

Implementation of Multi-Resolution On-Chip AHB Bus Tracer with Real Time Lossless Compression

Dr. K. Babulu¹, P. Anvesh²

Professor, Department of E.C.E, JNTU, Kakinada, India¹

M.Tech Student, Department of E.C.E, JNTU, Kakinada, India²

Abstract: An On-Chip AHB Bus Tracer is a significant infrastructure that is needed to monitor the on chip-bus signals, which is vital for debugging and performance analysis and also optimizing the SoC (System On Chip). So in this paper we implement the on-chip AHB bus tracer, that traces with different resolutions, i.e. with different signal and abstraction levels depending on the need to match with specific debug /analysis needs. In addition it allows the user to switch the resolution on-the-fly. Subsequently compression of the trace without any loss of the actual trace which when reconstructed at the analyser will remain the same. This Bus Tracer adopts three trace compression techniques to achieve high compression ratio. The On-Chip AHB bus tracer with Real-Time Compression and Dynamic Multi-Resolution was designed successfully; the RTL simulations were performed successfully along with successful synthesis using Xilinx ISE.

Keywords: AMBA, AHB Bus Tracer, Real Time Compression, Dynamic Multi Resolution.

I. INTRODUCTION

As more and more IP cores are integrated into a System On-Chip (SoC) design, the communication flow between IP cores has increased drastically and the efficiency of the on-chip bus has become a dominant factor for the performance of the system. ADVANCE MICROCONTROLLER BUS ARCHITECTURE (AMBA) usage is expanding in both FPGA and Military applications. In the present technology, high performance and speed are required which is convincingly met by AMBA-AHB.

The AMBA-AHB defines a point to point interface between two communicating entities such as IP cores and bus interface modules. One entity acts as a master of the AMBA-AHB instance, and the other as a slave. Only the master can present commands and is the controlling entity. The slave responds to commands presented to it, either by accepting data from the master, or presenting data to the master.

So monitoring the on-chip bus signals is very essential to make the SoC perform efficiently. But as these signals are deeply embedded in the SoC which makes the user difficult to observe. Hence as a solution these signals are needed to be abstracted from the bus, which we call as tracing of signals, stored on some on-chip storage and will be off loaded to the analyser for analysis. But unfortunately, the bus trace size grows at a rapid rate. This is because the rate of signals from different IP cores on a SoC is very high. As we cannot increase the on chip memory at this rate so we will compress the trace accordingly without any loss of trace as that when re-constructed at the analyser the trace remains the same. And along with this different abstraction levels are adopted depending on the designers needs some require all signals at cycle level, while others require only transactions. Thus, there must be a way for capturing traces at different abstraction levels based on the debugging and analysis needs. If the given Trace memory is fixed then the user can trade off between the trace granularities and trace length. This feature provides a more flexible tracing.

II. RELATED WORK

There are hard ware approaches to compress the trace, which can be divided in lossy and lossless categories. The former one i.e. lossy compression technique is used when the need is that whether any problem have occurred or not and so the original trace cannot be reconstructed. Anis and Nicolici used the Multiple Input Signature Register (MISR) to achieve this compression.

Then the results are compared with the golden patterns so as to find the erroneous part. This process has to be repeated until the error is found, hence this method is preferable only to the simple SoC's. Hence we go for lossless techniques for complex SoC's.

Some appropriate compressing methods have been available for different types and parts of bus signals. Branch/target filtering is one common technique for program address compression, which has been used in TRICOR, ARM Embedded Trace Macro cell. And for data addresses and value compression, differential and dictionary based compression techniques are popular. Hopkins and Mc-Donald-Maier showed that the differential method can reduce the data address and data value by 40% and 14%. Several companies have recognised the need to support On-chip tracing. Gaisler Research provides an AHB Trace Buffer (AHBTRACE) in GRLIB IP Library. ARM provides AMBA AHB Trace Macro cell (HTM) to trace the address and data signals of AHB bus. First Silicon Solutions (FS2) develops AMBA Navigator for monitoring the signals on the AHB.

Another approach for the program address compression is slicing. In this technique the binary data is partitioned into several slices and then records all the initial slices. Afterwards the slices that are different from the initial slices to the corresponding slices of the previous ones are only recorded. For control signals, ARM HTM encodes them with the slice compression approach the control signal is recorded only when the value changes.

Compressing all signals at the cycle-acute-level does not always meet the debugging needs. The transaction-level debugging becomes increasingly important as SoC's become more complex which helps designers focus on the functional behaviour instead of interpreting complex signals. Tabbara and Hashmi propose the transaction level SoC modelling and debugging methods. The proposed transactors, attaching to the on-chip bus, recognize/monitor signals and abstract the signals into transactions. The transactions, bridging the gap between algorithm-level and the signal-level, enable easy design exploration/debugging/monitoring. Encouraged by the interrelated works, our bus tracer combines abstraction and compression techniques in more belligerent way.

III. OVER VIEW OF AMBA BUS TRACER ARCHITECTURE

This section provides an overview of the proposed AMBA AHB Bus Tracer (post) architecture, which is shown in Figure 1. The bus tracer consists of Four major functional modules (1) Event Generation (2) Abstraction (3) Compression (4) Packing. The signal Abstraction module traces the corresponding AHB signals at proper time according to user configuration. The trace compression module compresses the trace data in accordance with signal characteristics. Finally, in the data packing module, the trace data is arranged compactly for output to the internal on-chip trace memory or external off-chip storage. When the on-chip trace memory is full, it sends an interrupt to the microprocessor then this processor reads the data from the trace memory and transfers the trace data to off-chip storage through AMBA.

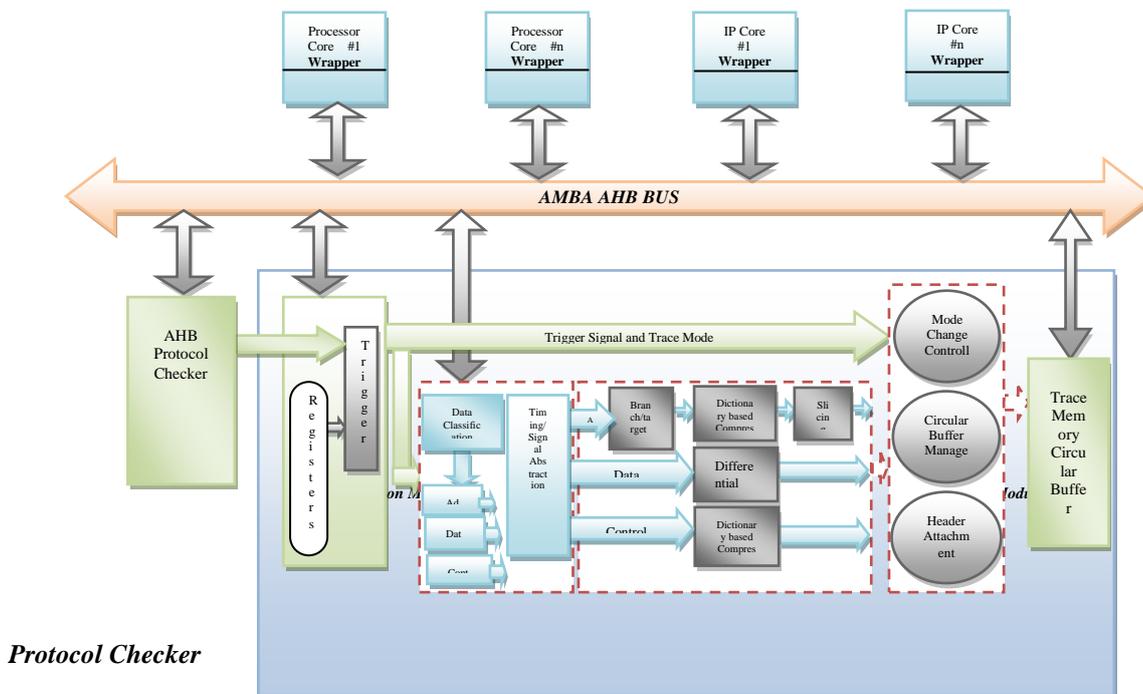


Fig.1. The Block Diagram of AMBA AHB Bus Tracer architecture.

HP Checker is a rule-based protocol checker, thus how to set up a set of well-defined rules is very important. We reference Synopsys verification intellectual property (VIP) to establish 67 rules. Besides, according to our design experiences, we add 6 new rules to increase our error finding ability. In conclusion, our protocol checker has total 73 rules, including 31 master-related rules, 16 slave-related rules, 11 reset-related rules, and 15 bus components-related rules. Bus components include arbiter and decoder.

Protocol Checker is the main core of HP Checker, the inputs are all AHB bus signals, and the outputs are 73-bit ERROR signals and corresponding master and slave IDs. Every rule has its own corresponded bit because every cycle maybe occur more than one error. If the i th bit of ERROR is set, which indicates current bus signals violate i^{th} rule. The Master/Slave ID is formed by the HMASTER signal. If an error occurs, the HP Checker will output the corresponded master ID number or slave ID number to indicate which master or slave violates the AHB protocol.

A. Event Generation

The Event Generation Module decides the starting and stopping of a trace and its trace mode. The module has configurable event registers which specify the triggering events on the bus and a corresponding matching circuit to compare the bus activity with the events specified in the event registers.

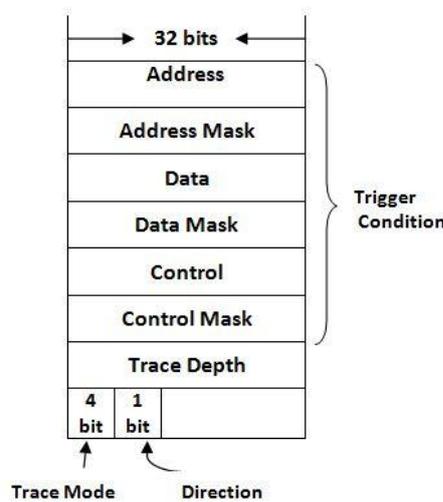


Fig.2. Event Register.

Figure. 2 is the format of an event register. It contains four parameters: the trigger conditions, the trace mode, the trace direction, and the trace depth. The trigger conditions can be any combination of the address value, the data value, and the control signal values. Each of the value has a mask field for enabling partial match. For each trigger condition, designers can assign a desired trace mode, e.g., Mode FC, Mode FT, etc., which allows the trace mode to be dynamically switched between events. The Event Generation Module monitors the AHB bus for target conditions. There are two types of target conditions: breakpoints and AHB protocol checker. The breakpoints are the target values of AHB bus signals. On the other hand, the AHB protocol checker (HP Checker) is a hardware implementation of the non synthesizable AHB protocol rules described in Synopsys AHB Verification IP (VIP) and Gaisler Research’s GRLIB IP library.

B. Abstraction Module

Title must be in 24 pt Regular font. Author name must be in 11 pt Regular font. Author affiliation must be in 10 pt Italic. Email address must be in 9 pt Courier Regular font. The abstraction here means the granularity of signal and timing observation. At signal dimension, it has three abstraction levels which are all signals, bus state, and master operation. The all signals level means all bus signals according to the bus transfer operation will be traced; bus state level means that using master transition states to represent the transfer status and does not record related control signals. Master operation level means that bus tracer traces signals only related to the current bus master transfer operation such as address/data bus signals, and omits the bus transition states.

We abstract the AMBA AHB bus master transition states from its bus transfer behaviour as shown in Figure 3. Each state represents the combination of corresponding control signals under current bus transfer. For example, the state number 2

(Normal) means that the bus master is performing a transfer and the control signals of HREADY must be 'OK' and HTRANS must be 'Non-SEQ' or 'SEQ'. When the user chooses the trace mode BC or mode BT (bus state abstraction level), the bus tracer records the bus states instead of control signals, and can save the trace size.

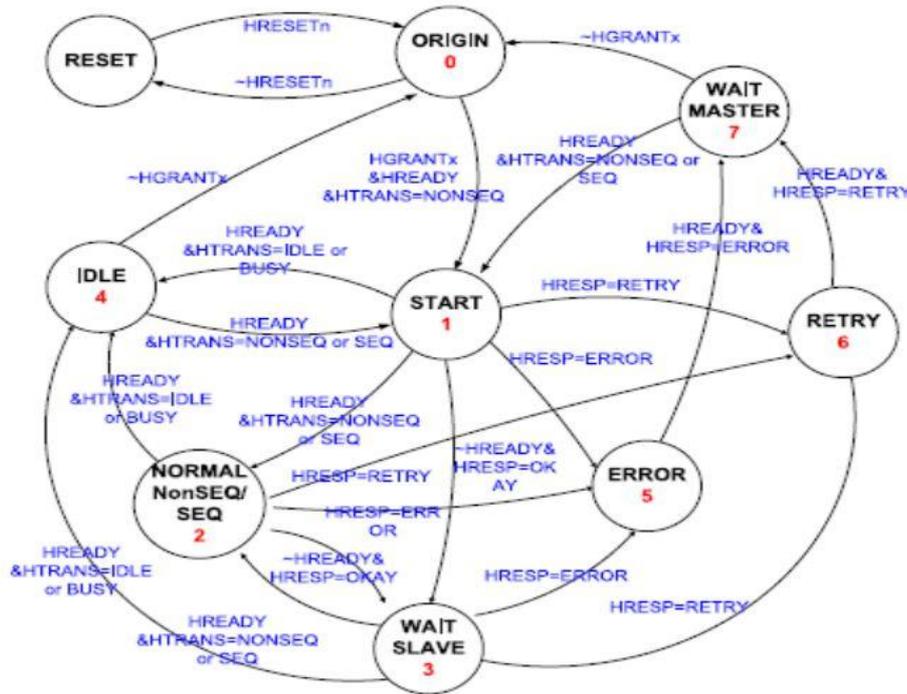


Fig.3. AMBA AHB Bus Transition States.

At timing dimension, it has two abstraction levels which are cycle level and transaction level. The bus tracer records bus signals cycle-by-cycle in cycle level; in transaction level, bus tracer records the trace only when signals transactions occurred. In other words, if a signal remains the same value during a transfer, the bus tracer does not record the signals value besides at the first time. For example, the transfer direction such as READ or WRITE would not be changed during a bus transfer, and the bus tracer records the transfer direction only at the first cycle of current bus transfer in transaction level. According to different levels of abstraction, we define five trace modes of combinations of these levels.

At Mode FC, the tracer traces all bus signals cycle-by-cycle so that designers can observe the most detailed bus activities. This mode is very useful to diagnose the cause of error by looking at the detail signals. However, since the traced data size of this mode is huge, the trace depth is the shortest among the five modes. Fortunately, it is acceptable since designers using the cycle-level mode trace only focus on a short critical period.

In Mode FT, the tracer traces all signals only when their values are altered. In other words, this mode traces the untimed data transaction on the bus. Comparing to Mode FC, the timing granularity is abstracted. It is useful when designers want to scan the behaviours of all signals as a substitute of looking at them cycle-by-cycle. Another advantage of this mode is that the space can be saved without losing significant information. Thus, the trace depth increases.

In Mode BC, the tracer uses the BSM(Bus State Machine), for instance NORMAL, IDLE, ERROR, and so on, to represent bus transfer activities in cycle accurate level. Comparing to Mode FC, even though this mode still captures the signals cycle-by-cycle, the signal granularity is abstracted. Thus, designers can observe the bus handshaking states without analyzing the detail signals. The benefit is that designers can still observe bus states cycle-by-cycle to analyze the system performance.

In Mode BT, the tracer uses bus state to represent bus transfer activities in transaction level. The traced data is abstracted in both timing level and signal level; it is a combination of Mode BC and Mode BT. In this mode, designers can easily understand the bus transactions without analyzing the signals at cycle level.

At Mode MT, the tracer only records the master behaviours, such as read, write, or burst transfer. It is the highest abstraction level. This feature is very suitable for analyzing the masters' transactions. The major difference compared with Mode BT is that this mode does not record the transfer handshaking activities and does not capture signals when the bus state is IDLE, WAIT, and BUSY. Thus, designers can focus on only the masters' transactions. Please note that there is no mode supporting master

operation trace at cycle level, since the intension of observing master behaviours is to realize the whole picture. Tracing master behaviours at cycle level is meaningless and can be replaced with Mode BC.

C. Compression Module

Although the higher abstraction modes omit the unnecessary bus signals to be traced, the trace size may be still large. To achieve the effect of high compression ratio, three compression methods for corresponding signals, which are address bus, data bus, and transfer control signals, are proposed to obtain higher bus trace compression ratio.

1) Address Bus Trace Compression: We use two phases approach to compress address data. In the first phase, we omit the sequential addresses and only record the non-sequential addresses. In the second phase, we use a dictionary table to store the recently used of non sequential addresses, and record the index value instead of original address value, which is shown in Fig.6.

Phase 1: Branch/Target Filtering Approach:

A software program, when compiled to the assembly or binary code, consists of a number of basic blocks. A basic block consists of a sequence of linearly executed instructions. The first and last instruction in a basic block is called a target and branch instruction respectively. Since the instructions within a basic block are executed as a group, it suffices to record only the addresses of the target and branch instructions when tracing the (program) address bus signals.

Phase 2: Dictionary Approach:

In this phase, branch and target addresses are stored in a CAM-based dictionary table sequentially. If the current address can be found in the table (dictionary hit), the corresponding index value would be recorded. On the other hand, if the current address cannot be found in the table (dictionary miss), the full address value would be recorded and this address would be stored in the table. When the table is full, the next 'miss address' would be stored in the first entry of the table and replace the original address value.

For the diagram shown in the figure 4 each input datum (din_i), the comparator compares the datum with the data in the dictionary. If the datum is not in the table (match=miss), the datum (uncompressed data) is written into the table and also recorded in a trace. Otherwise (match=hit), the index (match index) of the hit table entry is recorded instead of the datum. The hit index can be further compressed. As we know, a basic block is composed by a target address and a branch address, and the branch instruction address appears right after target instruction address. By the fact that basic blocks repeat frequently, if the target address is hit at the table entry ($i+1$), the branch address will hit at the table entry, since these entries are stored in the dictionary in a FIFO way. Therefore, instead of recording the hit index of that branch address, we create a special header, called the continuous hit, to represent that branch address if it meets this condition.

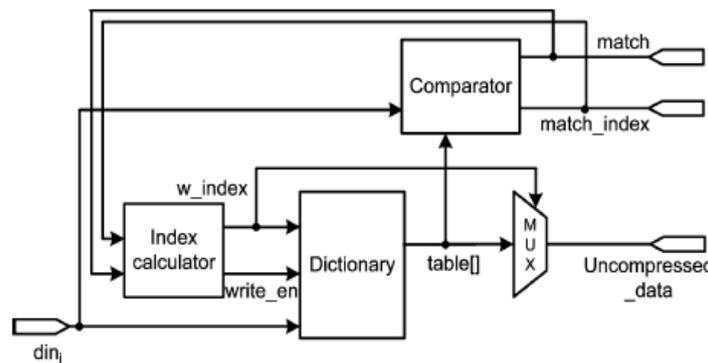


Fig.4. Block Diagram of Dictionary based Compression.

Phase 3: Slicing

The miss address can also be compressed with the Slicing approach. Because of the spatial locality, the basic blocks are often near each other, which mean the high-order bits of branch/target addresses nearly have no change. Therefore, the concept of the Slicing is to reduce the data size by recording only the different digits of two consecutive miss addresses. Figure 5 shows the hardware architecture. It has the register REG storing the previous data (din_{i-1}). The slice comparator compares the slices of the current datum (din_i) and the previous datum and produces the identical slice number ($size_i$). This information is forwarded to the packing module to generate the proper header. This is the packet format 3 in Figure 6.

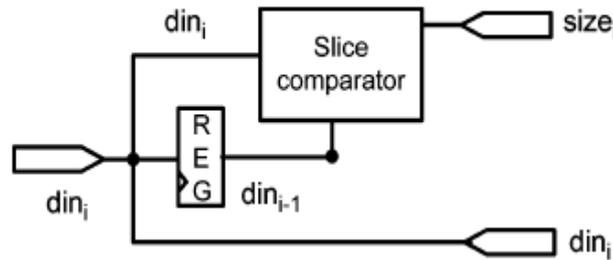


Fig.5. Block Diagram of Slicing Circuit.

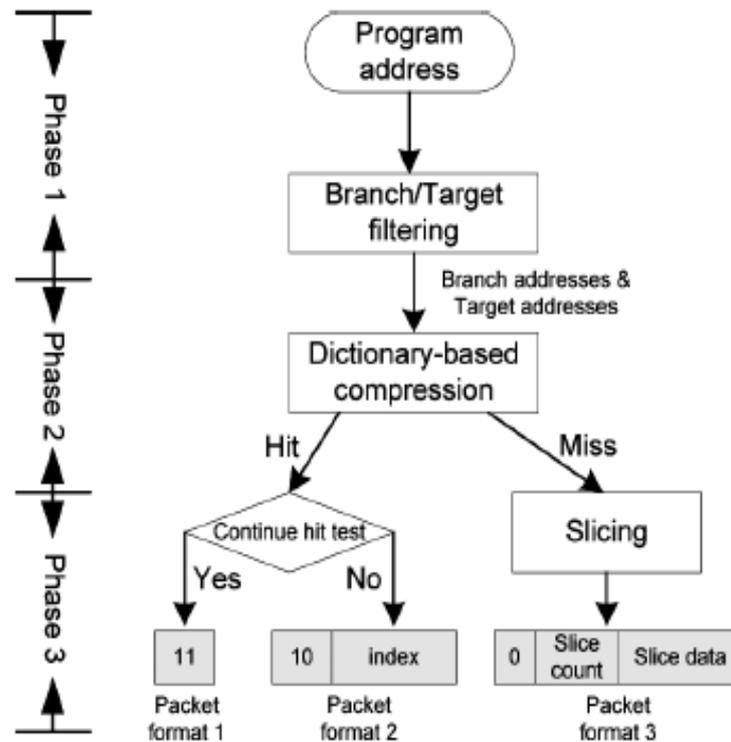


Fig.6. Program Address Compression Flow.

2) *Data Bus Trace Compression:* Since the signal variations on the data bus 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 is small but the compression ratio is low (about 20% -30%).

Figure 7 shows hardware compressor. The register REG saves the current datum din_i and outputs the previous datum din_{i-1} . By comparing the current datum with the previous data value, the three modules comp, differential, and sizeof output the encoded results. The comp module computes the sign bit (signed_bit) of the difference value. The differential module calculates the absolute difference value (value). Since the absolute difference between two data value may be small, we can neglect the leading zeros and use fewer digits to record it. Therefore, the sizeof module calculates the nonzero digit number ($size_i$) of the difference. Finally, the encoded datum is sent to the packing module along with $size_i$.

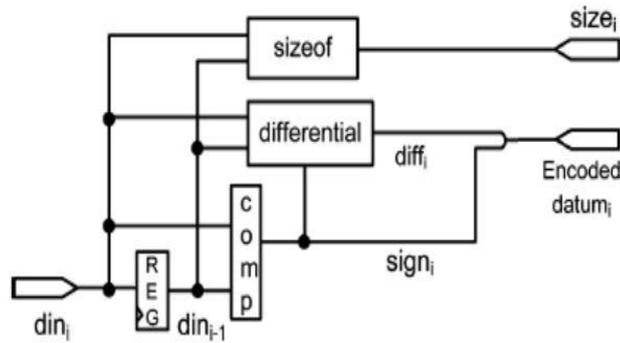


Fig.7. Block Diagram Of differential Compression.

3) *Control Signal Trace Compression*: A level-3 heading must be indented, in Italic and numbered with an Arabic numeral followed by a right parenthesis. The level-3 heading must end with a colon. The body of the level-3 section immediately follows the level-3 heading in the same paragraph.

For example, this paragraph begins with a level-3 heading. When a bus master is performing a bus transfer, the control signals, such as read/write, width of the transfer, transfer size, etc. don't change their 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.

For example, in an AMBA platform, the control signals, e.g. HWRITE, HBURST [2:0], HSIZE[2:0], HPROT[3:0], and HMASTER[3:0] don't change their value during a bus transfer. Therefore the original trace size of these control signals is 15bits. If we use 3bits to encode the combination of these control signals, we can reduce trace size by about $(1 - 3/15) \times 100\% = 80\%$. In an AMBA system, the combinations of control signals are more than 8 (23), the control signals trace compression module provides a CAM based dictionary table.

The concept is similar to compress the address bus (phase 2). If the current combination of control signals is appeared in the table, the index value (3- bit) would be recorded. On the other hand, we will record the 15-bits control signals when the table miss occurred.

D. Packing Module

The Packing Module is the last phase. It receives the compressed data from the compression module, processes them, and writes them to the trace memory. It is responsible for three jobs: packet management, circular buffer management, and mode change control. 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.

In this paper, we call a compressed data with a header as a packet. Since the header generation takes time, to avoid long cycle time, the header generation is implemented in one pipeline stage. For circular buffer management, it manages the accesses to the trace memory. Since the size of a packet is variable but the data width of the trace memory is fixed, this module collects the trace data in a first-input, first-output (FIFO) buffer and outputs them to the trace memory until the data size in the FIFO buffer is equal/larger than the data width. If the tracing stops and the data size in the FIFO buffer is smaller than the data width, one additional cycle is required to output the remaining data to the trace memory.

IV. EXPERIMENTAL RESULTS

This section deals with the simulation and synthesis results of the implemented On-Chip AHB Bus Tracer with Real-Time Compression and Multi-resolution. Here Modelsim tool is used in order to simulate the design and checks the functionality of the design. Once the functional verification is done, the design will be taken to the Xilinx tool for Synthesis process and the net-list generation.

A. Simulation Results

1) MODE FC:

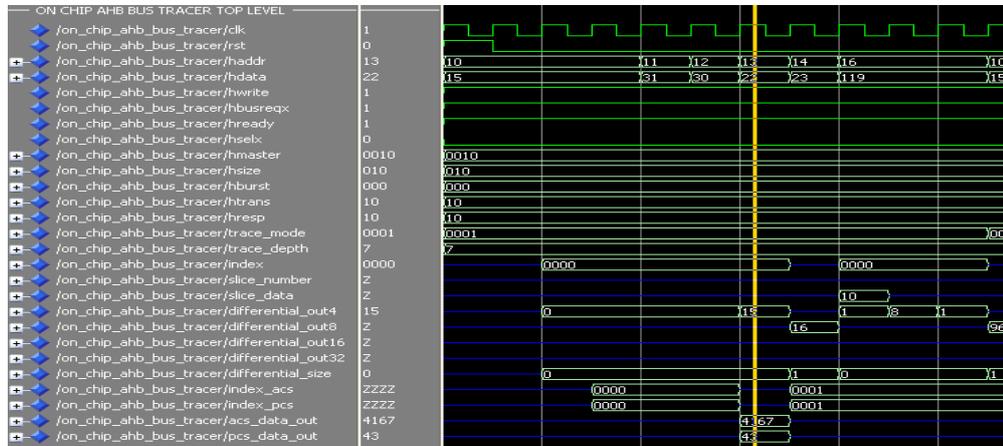


Fig.8. Simulation results of Mode FC.

2) MODE FT:

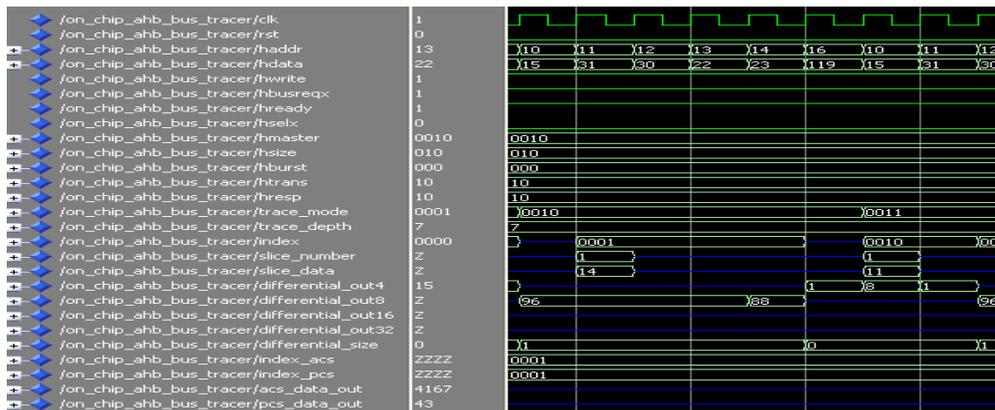
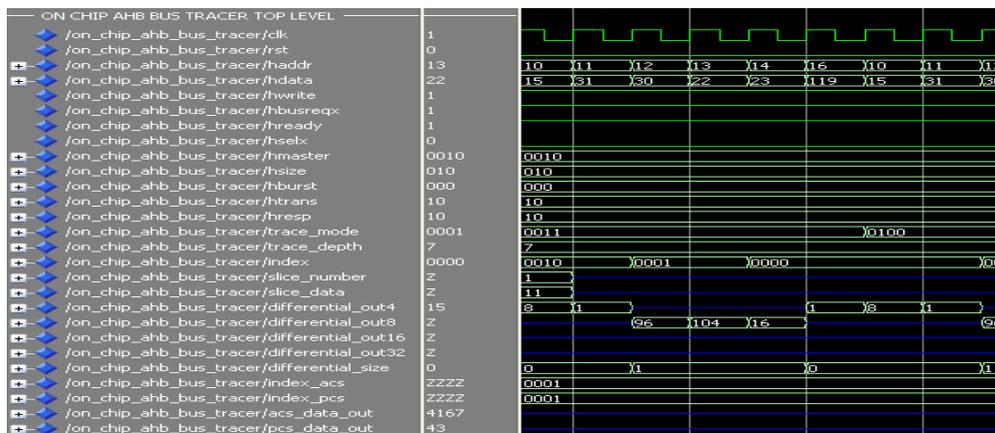


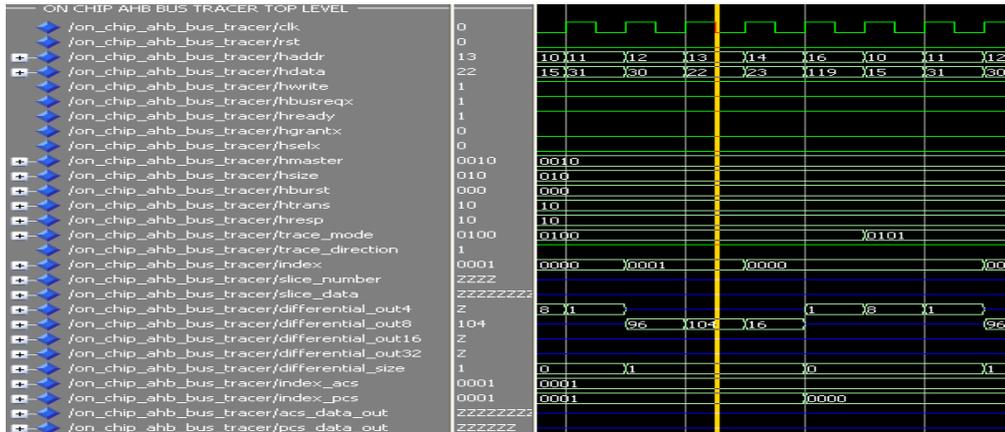
Fig.9. Simulation results of Mode FT.

3) MODE BC:



4) *MODE BT:*

Fig.10. Simulation results of Mode FT.



5) *MODE MT:*

Fig.11. Simulation results of Mode BT.

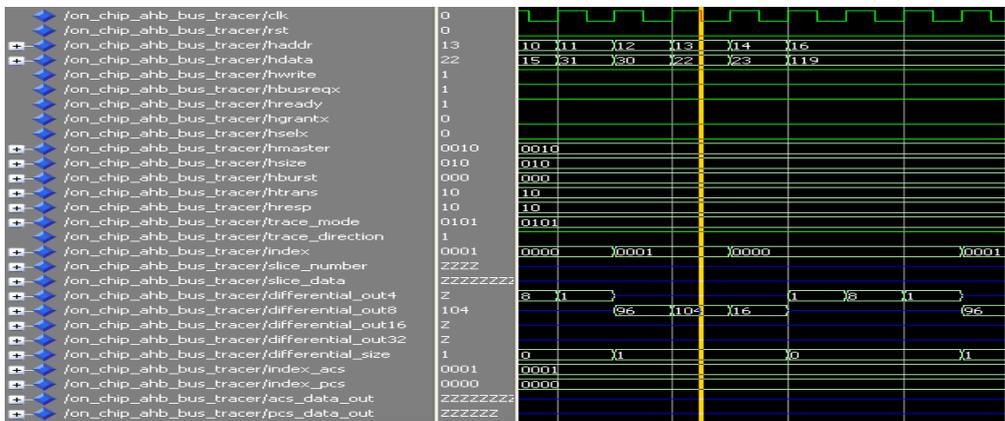


Fig.12. Simulation results of Mode MT.

B. *Synthesis Results*

Once the functional verification is done, the RTL model as shown in Figure 14 is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level net-list mapped to a specific technology library. This AES algorithm design can be implemented on FPGA (Field Programmable Gate Array) family of Virtex-5. Here in this Virtex-5 family, many different devices were available in the Xilinx ISE tool. In order to implement this AES design the device named as “XC2VLX330T” has been chosen and the package as “FF1738” with the device speed as “-2”.

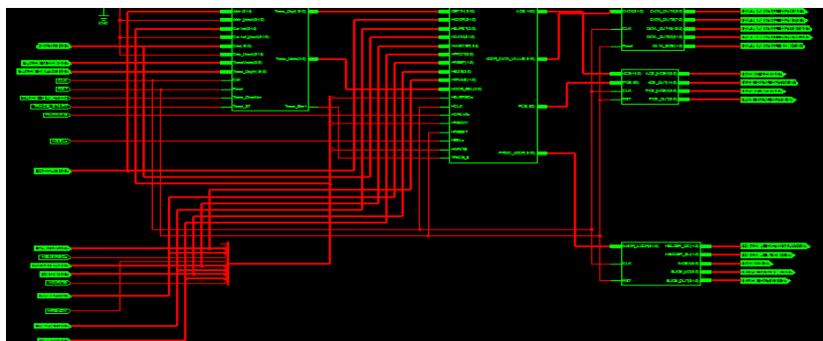


Fig.13. Internal view of RTL Schematic View of On-Chip AHB Bus Tracer.

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	2,144	207,360	1%	
Number used as Flip Flops	2,144			
Number of Slice LUTs	3,442	207,360	1%	
Number used as logic	3,419	207,360	1%	
Number using O6 output only	2,309			
Number using O5 output only	690			
Number using O5 and O6	420			
Number used as exclusive route-thru	23			
Number of route-thrus	713	414,720	1%	
Number using O6 output only	713			
Slice Logic Distribution				
Number of occupied Slices	1,266	51,840	2%	
Number of LUT Flip Flop pairs used	4,121			
Number with an unused Flip Flop	1,977	4,121	47%	
Number with an unused LUT	679	4,121	16%	
Number of fully used LUT-FF pairs	1,465	4,121	35%	
Number of unique control sets	74			
IO Utilization				
Number of bonded IOBs	259	960	26%	

Fig.14. Design summary of On-Chip AHB Bus Tracer.

C. *Timing Summery*

Speed Grade: -2

Minimum period: 5.037ns (Maximum Frequency: 198.515MHz)

Minimum input arrival time before clock: 4.864ns

Maximum output required time after clock: 2.934ns

Maximum combinational path delay: 3.306ns

In timing summery, details regarding time period and frequency is shown are approximate while synthesize. After place and routing is over, we get the exact timing summery. Hence the maximum operating frequency of this synthesized design is given as **198.515MHz** and the minimum period as **5.037ns**. **OFFSET IN** is the minimum input arrival time before clock and **OFFSET OUT** is maximum output required time after clock.

V. **FUTURE SCOPE**

As future work, This work can be improved by implementing it with Advanced RISC Machines (ARM) Processors. This design can also be used in all System-On-A-Chip SoC applications where debugging and performance analysis is difficult. As for the circuit speed, the bus tracer is capable of running at 198.515 MHz, which is sufficient for most SoC's with a synthesis approach under Xilinx Synthesis technology. If a faster clock speed is necessary, our bus tracer could be easily partitioned into more pipeline stages due to its streamlined compression/packing processing flow.

VI. **CONCLUSION**

The On-chip AHB bus tracer with Real-time Compression and Dynamic Multi-Resolution was designed successfully and the coding was done in VHDL. The RTL simulations were performed using Modelsim from Mentor Graphics. The synthesis was done using Xilinx ISE. The On-chip AHB bus tracer with Real-time Compression and Dynamic Multi-Resolution works at a frequency of **198.515MHz**. The **Designed Tracer** works properly for all the Modes such as Mode FC, Mode FT, Mode BC, Mode BT, Mode MT . Tracer design is verified for all test cases. The specification of the implemented bus tracer has been implemented, RTL, FPGA,. The bus tracer costs only about 2144 slice registers which uses 2144 flip-flops , which is relatively small in a typical SoC. The reason is that this paper optimizes the ping-pong architecture by sharing most of the data path instead of duplicating all the hardware components.

REFERENCES

- [1] ARM Ltd., San Jose, CA, "AMBA Specification (REV 2.0) ARM IHI0011A," 1999.
- [2] E. Rotenberg, S. Bennett, and J. E. Smith, "A trace cache micro architecture and evaluation," IEEE Trans. Comput., vol. 48, no. 1, pp. 111–120, Feb. 1999.
- [3] B. Tabara and K. Hashmi, "Transaction-level modelling and debug of SoC's," presented at the IP SoC Conf., France, 2004.
- [4] A. B. T. Hopkins and K. D. McDonald-Maier, "Debug support strategy for systems-on-chips with multiple processor cores," IEEE Trans. Comput., vol. 55, no. 1, pp. 174–184, Feb. 2006.
- [5] E. Anis and N. Nicolici, "Low cost debug architecture using lossy compression for silicon debug," in Proc. IEEE Des., Autom. Test Eur. Conf., Apr. 16–20, 2007, pp. 1–6.
- [6] B. Vermeulen, K. Goosen, R. van Steeden, and M. Bennebroek, "Communication- centric SoC debug using transactions," in Proc. 12th IEEE Eur. Test Symp., May 20–24, 2007, pp. 69–76.
- [7] Y.-T. Lin, C.-C. Wang, and I.-J. Huang, "AMBA AHB bus protocol checker with efficient debugging mechanism," in Proc. IEEE Int. Symp. Circuits Syst., Seattle, WA, May 18–21, 2008, pp. 928–931.
- [8] 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. Des. Autom. Conf. (DAC), Jul. 2008, pp. 862–865.
- [9] Infineon Technologies, Milipitas, CA, "TC1775 Tri-Core user's manual system units," 2001.
- [10] First Silicon Solutions (FS2) Inc., Sunnyvale, CA, "AMBA navigator spec sheet," 2005.
- [11] ARM Ltd., San Jose, CA, "Embedded trace macro cell architecture specification," 2006.
- [12] ARM Ltd., San Jose, CA, "ARM. AMBA AHB Trace Macro cell (HTM) technical reference manual ARM DDI 0328D," 2007.
- [13] J. Gaisler, E. Catovic, M. Isomaki, K. Glembo, and S. Habinc, "GRLIB IP core user's manual. gaisler research," 2009.

Biography



Dr.K.Babulu¹ received the Ph.D. degree in VLSI from JNTUA, India in the year 2010. At present he has been working as Professor and headed with the department of ECE, UCEK, JNTUK, Kakinada, India. His research interests include VLSI Design, Embedded System Design, ASIC Design and PLD design. He has published over 29 papers in various national and international journals and conferences.



P. Anvesh² is a Project Associate pursuing his M.Tech Degree with Computers & Communications specialization at the department of Electronics and Communication Engineering, UCEK, JNTUK, Kakinada.