Configurable AMBA On-Chip Real-Time Signal Tracer

Chung-Fu Kao, Chi-Hung Lin, and Ing-Jer Huang

Dept. of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, 804, Taiwan
Tel : +886-7-5254337
Fax : +886-7-5254301
e-mail: ijhuang@cse.nsysu.edu.tw

Abstract – This paper purpose an embedded AMBA signal tracer for microprocessor-based SoC's. This tracer provides five trace resolution modes that can perform a cycle-accurate or a transaction-based trace collection in an unlimited time. Also this tracer is implemented in a Soft-IP style. It provides four parameters for tracing configuration. The experimental results show that the bus tracer can reach a good compression ratio of 96%.

I Introduction

One of the chip debugging and testing challenges is that the designer could only observe the external signals via chip I/O pins. The traditional methods such as scan chain takes long testing time, low operation speed, and more area overhead. The tracing method can help designer to observe and analyze the internal signals transaction behavior but the trace size are too large at real-time tracing. To address the bus real-time tracing issue, we propose an on-chip AMBA signal tracer that can reduce and compress trace data efficiently. The bus tracer consists of two modules. The first module, called monitor and tracing, traces bus signals which user wants to observe and reduce unnecessary signals. The second module, called trace reduction, uses three data compression methods to compress corresponding AMBA signals. The bus tracer is integrated with ARM EASY (Example AMBA SYstem) [1] environment to demonstrate that our approach can trace bus signals deeply and reduce trace size efficiently.

II. On-chip Bus Tracer Architecture

The block diagram of proposed on-chip bus tracer is shown in Fig. 1. The functions of two major blocks: *monitor & tracing* and *trace reduction* are described as follows.

A. Monitor and Tracing

A.1 Signal/Timing Abstraction

The AMBA signal tracing can be divided into two abstract dimensions: signal abstraction and timing abstraction. The abstraction here means the granularity of signal and timing observation. According to different level of abstraction, we define five tracing modes of combinations of these levels, as shown in Fig. 2. During bus signals tracing process, user can select the tracing mode according to the granularity of signal and timing observation. For example, the mode 4 is combined signal abstraction at selected level with timing abstraction at transaction level. The selected level means user can decide

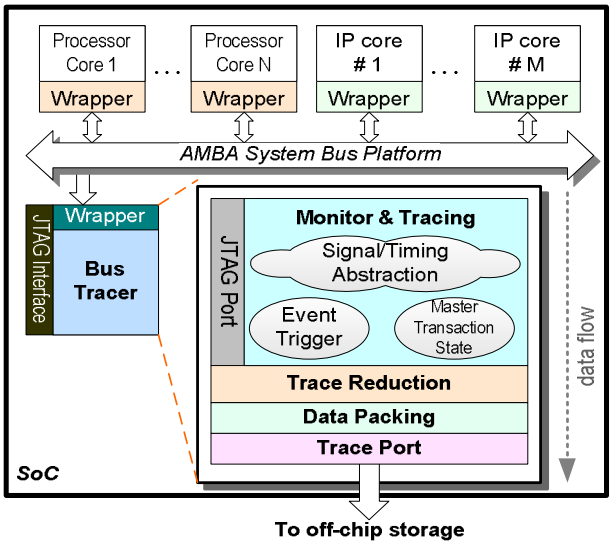

Fig. 1. The block diagram of AMBA signals tracer

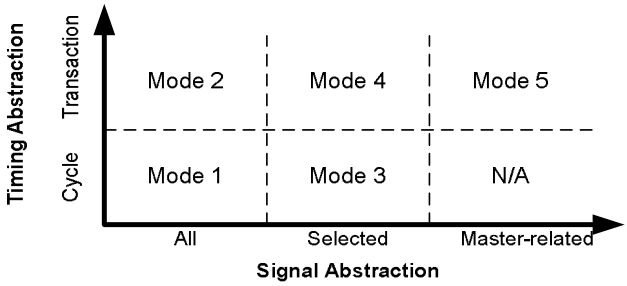

Fig. 2. Tracing modes of combinations of different signal and timing abstraction levels

which bus signals to be traced, and the transaction level means the bus tracer traces the bus signals only when signals transactions occurred. The advantage of signal and timing abstraction is that user can choose lower level, for example, mode 1 or mode 2, to observe detail signals transition. On the other hand, user chooses higher level when he wants to observe only bus transaction behavior. The higher abstraction level can reduce more unnecessary signals to be traced.

A.2 Event Trigger

The event trigger module (ETM) provides two parameters. User setups the parameters at host and these parameters been transferred to event trigger module via JTAG interface.

- *Bus tracing mode parameter*. User can select the bus tracing mode which described at section A.1 before bus tracing process.
- *Watchpoint parameter for enable bus tracing*. The ETM

provides two watchpoint registers and supports logic combination of these two watchpoint conditions. For example, user can define that only when [(0x0030) appears on the address bus] AND [transfer direction is 'write'] then starts bus tracing process.

B. *Trace Reduction*

We use three compression methods for three kinds of signals to obtain higher compression ratio. For address bus, we omit the sequential addresses and only record the non-sequential addresses. Since the signal variations on data bus are not regular, we only use the differential approach based on subtraction to reduce the data bus trace size and the hardware cost of subtraction is small but the compression ratio is low. The bus control signals such as read/write, width of the transfer, and transfer size, *etc.*, don't change the value during a complete bus transfer. Therefore, we can use few bits to encode the combinations of these control signals, and record the encoded value instead of record all control signals value.

III. Hardware Implementation Verification

A. *Hardware Implementation*

We use UMC 0.18um technology and Artisan cell library to synthesize the bus tracer, and the synthesis results are shown in Table 1.

Table I
The synthesis results of AMBA signal tracer

Critical Path (unit: ns)	4.69
Gate Count (unit: gate counts)	38775

B. *RTL, FPGA and Chip Verification*

In RTL-level, our verification environment is the EASY platform. We integrate the bus tracer into EASY AHB, and save the tracing results to a file (A). We plug a bus monitor to dump all bus signals cycle-by-cycle and save the results to a file (B). Then we decompress the tracing file (A) to original bus signals and save to a file (C). Finally, we compare two files, B and C, to verify the tracing and compression functions of our bus tracer. In FPGA, we download the bus tracer into ARM Versatile prototyping board to demonstrate our design. The verification environment shows in Fig. 3 is a 3D graphics acceleration SoC. We use a logic analyzer to capture the bus trace data and decompress the trace data at host, then compare the results with original bus signals.

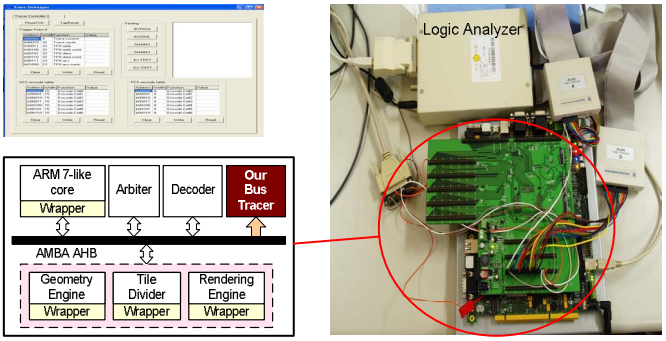

Fig. 3. FPGA prototyping: 3D graphics acceleration application

The AMBA signal tracer been integrated into a 3D application SoC which consists of a 32-bit CPU, a 3D graphics acceleration engine, an AMBA platform, and our trace module. This SoC has been taped out in this August.

IV. Experimental Results

In AMBA AHB environment, we define 12 kinds of signals to be traced and there are total 88 bits to be traced. We run 10,000 cycles for each of six programs and the experimental results are shown in Table 2. The original trace size of all signals at cycle-by-cycle is 880,000bits (88 x 10000). The geometric mean of the compression ratio can achieve 96%.

V. Conclusion

Bus signal tracing can help designer to debug and analysis hardware software design during design implementation stage and final chip testing. Not only the address bus or data bus signals tracing, the proposed bus tracer can trace all bus signals and compress the trace size efficiently. The bus tracer can perform a cycle-accurate or successive trace collection in transaction level abstraction and can be used in various bus systems with proper bus interface. The bus tracer has integrated into ARM EASY system and the experimental results show that the bus tracer can reach compression ratio of 96%.

References

[1] ARM, Example AMBA SYstem User Guide ARM DUI 0092C, http://www.arm.com/pdfs/DUI0092C.zip

Table II
The data trace compression ratio in different tracing modes

	Original trace size (88bits/cycle)	Compression Ratio After Signal/Timing Abstraction and Trace Reduction				
		@ mode 1	@ mode 2	@ mode 3	@ mode 4	@ mode 5
Perpetual Calendar	880000	77.58%*	83.00%	90.05%	94.67%	96.99%
Fibonacci Sequence	880000	78.32%*	80.05%*	87.91%	90.45%	94.73%
G. C. D.	880000	83.22%	83.45%	89.14%	92.76%	95.68%
Towers of Hanoi	880000	78.64%*	80.61%*	85.99%	88.58%	92.56%
Knight Problem	880000	78.85%*	80.51%*	96.39%	97.16%	98.32%
Quick Sort	880000	78.27%*	79.99%*	99.64%	99.71%	99.80%
Geometric mean	-	79%	81.26%	91.39%	93.81%	96.32%

(*): the star sign means the trace buffer (256 bits) overflow