International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 11, November 2015, PP 29-38
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Interfacing OCP on A On-Chip Bus

Mohammad Yousufuddin[1], G. Bharathi Subhashini[2]

[1]Department of ECE, MREC (Autonomous), JNTUH, Hyderabad, India (PG Scholar)
[2]Department of ECE, MREC (Autonomous), JNTUH, Hyderabad, India (Associate Professor)

ABSTRACT

This paper proposes a multi modes AHB on-chip bus tracer for versatile System on Chip (SOC) debugging and monitoring named AHB multi resolution bus tracer. This bus tracer is capable of capturing the bus trace with different resolutions, with all the efficient built-in compression mechanism to meet different range of needs. In addition, it also allows the users to switch the tracer resolution dynamically so that the appropriate resolutions levels can be applied to different segments of the trace. In this paper for this work we are using the well-defined interface standard, the Open Core Protocol (OCP) and will be focusing on the design of the internal bus architecture. We are going to built an efficient bus architecture that support the advanced bus functionalities which are defined in Open core Protocol (OCP), which also includes different types of transactions using modes. Here we are going to compare the proposed method with the existing method in order to prove the proposed method is more efficient than the existing. We also prove some constraints like area, delay, power, efficiency. It is done in XILINX 14.2 version using Verilog HDL language.

Keywords: soc, ocp, tracing, resolution, modes.

INTRODUCTION

The On-chip bus is an important System on Chip (SOC) infrastructure. In other words, grouping lot of protocols or any other blocks on a single chip is said to be System on Chip (SOC) which connects other major components. Monitoring the On-chip bus is crucial to the SoC debugging and performance analysis/optimization. Because of this, such signals are difficult to observe since they are embedded in a SoC and there are often no sufficient I/O pins to access these signals. Therefore, a straight forward approach is taken where a bus tracer in SoC will capture the bus signal trace and store the trace in an on-chip storage in such a way that the trace memory can be offloaded to outside world for analysis. SoC chip mainly contains a large number of IP cores that communicate with each other through on-chip buses. In recent days, the growth of SoC chips and reusable IP cores were given higher priority because of its less cost and reduction in the period of time to market. These interfaces play an important role in SoC and should be taken care because of the communication between the IP cores. As the VLSI technology is increasing continuously, the data communication between the IP cores also increasing substantially. Due to which the ability of the on-chip bus to deal with large amount of data is becoming a dominant factor for all the over-all performance. The on-chip bus design can be mainly divided into two parts bus architecture and bus interface. The bus interface refers to the set of their interface signals and corresponding timing relationship. Whereas the bus architecture contains the internal components of buses and 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 component of multiplexors. The multiplexer based bus architecture works-well with a design which has less/small number of IP cores. When the number of integrated IP cores increases, it is quite obvious that the data communication between the IP cores also increases. And it will become quite frequent that two or more master IP's

Address for Correspondence:

Yousufuddin425@gmail.com

would request the data from different salves at the same time. Here, in this work the bus tracer is capable of tracing signals before/after the event triggering, named pre-T/post-T tracing respectively. This factor provides more flexible tracing to focus on the interesting points.

OCP INTERFACE

Open Core Protocol (OCP) is an interface aiming to standardize and simplify the system integration problem. The Open Core Protocol (OCP) will focus mainly on the IP core requirements instead of the on-chip bus requirements. The Open Core Protocol (OCP) protocol is a communication protocol which defines point to point interface between two communicating entities, such as IP cores and bus interface modules. Where one entity acts as the Master and other as the Salve. If we want to communicate multiple masters and multiple slaves at a time with a grand signal with address and the data lines, there will be a miss communication occurs. There is data over riding miss communication occurs. To avoid data over riding and for the efficient communication we are making use of a protocol the OCP protocol. This will be deciding 'what master has to make use of the properties of what slave, and what slave has to make use of the properties of what master' at a time.

Most of the bus functionalities defined in both AXI and OCP are quite similar. The main difference between them is that divides the AXI address channel independent channel write address and read address channel such that both the write and read transactions can be processed at the same time. However the additional area of the channels is the punishment. Some previous work has examined on chip buses of different aspects. The work in [3] and [4] shows high-level AMBA bus models with fast simulation speed and high accuracy of the timing.

In this work we present a powerful on-chip bus design with OCP as the bus interface. We are choosing this OCP because this OCP will focus mainly on the IP core requirements instead of on-chip bus requirements. It is also be chosen because it is open to public and provide some free tools to verify this protocol. Moreover, the proposed bus is flexible in such a way that the bus architecture can adapt to system requirements.

ON CHIP BUS

The different bus functionalities are Burst, Lock, Pipelined and Out of order transactions.

Burst Transactions

The burst transactions are comprised of a set of transfer linked together having a defined address sequence and number of transfers. Different combinations of protocol phases are used by different types of transfers. It can be classified into multi-request burst and single request burst. Figure1 shows the two types of burst transactions. In Figure 1(a) shows the multi burst transaction where the address information must be issued for each command of a burst transaction. Due to which unnecessary overhead occurs. Whereas Figure 1(b) shows the single burst transaction where the address information is issued only once for each burst transaction.

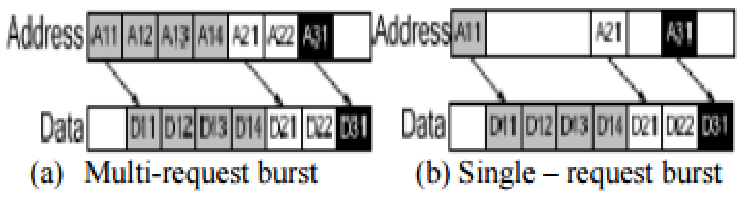


Fig1. *Burst transactions*

Lock Transactions

Lock Transaction is a protection mechanism for masters that have low bus priorities. Whenever a higher priority master issues a request, the read/write transactions of the master with the lower priority would be interrupted without this mechanism. It prevents an arbiter from performing arbitration and assures that the lower priority masters can complete its granted transactions without being interrupted.

Pipelined Transactions (Outstanding Transaction)

The below Figure 2(a) and (b) shows the difference between the Pipelined and Non-pipelined (also called Outstanding transaction). 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 E.g.; D21 is sent right after A21 is issued plus *t*. On other side for a pipelined transaction shown in Fig 2(b) this hard link is not required. Therefore, A21 can be issued right after A11 is issued without waiting for the return of data requested by A11(i.e., D11-D14).

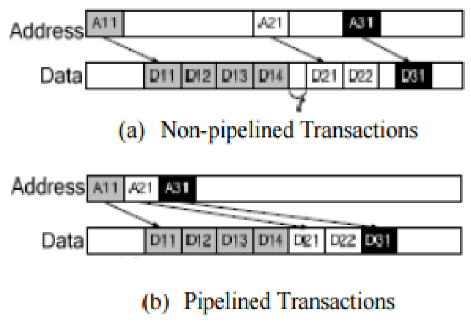


Fig2. *Pipelined transactions*

Out-of-Order Transaction

The out-of-order transactions allow the return order of responses to be different from the order of their requests. These types of transaction can definitely improves or increases the communication efficiency of the SoC systems containing IP cores with various access latencies as shown in the below Figure3. The Fig 3(a) shows the transaction which does not allow out-of-order transaction the corresponding responses of A21 and A31 must be returned after the response of A11. Where on the other hand Fig 3(b) support out-of-order transaction where the response with the shorter access latency (D21, D22 and D31) can be returned before those with longer latency (D11-D14) and thus the transaction can be completed in much less cycles.

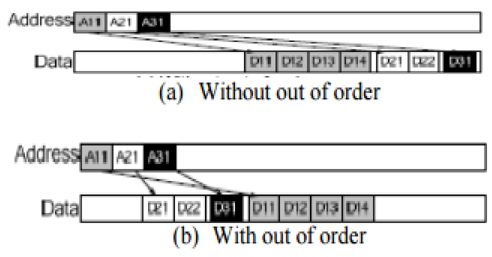


Fig3. *Out-of-order transactions*

ON CHIP BUS DESIGN

The architecture of the proposed on-chip bus is illustrated in Figure 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.

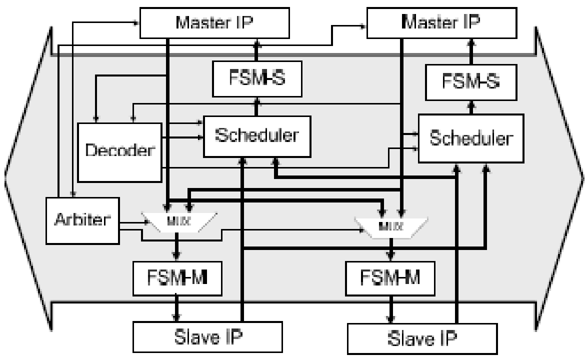


Fig4. *Block diagram of OCP bus architecture*

Arbiter

At a time multiple master is been granting or taking the service of a single slave then we can prioritize these masters with the help of a block called Arbiter block. In shared bus architecture, resource contention happens whenever more than one master request the bus at the same time. For crossbar/partial crossbar architecture, resource contention occurs when more than one master is to access the same slave simultaneously. Therefore, here each slave IP is associated with an arbiter that determines which master can access the slave.

Decoder

In the OCP protocol decoder selects the one slave among the slaves which are more than one exists in the system and decodes the address and decides which slave return response to the target master. Decoder also checks whether the transactions address is illegal or nonexistent and responses with an error message.

Multiplexer

A multiplexer solves the problem of resource contention whenever more than one slave returns the responses to the same master. It selects the response from the salve that has the highest priority.

FSM-M & FSM- S

The request and response processes is purely depends on the fact that whether a transaction is a read or a write operation. For a write transaction, the data which is to be written is sent out along with the address of the target slave, the transaction is considered to be completed only when the target slave accepts the data and acknowledges the reception of the data. For the read operation, the address of the target salve is first sent out and the target slave will issue an accept signal when it receives the message.

Scheduler

Out-of-order transactions in both OCP and AXI allow the order of the returned responses to be different from the order of the requests. In particularly OCP protocol, each out-of-order transaction is tagged with a tag ID by a master. For those transactions with the same Tag ID, they must be returned in the same order as requested.

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 continuous 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 continuously 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.

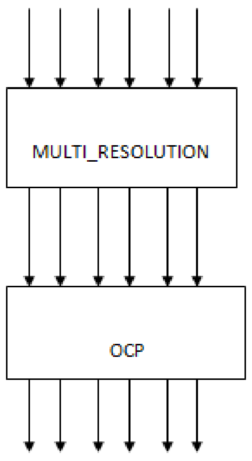


Fig. *Block Diagram for Multi-Resolution OCP*

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

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.

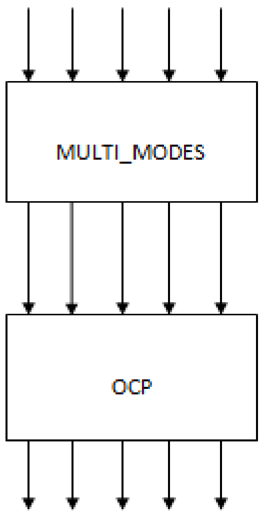


Fig. *Block Diagram for Multi-Mode OCP*

Combining the abstract level in the signal dimension and in the timing dimension, we are using five modes in different granularities as shown in the below figures. These five modes 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). Now, we will discuss the usage of these modes in the following.

At Mode FC (full signal, cycle level), the tracer traces all the bus signals cycle-by-cycle by which the designer can observe the most detailed bus activities. This Mode FC is very useful. By looking at the detail signals the cause of the error can be diagnosed with the help of this Mode. However, since the traced data size of this mode is huge, the trace depth is the shortest among the five modes.

At Mode FT (full signal, transaction level), in this Mode the tracer traces all the signals only when their values are changed. In other words, this mode traces the untimed data transaction on the bus. When compare to the Mode FC, the timing granularity is abstracted in the Mode FT. Hence the trace depth in the Mode FT increases.

At Mode BC (bus state, cycle level), to represent bus transfer activities in the cycle accurate level, the tracer uses the BSM such as NORMAL, IDLE, ERROR and so on. Compare to Mode FC, although this Mode BC still captures the signals cycle-by-cycle, the signal is abstracted. Thus, the designer can observe the bus handshaking states without analyzing the detail signals.

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

At Mode MT (master state, transaction level), here the tracer records only the master's behaviors such as read, write or burst transfer. It is the highest abstraction level. This feature is very suitable for analyzing the master's transaction. The major difference of the Mode MT compared with Mode BT is that, this Mode does not record the transfer handshaking activities and does not capture signals when the state of the bus is IDLE, WAIT and BUSY. Thus, designers can concentrate only on the master's behavior.

RESULTS & DISCUSSION

The proposed design is coded in VERILOG language and stimulated by using Xilinx ISE tool. The following figures show the stimulated waveforms for BC, BT, FC, FT and MT.

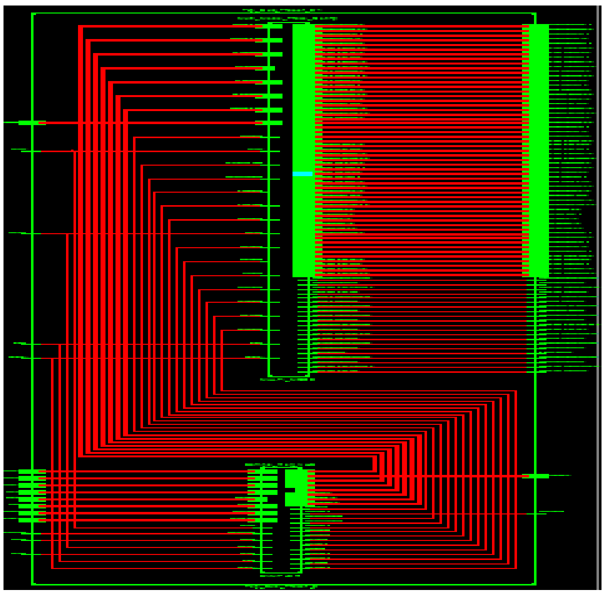


Fig. *rtl schematic*

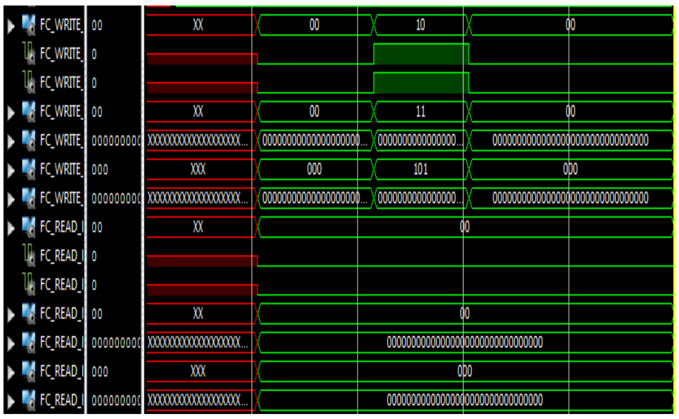


Fig. *FC_Mode*

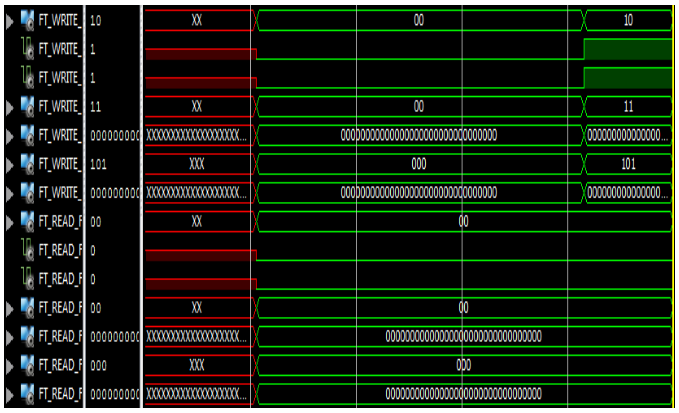


Fig. *FT_Mode*

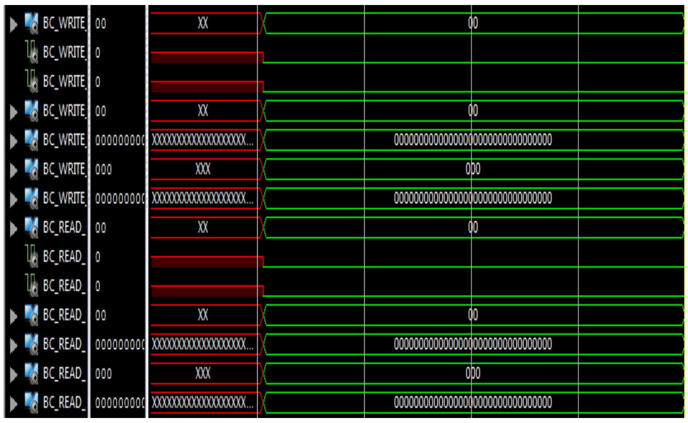


Fig. *BC_Mode*

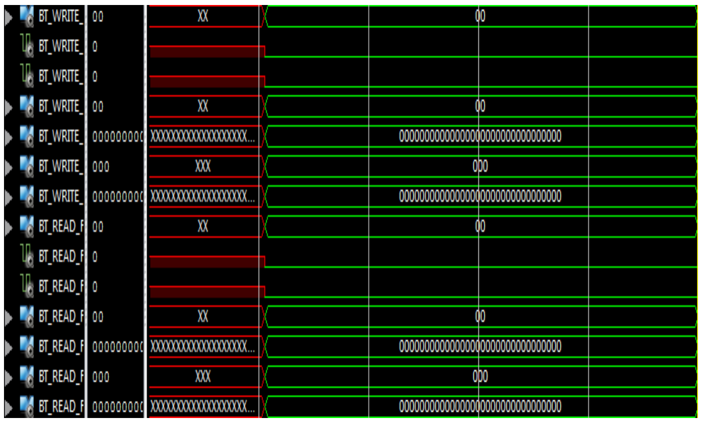


Fig. *BT_Mode*

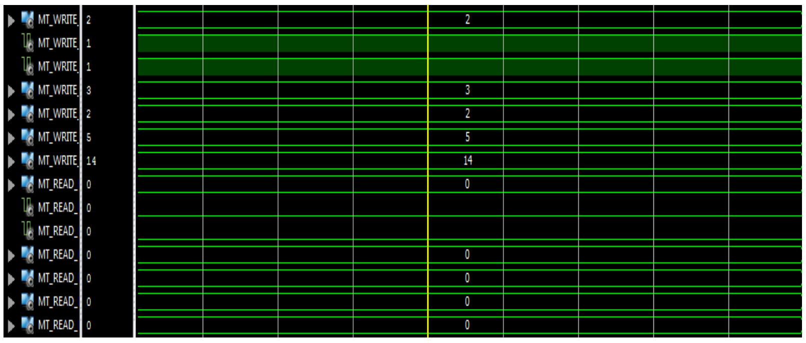
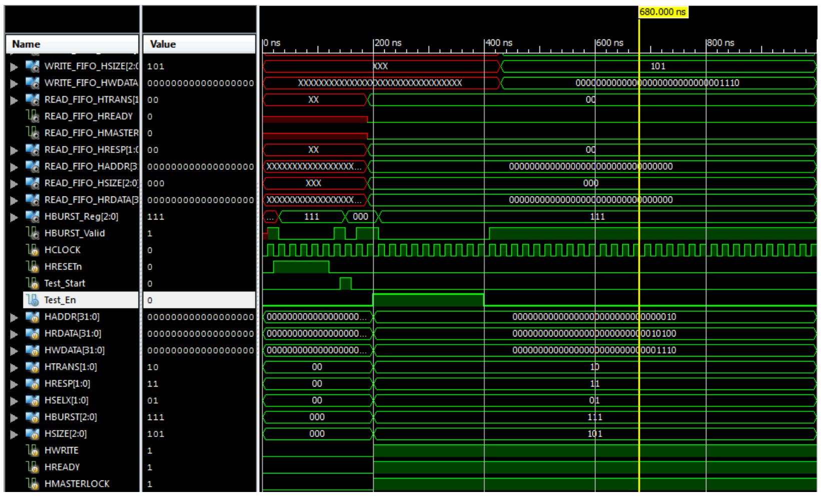
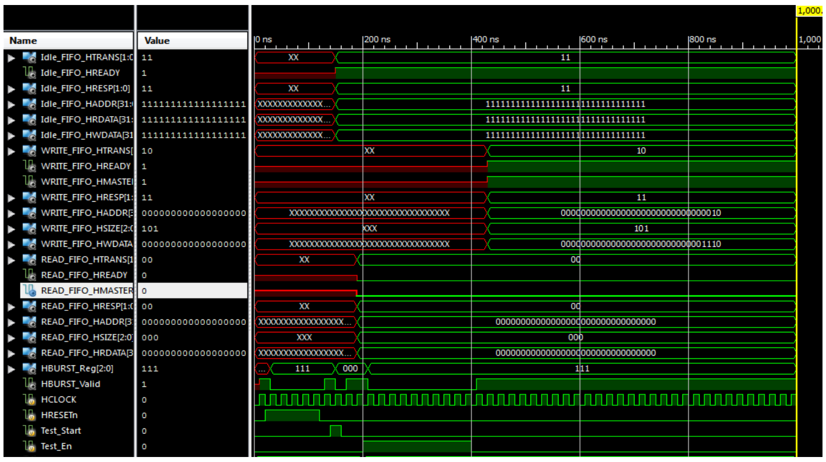


Fig. *MT_Mode*



(a)

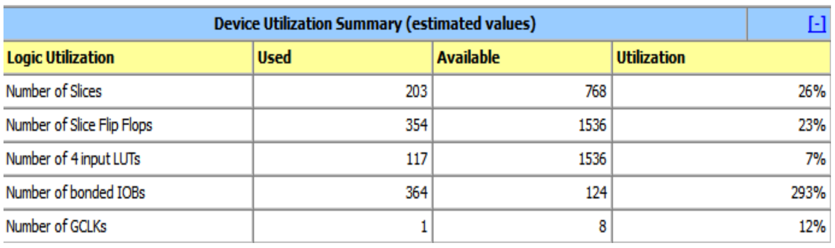


(b)

Fig. *(a) & b) Waveforms for MODE-OCP*

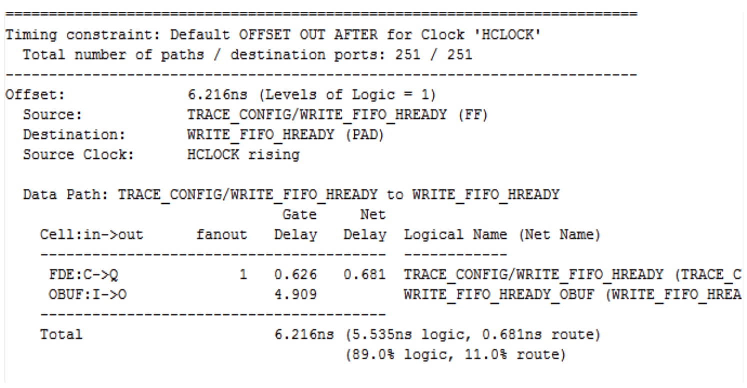
AREA & DELAY

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



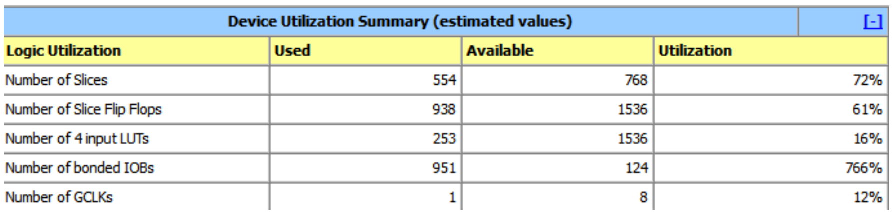
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	203	768	26%
Number of Slice Flip Flops	354	1536	23%
Number of 4 input LUTs	117	1536	7%
Number of bonded IOBs	364	124	293%
Number of GCLKs	1	8	12%

Fig. *exist_ area*



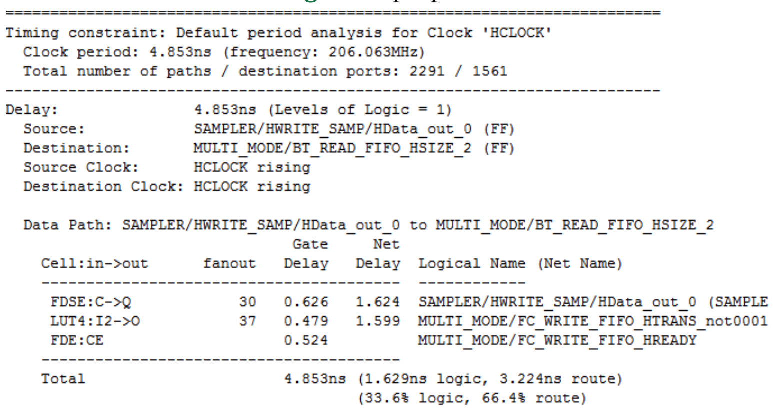
```
========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'HCLOCK'
  Total number of paths / destination ports: 251 / 251
------------------------------------------------------------------------
Offset:              6.216ns (Levels 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 _exist*



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. *area_proposed*



```
========================================================================
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*

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 is designed successfully and the coding is done in VERILOG. The synthesis is done by using the Xilinx ISE tool. The Designed Tracer works properly for all the modes such as Mode FC, Mode FT, Mode BC, Mode BT and Mode MT. Tracer design is verified for all the test cases. So based on the experimental results the proposed method is proved to be very efficient way in data communication when compared to that of the existing method.

ACKNOWLEGEMENT

I am very much thankful to my project guide Mrs. G. BHARATHI SUBHASHINI, Associate Professor, for her extensive patience and guidance throughout my project work.

I am grateful to our Project Coordinator, Mr. Kesavan Gopal, Associate Professor, for extending his support and cooperation throughout my project work.

I would like to thank Dr. M. Ch. P. Jagadesh, Head of the ECE Department, for providing the freedom to use all the facilities available in the department, for successful completion of project.

I whole-heartedly express my deep sense of gratitude to our beloved Principal, Dr. S.SUDHAKAR REDDY, Principal, MALLA REDDY ENGINEERING COLLEGE (Autonomous) for providing me adequate facilities for my successful completion of both course work and project work.

I sincerely thank all the staff of the Department, for their timely suggestions, healthy criticism and motivation during the course of my project work.

I would also like to thank all my friends for providing help and moral support at the right timing. With great affection and love, I thank my parents who were the backbone behind my deeds.

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) ARMIHI0011A," 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, "GRLIBIP 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 macro cell 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.

AUTHOR'S BIOGRAPHY

Mohammad Yousufuddin

Department of ECE, MREC (Autonomous), JNTUH, Hyderabad, India (PG Scholar)