ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design & Implementation of OCP on a On-Chip Bus

K.Mounika
Student,
Department of ECE,
Vidya Bharathi Institute of Technology.

B.Ajay Kumar Yadidya, M.E
Assistant Professor & Internal Guide, Department of ECE,
Vidya Bharathi Institue of Technology.

B.Pragathi, Ph.D
Assistant Professor & HOD,
Department of ECE,
Vidya Bharathi Institue of Technology.

ABSTRACT:

This paper proposes a multi modes AHB on-chip bus tracer named AHB multi resolution bus tracer for versatile system-on-chip (SoC) debugging and monitoring. The bus tracer is capable of capturing the bus trace with different resolutions, all with efficient built-in compression mechanisms, to meet a diverse range of needs. In addition, it allows users to switch the trace resolution dynamically so that appropriate resolution levels can be applied to different segments of the trace. In this work we adopt the well-defined interface standard, the Open Core Protocol (OCP), and focus on the design of the internal bus architecture. We develop an efficient bus architecture to support most advanced bus functionalities defined in OCP, including different types of transactions using modes. So,here we are going to compare the proposed method with the existing method in order to prove the proposed method is efficient.

KEYWORDS:

OCP,SOC,TRACING,RESOLUTI-ON,MODES

INTRODUCTION:

The On-chip Bus is an important system-on-chip (SoC) infrastructure that connects major hardware components. Monitoring the on-chip bus signals is crucial to the SoC debugging and performance analysis/optimization. Unfortunately, such signals are difficult to observe since they are deeply embedded in a SoC and there are often no sufficient I/O pins to access these signals. Therefore, a straightforward approach is to embed a bus tracer in SoC to capture the bus signal trace and store the trace in an on-chip storage such as the trace memory which could then be off loaded to outside world (the trace analyzer software) for analysis. SOC chip usually contains a large number of IP cores that communicate with each other through on-chip buses.

As the VLSI process technology continuously advances, the frequency and the amount of the data communication between IP cores increase substantially. As a result, the ability of on-chip buses to deal with the large amount of data traffic becomes a dominant factor for the overall performance. The design of on-chip buses can be divided into two parts: bus interface and bus architecture. The bus interface involves a set of interface signals and their corresponding timing relationship, while the bus architecture refers to the internal components of buses and the interconnections among the IP cores. The widely accepted on-chip bus, AMBA AHB [1], defines a set of bus interface to facilitate basic (single) and burst read/write transactions. AHB also defines the internal bus architecture, which is mainly a shared bus composed of multiplexors. The multiplexer-based bus architecture works well for a design with a small number of IP cores. When the number of integrated IP cores increases, the communication between IP cores also increase and it becomes quite frequent that two or more master IPs would request data from different slaves at the same time. The shared bus architecture often cannot provide efficient communication since only one bus transaction can be supported at a time. In addition, the bus tracer is capable of tracing signals before/after the event triggering, named pre-T/post-T tracing, respectively. This feature provides a more flexible tracing to focus on the interesting points.

II. OCP INTERFACE:

Most of the bus functionalities defined in AXI and OCP are quite similar. The most obvious difference between them is that divides the AXI address channel independent channel write address and read address channel such that read and write transactions can be processed simultaneously. However, the additional area of the channels separately address the punishment. Some previous work has examined onchip buses of various aspects. The work in [3] and [4] develops high-level AMBA bus models with fast simulation speed and high accuracy of the timing.The authors in [7] propose an automatic approach to generate high-level bus models from a formal channel model of OCP. In both of the above work, the authors focus on fast and accurate simulation high level, but gave no real hardware implementation details. In [9], the authors implement the AXI interface on the shared bus architecture. Even though it costs less in area, the advantage of AXI in communication efficiency are limited by the shared bus architecture.In this article we present a powerful on-chip bus design with OCP as bus interface. We choose OCP because it is open to the public and the OCP-IP has some free tools to check this Protocol. Our proposed bus architecture is characterized lat / partial crossbarbased interconnect and realizes most transactions within the meaning of OCP, including

1) some transactions,

2) burst transactions,

3) lock transactions,

4) pipelined transactions, and

5) out-of - order transactions.

Moreover, the proposed bus is so flexible that the bus architecture can adapt to the system requirements.

III. ON CHIP BUS FUNCTIONALITIES

The various bus functionalities includes Burst, lock, pipelined, and out-of-order transactions.

A. Burst transactions:

The burst transactions allow the grouping of multiple transactions that have a certain address relationship, and can be classified into multi-request burst and single-request burst according to how many times the addresses are issued. Fig.1 shows the two types of burst read transactions. The multi-request burst as defined in AHB is illustrated in Fig.1(a) where the address information must be issued for each command of a burst transaction (e.g., A11, A12, A13 and A14).This may cause some unnecessary overhead. In the more advanced bus architecture, the single-request burst transaction is supported. As shown in Fig.1(b), which is the burst type defined in AXI, the address information is issued only once for each burst transaction. In the proposed bus design both burst transactions are supported such that IP cores with various burst types can use the proposed on-chip bus without changing their original burst behavior.

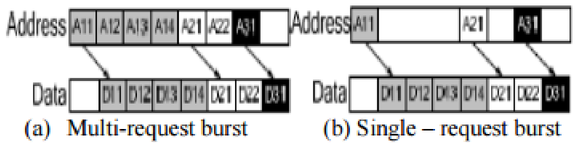
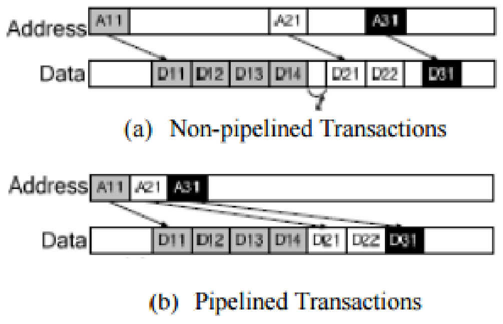


Fig.1 Burst transactions

B. Lock transactions Lock is a protection mechanism for masters that have low bus priorities. Without this mechanism the read/write transactions of masters with lower priority would be interrupted whenever a higher-priority master issues a request. Lock transactions prevent an arbiter from performing arbitration and assure that the low priority masters can complete its granted transaction without being interrupted.

C. Pipelined transactions (outstanding transactions) Fig. 2(a) and 2(b) show the difference between nonpipelined and pipelined (also called outstanding in AXI) read transactions. In Fig. 2(a), for a non-pipelined transaction a read data must be returned after its corresponding address is issued plus a period of latency. For example, D21 is sent right after A21 is issued plus t. For a pipelined transaction as shown in Fig. 2(b), this hard link is not required. Thus A21 can be issued right after A11 is issued without waiting for the return of data requested by A11 (i.e., D11-D14).



ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Fig. 2 Pipelined transactions.

D. **Out-of-order transactions :** The out-of-order transactions allow the return order of responses to be different from the order of their requests. These transactions can significantly improve the communication efficiency of an SOC system containing IP cores with various access latencies as illustrated in Fig. 3. In Fig. 3(a) which does not allow out-of-order transactions, the corresponding responses of A21 and A31 must be returned after the response of A11. With the support of outof-order transactions as shown in Fig. 3(b), the response with shorter access latency (D21, D22 and D31) can be returned before those with longer latency (D11-D14) and thus the transactions can be completed in much less cycles.

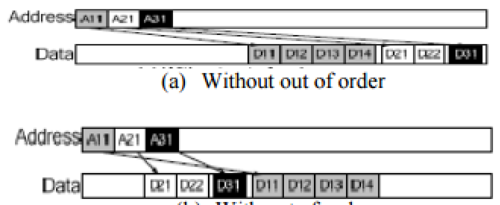


Fig.3. Out-of-order transactions

IV. ON-CHIP BUS DESIGN

The architecture of the proposed on-chip bus is illustrated in Fig. 4, where an example with two masters and two slaves is shown. A crossbar architecture is employed such that more than one master can communicate with more than one slave simultaneously. If not all masters require the accessing paths to all slaves, partial crossbar architecture is also allowed.

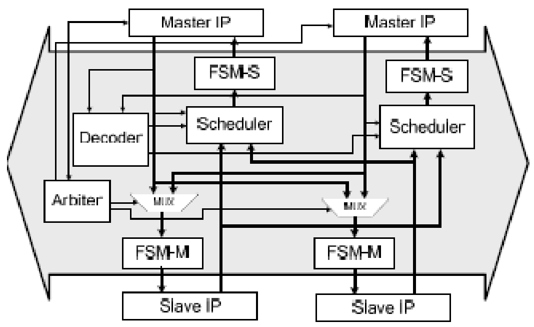


Fig.4 Block diagram of OCP bus architecture

A. **Arbiter**: In traditional shared bus architecture, resource contention happens whenever more than one master requests the bus at the same time. For a crossbar or partial crossbar architecture, resource contention occurs when more than one master is to access the same slave simultaneously. In the proposed design each slave IP is associated with an arbiter that determines which master can access the slave.

B. **Decoder** : Since more than one slave exists in the system, the decoder decodes the address and decides which slave return response to the target master. In addition, the proposed decoder also checks whether the transaction address is illegal or nonexistent and responses with an error message if necessary.

C. **Multiplexer**: A multiplexer is used to solve the problem of resource contention when more than one slave returns the responses to the same master. It selects the response from the slave that has the highest priority.

D.FSM-M & FSM-S: Depending on whether a transaction is a read or a write operation, the request and response processes are different. For a write transaction, the data to be written is sent out together with the address of the target slave, and the transaction is complete when the target slave accepts the data and acknowledges the reception of the data. For a read operation, the address of the target slave is first sent out and the target slave will issue an accept signal when it receives the message. The slave then generates the required data and sends it to the bus where the data will be properly directed to the master requesting the data.

The read transaction finally completes when the master accepts the response and issues an acknowledge signal. In the proposed bus architecture, we employ two types of finite state machines, namely FSM-M and FSM-S to control the flow of each transaction. FSM-M acts as a master and generates the OCP signals of a master, while FSM-S acts as a slave and generates those of a slave. These finite state machines are designed in a way that burst, pipelined, and out-or-order read/write transactions can all be properly controlled.

E. **Scheduler**

Out-of-order transactions in either OCP [2] or AXI [1] allow the order of the returned responses to be different from the order of the requests. In the OCP protocol, each out-oforder transaction is tagged with a TagID by a master. For those transactions with the same TagID, they must be returned in the same order as requested, but for those with different TagID, they can be returned in any order. In general, both in order and out-of-order transactions are supported in an out-of-order SOC system. A multiplexer, MUX1, is used to solve the problem of resource contention when more than one slave returns the responses to the same master. It selects the response from the slave that has the highest priority. The function of MUX2 will be described shortly. The recorder shown in the figure is used to keep track of the ID of the target slave and the TagID of every out-of-order transaction. Whenever a response arrives,the comparator determines whether the ordering restriction is violated or not by comparing the ID of the target slave and TagID. If no ordering restriction is violated, the response is sent forward to the priority setter. If the restriction is violated, the response is sent backward to one of the inputs of MUX2, which is always a preferred input over the input from MUX1.

The responses sent forward are given a priority, which is different from the slave priority, according to the TagID and are stored in the priority queue. For the transactions without TagID, which are regarded as in-order transactions, the priority setter sets the priority to 0 or the largest value to reflect whether in-order first or out-of-order first policy is used. Finally, the responses stored in the priority queue are returned to the masters from the first priority to the last priority such that the objective of “transactions with the same TagID are returned in-order, and transactions with different TagID can be returned out-of-order” can be achieved. To further improve the efficiency of the scheduler, the response can be forwarded to the master directly without going throughthe priority queue when the priority queue is empty.

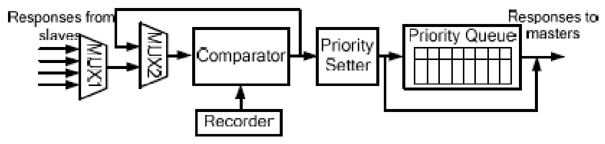


FIGURE 5. Block diagram of the scheduler

V. EXISTING METHOD:

Here, the existing method was about the Resolution concept. In this method we are interfacing OCP with the Multi- Resolution based AHB Bus. Resolution architecture mainly contains Master, arbiters, slave and their corresponding responses. This method is a continuos process i.e, here there exists a response signal between master to arbiter, arbiter to slave and from slave to arbiter which is a bi-directional response. But there is no such a response exists between slave to master.

Operation: Whenever we transmit the data to the master it will sends that data to the arbiter based on acknowledgement and then arbiter send response first to the slave in order to know whether slave is ready to take data or not. Slave receives the signal from the arbiter and it again resends the response to the arbiter whether empty or not.so based on slave’s response arbiter will sends the data to slave. But because of lack of response signal between the master and slave there is no such a response reply’s exist between slave and master so that master doesn’t know whether the data received or not so master waits for some amount of time and it again sends the new data like this it continuosly repeats its process without any Break in the signal so, because of this type of continuous process whenever we need a particular data from that address there will be data loss occurs and delay will gets increases and efficiency get decreases.

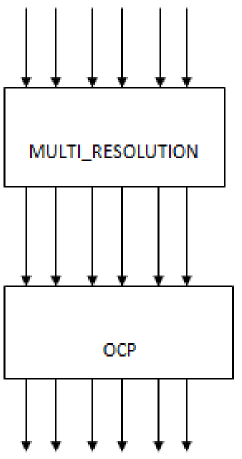


Figure: Block Diagram for Multi-Resolution OCP

So, in order to overcome all the above constraints we implemented a new method which is a Modes based OCP.

VI. PROPOSED METHOD:

In order to overcome the constraints in existing method we implemented Modes AHB Bus using OCP. Here, we are interfacing OCP with the Mode based architecture.

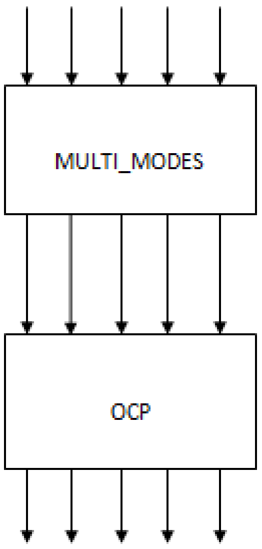


Figure: Block Diagram for Multi-Mode OCP

Combining the abstraction levels in the timing dimension and the signal dimension, we provide five modes in different granularities, as Fig shows. They are Mode FC (full signal, cycle level), Mode FT (full signal, transaction level), Mode BC (bus state, cycle level), Mode BT (bus state, transaction level), and Mode MT (master state, transaction level). We will discuss the usage of each mode in the following. 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. At **Mode FT**, the tracer traces all signals only when their values are changed. In other words, this mode traces the untimed data transaction on the bus. Comparing to Mode FC, the timing granularity is abstracted.

Another benefit of this mode is that the space can be saved without losing meaningful information. Thus, the trace depth increases. At **Mode BC**, the tracer uses the BSM, such as NORMAL, IDLE, ERROR, and so on, to represent bus transfer activities in cycle accurate level. Comparing to Mode FC, although 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. At **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 behaviors, such as read, write, or burst transfer. It is the highest abstraction level. This feature is very suitable for analyzing the master’s 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 master’s transactions.

VII. RESULTS & DISCUSSION

The proposed design is coded in VERILOG language and simulated using Xilinx ISE tool. The simulated waveforms for BC, BT,FC,FT, MT are shown in following figures.

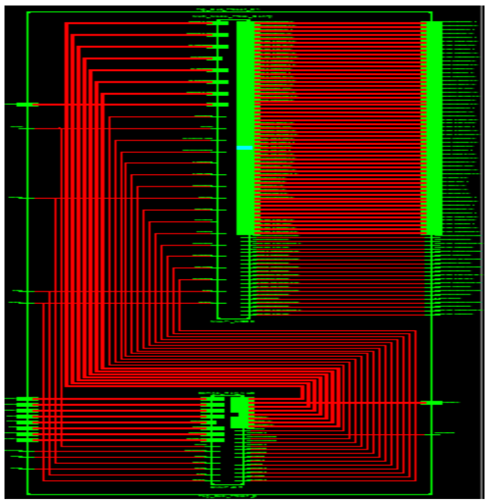


Fig: rtl schematic

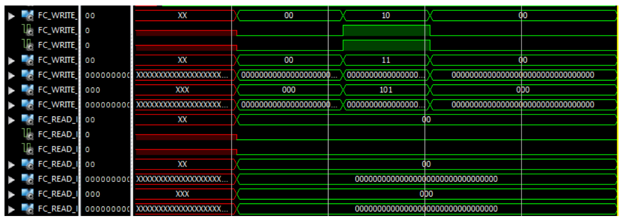


Fig: FC_Mode

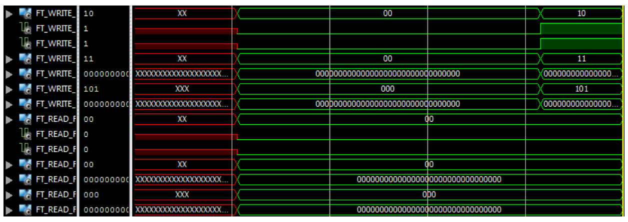


Fig: FT_Mode

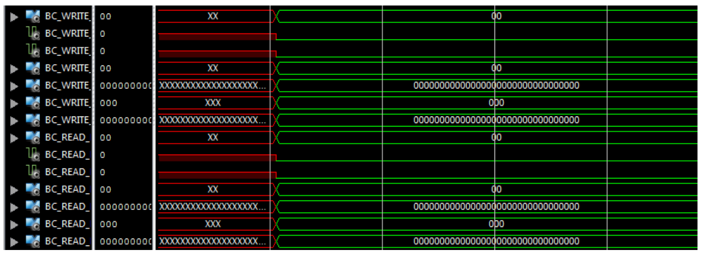


Fig: BC_Mode

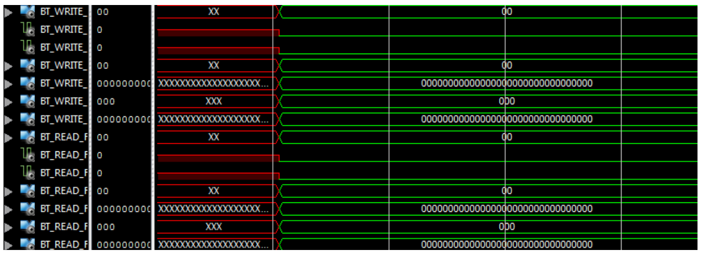


Fig: BT_Mode

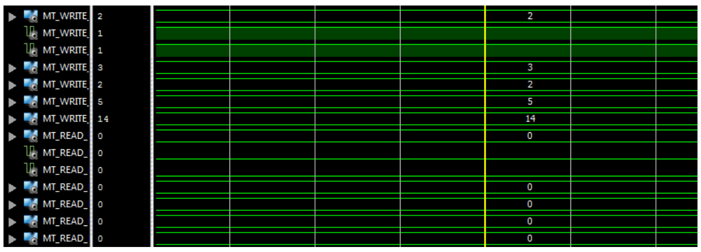
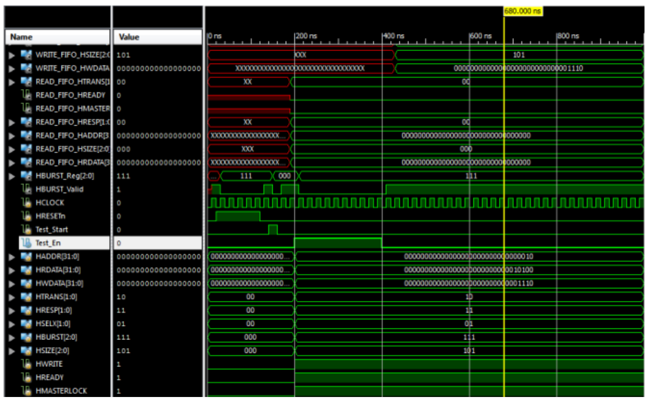


Fig: MT_Mode


a)

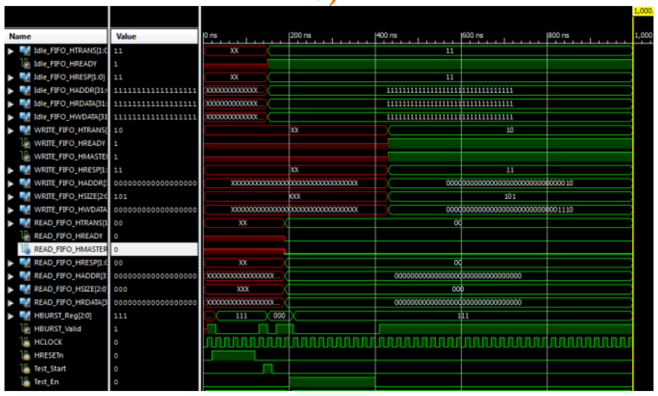
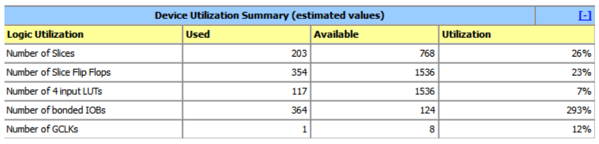

b)

Fig: a) & b) Waveforms for MODE-OCP

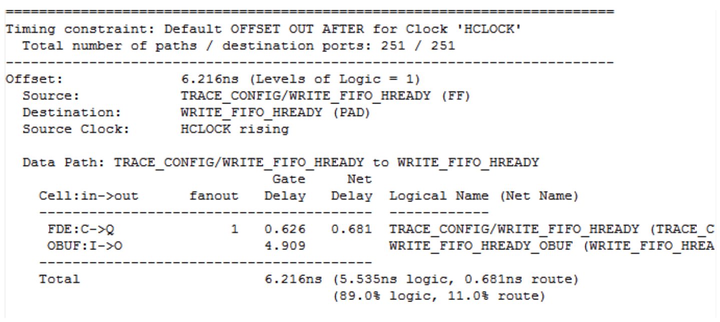
VIII. AREA & DELAY REPORTS

The following figures shows the area and delay reports for existed and proposed methods



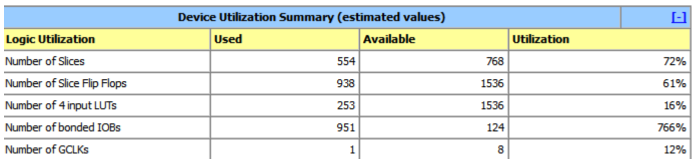
Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	203	768	26%
Number of Slice Flip Flops	354	1536	23%
Number of 4 input LUTs	117	1536	7%
Number of bonded IOBs	364	124	293%
Number of GCLKs	1	8	12%

Fig: exis area



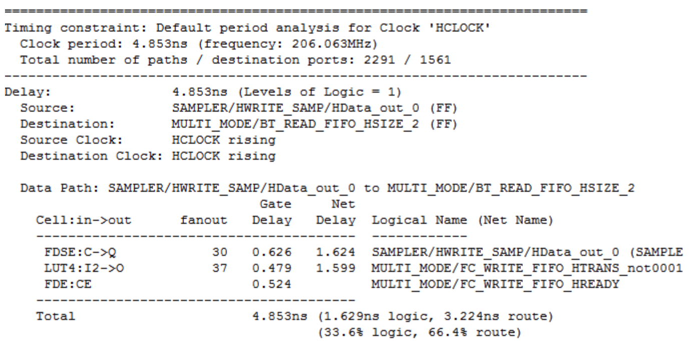
```
==========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'HCLOCK'
  Total number of paths / destination ports: 251 / 251
--------------------------------------------------------------------------
Offset:              6.216ns (Levels of Logic = 1)
  Source:            TRACE_CONFIG/WRITE_FIFO_HREADY (FF)
  Destination:       WRITE_FIFO_HREADY (PAD)
  Source Clock:      HCLOCK rising

  Data Path: TRACE_CONFIG/WRITE_FIFO_HREADY to WRITE_FIFO_HREADY
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDE:C->Q              1   0.626   0.681  TRACE_CONFIG/WRITE_FIFO_HREADY (TRACE_C
     OBUF:I->O                 4.909          WRITE_FIFO_HREADY_OBUF (WRITE_FIFO_HREA
    ----------------------------------------
    Total                      6.216ns (5.535ns logic, 0.681ns route)
                                       (89.0% logic, 11.0% route)
```

Fig delay _exis



Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	554	768	72%
Number of Slice Flip Flops	938	1536	61%
Number of 4 input LUTs	253	1536	16%
Number of bonded IOBs	951	124	766%
Number of GCLKs	1	8	12%

Fig: proposed area



```
=========================================================================
Timing constraint: Default period analysis for Clock 'HCLOCK'
  Clock period: 4.853ns (frequency: 206.063MHz)
  Total number of paths / destination ports: 2291 / 1561
-------------------------------------------------------------------------
Delay:               4.853ns (Levels of Logic = 1)
  Source:            SAMPLER/HWRITE_SAMP/HData_out_0 (FF)
  Destination:       MULTI_MODE/BT_READ_FIFO_HSIZE_2 (FF)
  Source Clock:      HCLOCK rising
  Destination Clock: HCLOCK rising

  Data Path: SAMPLER/HWRITE_SAMP/HData_out_0 to MULTI_MODE/BT_READ_FIFO_HSIZE_2
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDSE:C->Q            30   0.626   1.624  SAMPLER/HWRITE_SAMP/HData_out_0 (SAMPLE
     LUT4:I2->O           37   0.479   1.599  MULTI_MODE/FC_WRITE_FIFO_HTRANS_not0001
     FDE:CE                    0.524          MULTI_MODE/FC_WRITE_FIFO_HREADY
    ----------------------------------------
    Total                      4.853ns (1.629ns logic, 3.224ns route)
                                       (33.6% logic, 66.4% route)
```

Fig: delay _propose

XI. CONCLUSION:

In proposed method we implemented OCP Bus using Multi resolution modes, so that based on modes the system will transfer the data properly from Master to Slave. The Real-time Compression and Dynamic Multi-Resolution AHB bus tracer in SoC was designed successfully and the coding was done in VERILOG.. The synthesis was done using Xilinx ISE. 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. So based on experimental results the proposed method was proved to be very efficient way in data communication when compared to that of the existing method.

REFERENCES:

[1] Advanced Microcontroller Bus Architecture (AMBA) Specification Rev 2.0 & 3.0, http://www.arm.com.
[2] Open Core Protocol (OCP) Specification, http://www.ocpip.org/home.
[3] Y.-T. Kim, T. Kim, Y. Kim, C. Shin, E.-Y. Chung, K.-M. Choi, J.-T. Kong, S.-K. Eo, "Fast and Accurate Transaction Level Modeling of an Extended AMBA2.0 Bus Architecture," Design, Automation, and Test in Europe, pages 138-139, 2005.
[4] G. Schirner and R. Domer, "Quantitative Analysis of Transaction Level Models for the AMBA Bus," Design, Automation, and Test in Europe, 6 pages, 2006.
[5] C.-K. Lo and R.-S. Tsay, "Automatic Generation of Cycle Accurate and Cycle Count Accurate Transaction Level Bus Models from a Formal Model," Asia and South Pacific Design
Automation Conference, pages 558-563, 2009.
[6] N.Y.-C. Chang, Y.-Z. Liao and T.-S. Chang, "Analysis of Shared-link AXI," IET Computers & Digital Techniques, Volume 3, Issue 4, pages 373-383, 2009.
[7] IBM Corporation, "Prioritization of Out-of-Order Data Transfers on Shared Data Bus," US Patent No. 7,392,353 2008.
[8] David C.-W. Chang, I.-T. Liao, J.-K. Lee, W.-F. Chen, S.-Y. Tseng and C.-W. Jen, "PAC DSP Core and Application Processors," International Conference on Multimedia and Expo, pages 289-292, 2006.
[9] CoWare website, http://www.coware.com
[10] ARM Ltd., San Jose, CA, "AMBA Specification (REV 2.0) ARM IHI0011A," 1999.
[11] 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.
[12] ARM Ltd., San Jose, CA, "ARM. AMBA AHB Trace Macrocell (HTM) technical reference manual ARM DDI 0328D," 2007.
[13] First Silicon Solutions (FS2) Inc., Sunnyvale, CA, "AMBA navigator spec sheet," 2005.
[14] J. Gaisler, E. Catovic, M. Isomaki, K. Glembo, and S. Habinc, "GRLIB IP core user's manual, gaisler research," 2009.
[15] Infineon Technologies, Milipitas, CA, "TC1775 TriCore users manual system units," 2001.
[16] ARM Ltd., San Jose, CA, "Embedded trace macrocell architecture specification," 2006.
[17] E. Rotenberg, S. Bennett, and J. E. Smith, "A trace cache microarchitecture and evaluation," IEEE Trans. Comput., vol. 48, no. 1, pp. 111–120, Feb. 1999.
[18] 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.
[19] B. Tabara and K. Hashmi, "Transaction-level modeling and debug of SoCs," presented at the IP SoC Conf., France, 2004.
[20] 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.

[21] 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.
[22] 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 curcular buffer,” in Proc. Des. Autom. Conf. (DAC), Jul. 2008, pp. 862–865.
[23] ARM Ltd., San Jose, CA, “Example AMBA system user guide ARM DUI0092C,” 1999.
[24] R. -T Gu, T.-C Yeh, W.-S Hunag, T.-Y. Huang, C.-H Tsai, C.-N Lee, M.-C Chiang, S.-F Hsiao, and I.-J.H Yun-Nan Chang, “A low cost tilebased 3D graphics full pipeline with real-time performance monitoring support for opengl es in consumer electronics,” in Proc. ISCE, Jun.20–23, 2007, pp. 1–6.