International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.3, Issue.1, Jan-Feb. 2013 pp-317-323 ISSN: 2249-6645

On Chip Bus Tracer Based On Reverse Encoding In Soc

Blessy Babu [1], Karthika Manilal [2]
[1]PG Scholar, Dept.of Electronics and Communication Engg, TKM Institute of Technology, Kollam, Kerala, India
[2]Asst.Professor, Dept. of Electronics and Communication Engg, TKM Institute of Technology, Kollam, Kerala, India

***Abstract*:** *System on chip (SoC) is the integration of different components in to a single chip targeting a specific application. Real-time observability of the internal chip signals is crucial to SoC debugging, the obvious choice would be to use chip pins to observe them. However, this method is difficult to implement in the presence of high frequency internal clocks and limitation of the chip pins. A solution to this problem is to embed a bus tracer in SoC to capture bus signal, and to store the trace in on-chip storage (trace memory). The size of bus trace increases rapidly as the numbers of transactions are more. In order to reduce the trace size, compression is necessary. These compression mechanisms include slice compression, differential compression and dictionary based compression. The compression algorithms use forward encoding method i.e. first data is recorded as uncompressed and all others are encoded. Trace memory is a circular buffer, so in case of wrapping around this may result in loss of data and affects the decompression flow. Therefore a new reverse encoding method is used that results in efficient circular buffer utilization. To capture the bus signals, an on-chip communication architecture based on WISHBONE bus, with four masters and four slaves is designed and the hardware tracer based on reverse encoding is integrated to this. This is coded using VHDL, simulated using ModelSim 6.4c and synthesized using Xilinx ISE tool.*

I. INTRODUCTION

The growing complexity and shorter time to market constraint makes verification and debugging a major hurdle in shortening development cycle of system on chip.

In contrast to software debugging, hardware debugging methods provides insight to observability of internal state system and real time behavior in SoC. The limitation of this approach is unavailability of dedicated pins and increased gap of internal and external frequencies while moving trace out of chip. This leads to demand for on chip debugging units. The main problem dealing with the on chip debug units is that execution trace in SoC is large and hence for storing these much data results in increased cost of hardware implementation and data acquisition systems. The solution to this problem is to integrate a tracer in to the bus, which is the communication architecture in SoC, to capture and compress signal in real time and store them in an on-chip trace memory. The traced data can then be decompressed on the software terminal for performance analysis, bus utilization and verification.

Trace collection is controlled by user defined signal condition. They are two types of traces forward trace and backward trace. As shown in figure 1 the forward/backward trace refers to trace captured before/after a triggering event.
The forward trace is used for diagnosing errors at known time periods and backward trace is used for detecting the system
Backward trace Event trigger Forward trace

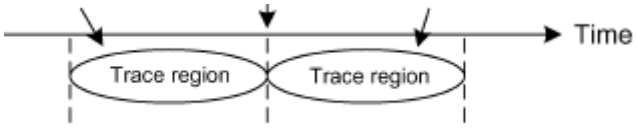

Fig.1. Difference between forward trace and backward trace

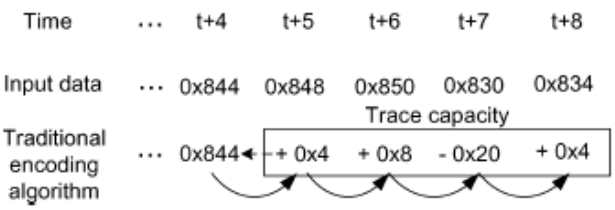

Fig.2. forward encoding differential compression algorithm malfunctioning.

Traces which are collected are stored in a trace memory; since capacity is limited it is usually used as circular buffer. In case of backward trace, signals are continuously collected, so when trace memory is full, wrapping around occurs i.e., initial data become overwritten. This causes a problem when the trace needs to be compressed.

For example, Fig.2 shows traditional differential compression algorithm that make use of forward encoding. It works as follows. First datum is recorded as uncompressed; when second datum comes it is encoded with respect to first, ie, difference between the first and second is stored. When third datum comes it is encoded with respect to second and so on. The problem occurs when first data become overwritten due to wrapping around in circular buffer. This initial data is in uncompressed format and all other dates are encoded with respect to that .Therefore when it is lost all other data's becomes useless ie, they cannot be decompressed.

In this paper a reverse encoding algorithm is proposed. This algorithm is applied to different popular compression algorithms and demonstrated in an on WISHBONE bus for SoC. The tracer supports both forward and backward trace with effective circular buffer utilization in addition to good compression ratios and small hardware overhead.
This paper is organized as follows. Section II discusses related work regarding tracers. Section III presents the proposed algorithm. Section IV introduces architecture of on chip bus tracer. In Section VI, the experimental results a are discussed. Section VIII concludes this paper.

II. RELATED WORKS

The execution traces in SoC is large so there needs a large amount of memory for storing these data's .This results the need of reduction/compression techniques. They techniques are divided into lossy and lossless approaches.

Lossy approach has the advantage of high compression ratio [3]. The idea is to capture only the erroneous signals in a trace buffer to increase the trace-buffer utilization. Here a three pass methodology is used. During the first pass, the rough error rate is measured, in the second pass, a set of suspect clock cycles where errors may be present is determined, and then in the third pass, the trace buffer captures only during the suspect clock cycles. In this manner, the effective observation window of the trace buffer can be expanded significantly, by up to orders of magnitude. This greatly increases the effectiveness of a given size trace buffer and can rapidly speed up the debug process.. But it is not suitable for real time on chip bus tracing

Lossless data compression algorithms can be classified into two main categories statistical coding algorithms and dictionary coding algorithms [5] Statistical based compression algorithms, such as Huffman coding or arithmetic coding can lead to an optimal average code length and hence good compression ratio. For the real-time requirement and the fact that debug data is not known a-priori these static methods are not suitable. The second category of data compression algorithms is the dictionary-based compression algorithms. The compression in these methods is achieved by encoding a symbol or a sequence of consecutive symbols into shorter code words which are represented by indices to the dictionary locations. But this approaches face difficulty when wrapping around occurs because they use forward encoding algorithm.

For effective debugging they must support both forward and backward trace .The logic analyzers [7] support forward trace but they have no compression ability and backward trace capability.

In ARM ETM [6] and NEXUS interface, data filtering is used to increase trace depth. They use a branch target technique to compress instruction address: it ignores consecutive address and stores only branch and target address. The hardware overhead of these works is small since
the filtering mechanism is simple to implement in hardware. However, the effectiveness of these techniques is mainly limited by the average basic block size, which is roughly around four or five instructions per basic block,

Periodical triggering supports both forward and backward trace[4].Here trace memory is divided into small segments using a ping pong organization .These segments are compressed separately hence destruction of one segment does not affect another. But has a disadvantage of more uncompressed data. Utilization ratio can be increased by increasing the segments but it reduces the compression ratio.

III. REVERSE ENCODING ALGORITHM

To overcome the disadvantage of data loss due to wrapping around in circular buffer a reverse encoding is applied to different compression algorithms in trace reduction stage.. In this newest data is set to uncompressed and all other preceding datum's are encoded with respect to the newest so when first data is removed, it doesn't affect the decompression because it is only a encoded value not the initial uncompressed data. As in reverse encoding, there is a difference from Fig.2.i.e.0x834 is set as uncompressed and all others are encoded with respect to that.

IV. SYSTEM ARCHITECTURE

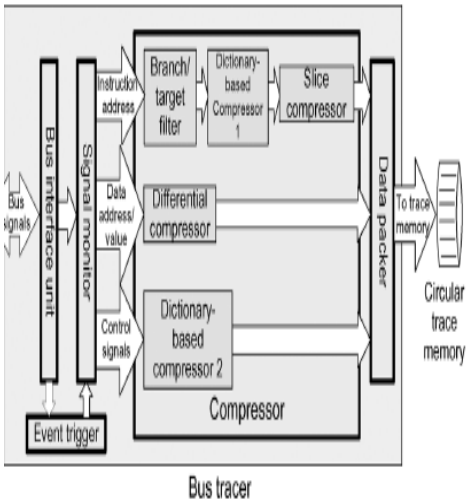

Fig.3.Bus tracer hardware architecture

Fig. 3 shows the hardware architecture of the bus tracer. It has two major functional modules (1) signal monitor /tracing and (2) trace compression. Signal monitor consist of event trigger, bus interface unit and signal monitor. Bus interface unit handles bus requests and collects the bus signal for storing, signal monitor synchronizes the signals and pass these signals to data reduction stage, where reverse encoding is applied. Event trigger module is configurable and controls trigger condition of start/stop a trace. When triggered the bus tracer starts capturing signals and stops until another trigger happens. Finally, the data packer packs the compressed data and writes them to the circular trace memory.

As shown in Fig. 3 bus signals are classified into three, instruction address, data address/value, and control signals. Instruction addresses are processed in three steps. First is to record only the non **s**equential jumping address. In virtual of the computer program usually having the property of spatial locality, each adjacent address pattern often is increased by a sequential offset .Hence, we can choose to only record the adjacent address pattern increased by a non-sequential offset, also called branch jump address, so as to eliminate the most address information. Secondly such branch/target addresses repeat several times because most programs are recursive. Therefore, dictionary-based compressor is used to reduce these repeated addresses. Third, the higher part between two adjacent address patterns is almost similar. Hence, if we can make the similarity between the current address and the previous address to be eliminated, the compression ratio may be rising dramatically. So the dictionary contents from dictionary-based compressor 1 are further compressed by a slice compressor. For control signal**s**, we use dictionary-based compressor 2 (shown in Fig. 3). Because the variety of transfer types in a system is usually small, the same transfer types appear frequently. Therefore, the control signal patterns indicating the transfer types also appear frequently, which enables the dictionary-based method to reduce the size of the repeated signal patterns. The signal variations on data bus (data address and data value buses) are not regular that compared with program address bus. Using the differential approach based on subtraction is the convenience way to reduce the data bus trace size and the hardware cost of subtraction module is small.

The following sections show how to apply our reverse-encoding algorithm to the compression methods used in this bus tracer

A. Differential Compression

Compared to forward encoding, reverse encoding differ in following areas (1) since forward encoding records the first datum as uncompressed while reverse encoding records last datum as uncompressed (2) forward encoding relates the present datum to the previous datum while the reverse encoding relates the present datum to the next datum. The encoded results of both forward and reverse encoding are same because absolute difference is not changed after interchanging the minuend and the subtrahend.

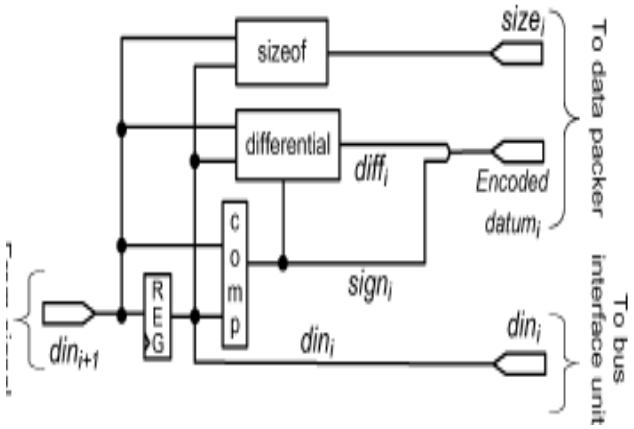

Fig. 4. Differential compressor hardware based on reverse encoding

Fig. 4 shows the differential compressor hardware based on reverse encoding. The present datum is represented as *dini* and next datum is represented as *dini+1* , *dini+1* is obtained by delaying the input data with a register *REG*.For each cycle three outputs are generated: 1) the absolute difference, which is obtained by subtracting the present datum from the next datum ; 2) the sign magnitude, which is obtained by comparing the two data, that results in identifying whether the result is positive or negative ; and 3) *sizei*) which indicates the number of meaningful bits of the difference. The reason for representing the difference as the absolute value and a signed magnitude is because the high-order bits of the absolute difference are usually zeros and can be discarded. The absolute difference and the sign magnitude are further packed together to form the final encoded datum . Finally, the data packer module stores the meaningful low-order bits of the encoded datum according to the size. In addition, to record the last datum in uncompressed format, we can read it from the register *REG* when the trace stops. Therefore, the register *REG* is implemented as the interface register of the tracer so that it can access via the bus.

B. Slice Compression

The basic principle of slice compression is that the higher parts between two adjacent branch/target address data are almost similar. Hence, by the elimination of the similar parts, the compression ratio can be raised. Table I shows an example of forward-/reverse-encoding slice compression for a set of instruction addresses. In the forward encoding, the first address is kept uncompressed. The succeeding address is encoded as its differential bits from the preceding address, .eg., 0x0300_6120 is encoded as 0x120 since the difference between 0x0300_6120 and 0x0300_6600 is 0x120. For reverse encoding, the reference direction is reversed; the last address is kept uncompressed while the previous ones are encoded based on the uncompressed last address.

Time	Original address	Forward encoding	Reverse encoding
t	0x0300_6600	**0x0300_6600**	0x600
t+1	0x0300_6120	0x120	0x20
t+2	0x0300_6130	0x30	0x300_6130
t+3	0x0080_4020	0x080_4020	0x020
t+4	0x0080_4400	0x400	**0x0080_4400**

Table I
Example of slice compression for forward/reverse encoding

The implementation of the reverse-encoding slice compression is similar to that of the differential compression, as shown in Fig. 4. The only difference is that the *sizeof*, *differential*, and *comp* modules now compare the bit difference of two data.

C. Dictionary-Based Compression

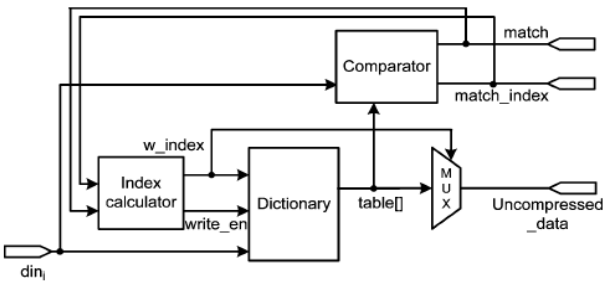

Fig.5. Dictionary compressor hardware based on forward encoding

In dictionary based compression the idea is to map the data to a table keeping frequently appeared data, and record the table index instead of the data to reduce size. Fig. 5 shows the hardware architecture of forward encoding dictionary compression. For each input datum, the comparator compares the datum with the data in the dictionary. If the datum is not in the table (Match=Miss), the datum is written into the table and also recorded in a trace. Otherwise, the index of the hit table entry (Match=Hit) is recorded instead of the datum. But the problem associated with this is that when wrapping around occurs, the uncompressed datum might be overwritten, resulting in data loss. Consider a example that we are sending a series of data's a1, a2, a3, a4, a5, a6.a1 to a3 has the same value as do a4 to a6. At first cycle, since the table is empty, input datum does not match with any table entry (known as a Miss). Therefore, it is inserted into the table and immediately recorded in an uncompressed format. At second and third cycle, a2 and a3 are the same; hence they are encoded as the indexes, respectively. At fourth clock cycle, a4 replaces a1 becausea4 is different from a1 (known as a Miss) and the table has only one entry, replaces. In this encoding algorithm, when a1 is overwritten, a2 and a3 cannot be decompressed.

Reverse encoding based dictionary compression can eliminate this problem. Concept of reverse encoding is that, the recording of a1 is delayed: It is only recorded until it is replaced with a4. As a result, the encoded data are placed before the uncompressed datum .With this order, the wrapping around does not cause data loss since the encoded data are lost before the related uncompressed data.

Fig.6. shows the dictionary-based compressor hardware based on reverse encoding. The input datum is compared with the data in the dictionary (din1) by the comparator. If there is a match, then the comparator sends the matched entry number (match index) to the data packer. If not, the input datum is inserted in the dictionary according to the index (w_index) calculated by the index calculator. At the same time, the replaced datum (replaced data) is output to the slice compressor for the further processing.

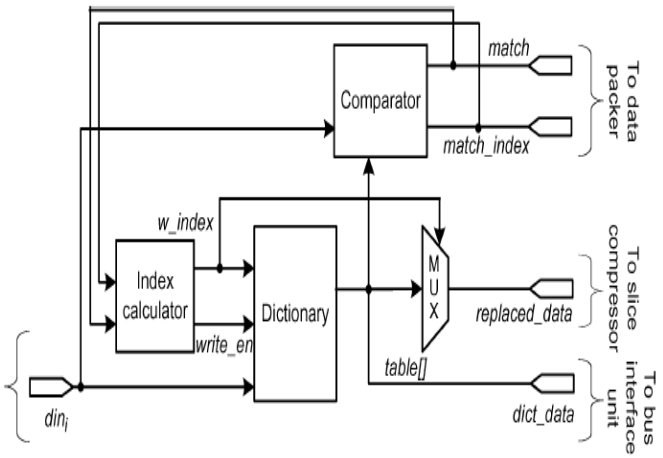

Fig.6. Dictionary compressor hardware based on reverse encoding

Main difference between forward and reverse encoding is that when data's are placed in dictionary they are not immediately recorded, it is recorded only when the data's are replaced.

V. DATA PACKER

Data packer receives the compressed data from the compression module, processes them, and writes them to the circular trace memory. It is responsible for: packet management and circular buffer management. For packet management, since the compressed data length and type are variable, every compressed data needs a header for interpretation. Therefore, this step generates a proper header and attaches it to each compressed datum. Fig.7 shows data packet example. Last packet contains the uncompressed date hence decompression starts from this. since the decompression works in a backward order, in order to interpret the content of a packet, the header is placed at the end of a packet so that the header can be read first before the trace data. It is obtained by reversing the bit order of a packet, since the sub headers should also be read before the compressed data.

VI. SIMULATION RESULTS

As an experimental study a communication architecture based on wishbone in SoC is designed consisting of four masters and four slaves .Master 1 is designed as processor and slaves as memories and the bus tracer based on reverse encoding is integrated to this architecture for tracing the processors program memory. Fig.8, Fig.9, Fig.10 shows results of master 1 communication to external memory

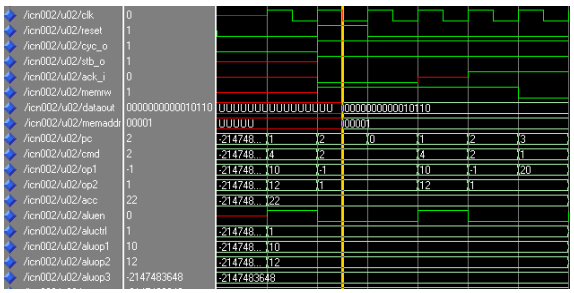

Fig.8.Simulation result of master 1 during write operation

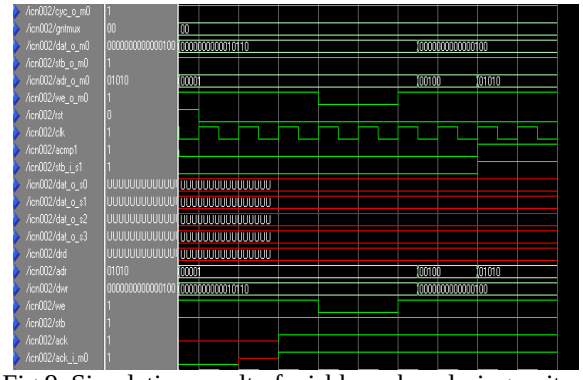

Fig.9. Simulation result of wishbone bus during write operation

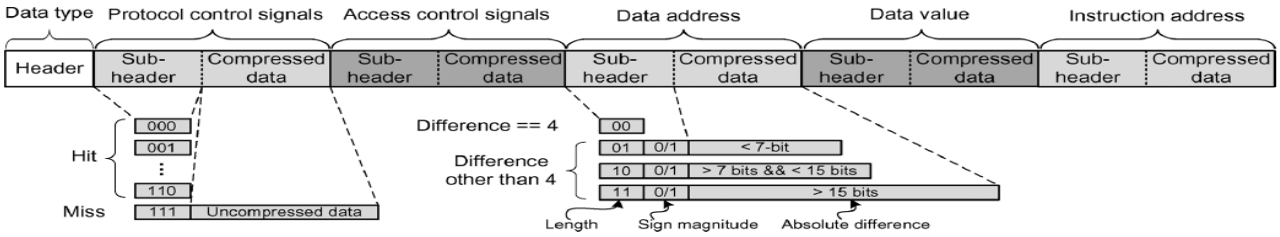

Fig.7. data packet format

The bus tracers traces both the communication, read and write to the external memory and compresses these using reverse encoding compression algorithms and stores them in circular trace memory. This is shown in Fig.11

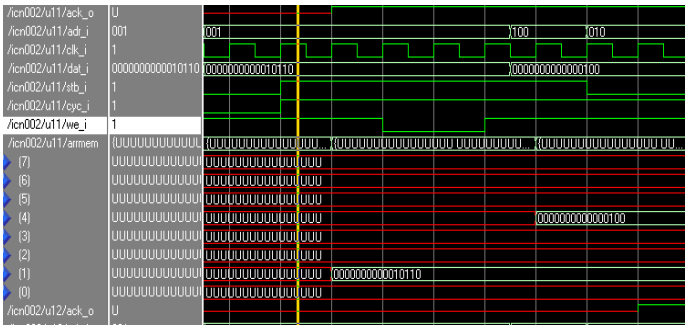

Fig.10. Simulation result external memory during write operation

The decompression is shown in Fig.12 which includes packet de-assemble and decompression of all these compression algorithms

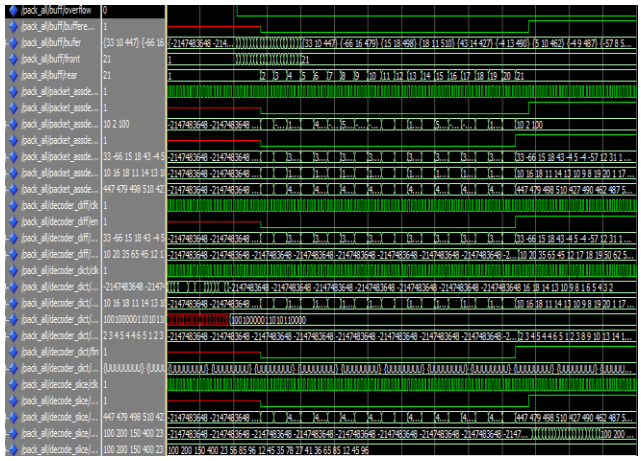

Fig.11.simulation results of reverse encoding compression algorithms

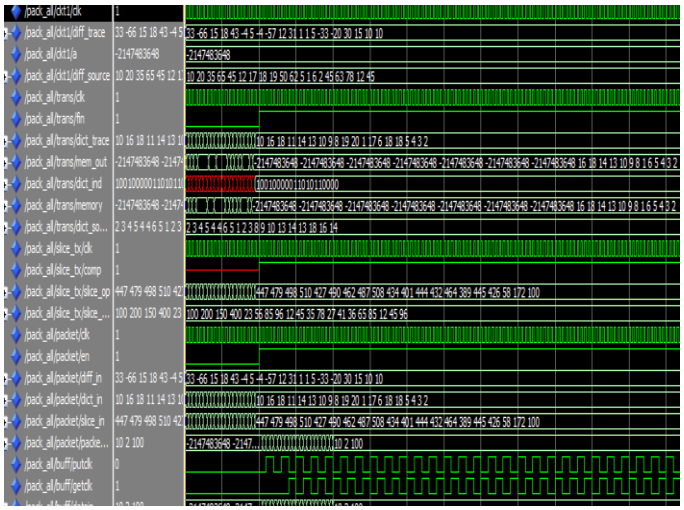

Fig.12.Simulation results of decompression of algorithms

The whole architecture is modeled using VHDL. Simulation was done using Modelsim XE III 6.4 and is synthesized using Xilinx ISE 9.2i

V. CONCLUSION

During design implementation stage and final chip testing bus signal tracing (bus tracer) can help designer to debug and analyze hardware-software design. Traditional tracers use forward-encoding compression algorithms to reduce the data size. However, such algorithms are not suitable for a backward trace with a circular buffer. So a reverse encoding based bus tracer is proposed that support both forward and backward trace. This tracer is integrated on wishbone bus consisting of four masters and four slaves. The result shows that this tracer can achieve good circular buffer utilization.

REFERENCES

[1] Fu-Ching Yang, "Reverse encoding based on chip bus tracer for effective circular buffer utilization", IEEE Trans. vlsi, vol. 18, no. 50, pp. 732–739, 2010.
[2] Ayas kantha swain," Design and verification of wishbone bus interface for system on chip integration", 2010 annual IEEE India conference
[3] J.-S. Yang and N. A. Touba, "Expanding trace buffer observation window for in-system silicon debug through selective capture," in Proc. IEEE 26th VLSI Test Symp., Apr. 27–May 1 2008, pp. 345–351.
[4] Y.-T. Lin, W.-C. Shiue, and I.-J. Huang, "A multi-resolution AHB bus tracer for read-time compression of forward/backward traces in a circular buffer," in Proc. DAC, Jul. 2008, pp. 862–865.
[5] E. Anis and N. Nicolici, "On using lossless compression of debug data in embedded logic analysis," in Proc. IEEE Int. Test Conf., Oct. 2007, pp. 1–10.
[6] ARM, Ltd., Cambridge, U.K., "Embedded trace macrocell architecture specification," 2002. [Online]. Available: http://infocenter.arm.com/
[7] Altera Corporation, San Jose, CA, "Signal Tap embedded logic analyzer mega function," 2001. [Online]. Available: http://www.altera.com/literature/ ds/dssignal.pdf