on the number of signals (Fig. 7). For the same scenario presented for ChipScope (sampling of 256 signals) the percentage used of LUTs, FFs, Slices and BRAMs has been 3,79%, 1,68%, 5,35% and 5% respectively. It is worth noting that only a BRAM is used (this memory stores the program that will be run by PicoBlaze).

V. Conclusion

In this contribution, a verification core based on Picoblaze for long-term on-chip verification is presented. The proposed solution allows developers to avoid the implementation of custom verification cores in many cases, greatly improving the design and verification time.

This core has been compared to ChipScope Pro on-chip logic analyzer in the verification of a real synchronization system. The results show that the ChipScope Pro tool is not suitable to verify the system because this would need excessive internal resources to store the captured data even for a small number of samples to acquire. The proposed core does not store data using internal resources but transmits them via serial port so the system can be verified indefinitely, only limited by external computer storage.

Acknowledgment

This work has been partially supported by the Ministry of Education and Culture of the Spanish Government through the TEC2007-61802/MIC (HIPER) project and the PROFIT-MITC SEPIC TSI-020100-2008-258 project.

References

[1] T. Wheeler, P. Graham, B. Nelson, and B. Hutchings, "Using Design-Level Scan to Improve Design Observability and Controllability for Functional Verification of FPGAs," in *2001 Proceedings International Conference on Field-Programmable Logic and Applications (FPL)*, Belfast, Northern Ireland (UK), Aug. 2001, pp. 483–492.

[2] P. S. Graham, "Logical Hardware Debuggers for FPGA-Based Systems," Bringham Young University, Department of Electrical and Computer Engineering, PhD Dissertation, 2001.

[3] N. Ohba and K. Takano, "Hardware debugging method based on signal transitions and transactions," in *IEEE Proc Asia and South Pacific conf on Design Automation*, Yokohama (Japan), Jan. 2006, pp. 454–459.

[4] Xilinx, *ChipScope Pro 11.1 Software and Cores User Guide*. Xilinx, Inc., Apr. 2009.

[5] K. Arshak, E. Jafer, and C. Ibala, "Testing FPGA based digital system using XILINX ChipScopeTM logic analyzer," in *29th International Spring Seminar on Electronics Technology (ISSE)*, St. Marienthal (Germany), May 2006, pp. 355–360.

[6] L. Ehrenpreis, P. Ellervee, and K. Tammemae, "Open Source On-Chip Logic Analyzer for FPGAs," in *2006 International Baltic Electronics Conference*, Tallinn (Estonia), Oct. 2006, pp. 1–4.

[7] Authors, "Design and implementation of a SNTP client on FPGA," in *Proc. 2008 IEEE International Symposium on Industrial Electronics (ISIE)*, Cambridge (United Kingdom), Jun. 2008, pp. 1971–1975.

[8] T.-Y. Lee, Y.-H. Fan, S.-C. Yen, C.-C. Tsai, and R.-S. Hsiao, "An Integrated Functional Verification Tool for FPGA Systems," in *International Conference on Innovative Computing, Information and Control*, Kumamoto (Japan), Sep. 2007, p. 203.

[9] G. Knittel, S. Mayer, and C. Rothlaender, "Integrating Logic Analyzer Functionality into VHDL Designs," in *2008 International Conference on Reconfigurable Computing and FPGAs*, Cancun (Mexico), Dec. 2008, pp. 127–132.

[10] K. Chapman, *PicoBlaze 8-Bit Embedded Microcontroller User Guide for Spartan-3, Virtex-II and Virtex-II PRO FPGAs*. Xilinx, Inc., Nov. 2005.

[11] D. L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," RFC 1305 (Draft Standard), Mar. 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1305.txt