Design and Development of Cache Coherent Interconnect based on ACE Protocol Specification

Soubhagya S Prabhu
Dept. of Electronics
ER&DCIIT
Thiruvananthapuram

Kadar A A
Principal Engineer
Dept. of Electronics
ER&DCIIT
Thiruvananthapuram

Jose Simon
Principal Engineer
Hardware Design Group
CDAC
Thiruvananthapuram

***Abstract*— In multiprocessor system, each core has a private cache to improve system performance. Cache coherency ensures that all processors are always provided with latest version of stored data. Hardware coherency which is implemented with the help of interconnect simplifies the software and reduces overheads associated with software coherency. On chip Interconnect provides a communication interface to multiple processor cores for loading and storing of data and instruction, managing multiple accesses, providing resource allocation and maintaining cache coherency. This work aims to design an interconnect system for a multicore processor after selection of appropriate flow control protocol, routing algorithm and a router microarchitecture. The proposed work involves realization of multiple channels using independent 2D mesh topology based on ACE protocol specification. The work is to be implemented using Bluespec SystemVerilog language.**

Keywords— Cache Coherent Interconnect (CCI); Router micro-architecture; ACE protocol; Mesh topology; Bluespec SystemVerilog language.

I. INTRODUCTION

The design of an on-chip network can be broken down into its various building blocks: its topology, routing, flow control, router microarchitecture. The topology of a network determines how the nodes and links are connected. While the topology determines all possible paths a message can take through the network, the routing algorithm selects the specific path a message can take from source to destination. The flow control protocol determines the resource allocation associated with a router node. This involves the determination of buffer size and channel width. Finally, the micro architecture of a router realizes the routing and flow control protocols in hardware.

In multi-core system, each core has its own private This requires maintenance of cache coherence. Coherency ensures only valid data is available to all cores at all times. The cache coherence problem is: multiple copies of the same data can exist in different caches simultaneously, and if processors are allowed to update their own copies independently, it can result in an inconsistent view of memory. Hardware coherency is the adopted solution wherein interconnect is responsible for ensuring coherency. This is achieved with the help of snoop filter module.

.

II. LITERATURE SURVEY

Reference [1] describes interconnect architecture consisting of 2D mesh networks (iMesh) that provide the transport system for on-chip memory access, I/O, tile to tile communication, interrupts and other communication activity[1]. A tiled multicore architecture is a multiple-instruction, multiple-data (MIMD) machine consisting of a 2D grid of homogeneous, general-purpose compute elements, called cores or tiles. Each tile consists of processor, cache and switching engine. The five networks are the user dynamic network (UDN), I/O dynamic network (IDN), static network (STN), memory dynamic network (MDN), and tile dynamic network (TDN). Four networks are dynamic and use packetized, _re and forget interface. Dimension Ordered Routing (DOR) with the header word denoting destination location, packet's length is used. Latency of one cycle occurs when packet goes in same direction and to turn at a switch to go in another direction takes another extra cycle. The static network allows a circuit switched communications channel and is ideal for streaming data. It contains an auxiliary processor for reconfiguring the network in a programmatic manner.

Reference [2] describes an ordered mesh Network-on-Chip (NoC) architecture with a separate fixed latency, bufferless network to achieve distributed global ordering [2]. It consists of 36 processors connected by 6x6 mesh network. Each tile includes a processor, split L1 I/D caches, a private L2 cache supporting snoopy coherence protocol, a network interface and router.

The notification network which is used for achieving global ordering, carry information regarding all the request sending nodes. Every node performs a local decision on the order in which it will service the requests. This fixes the global order for the actual messages in the system. This global order is captured through a counter maintained at each node called the expected source ID (ESID). Messages travel through the main network and are delivered to all nodes in the system. However, they are processed by the network interface (NIC) at every node in accordance to the global order. AMBA (Advanced Microcontroller Bus Architecture) ACE (AXI Coherency Extension) Protocol specification by ARM gives detailed description of the channels involved, control signals associated with each channel as well as allowed transactions and the ordering model [3].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 05, May-2019

III. ACE PROTOCOL SPECIFICATION

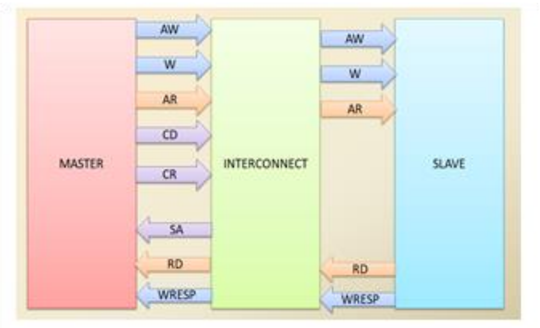


Fig. 1. ACE protocol based channels

ACE protocol supports system level coherency. It consists of 8 independent channels with corresponding signal definitions. Each channel has corresponding valid and ready handshakes to ensure information transfer takes place correctly. The main transactions supported by the protocols have also been included.

The 8 independent channels supported by the protocol are Write Address channel [AW], Read address channel [AR], Read data channel[RD], Write data channel[WD], Write Response channel[B], Snoop Address channel[AC], Snoop Data channel[CD] and Snoop Response [CR] channel. Write Address channel provides information regarding address locations to which master wants to write data. The corresponding data is provided via write data channel. When slave receives write transaction request from master, the response is provided using write response channel. Completion of a write transaction includes sending of WACK (write acknowledgement) signal from master to slave.

Read Address channel provides information regarding address locations from which master wants to read data. The slave provides requested data via read data channel.The completion of transaction is obtained by sending RACK (Read acknowledge) from master to slave. For both these operations, it might be necessary to perform certain snooping transaction to ensure data validity. These are provided with the help of snoop address channel. It specifies the address which is to be snooped from the snooped master. The response and data is provided in snoop data and snoop response channel respectively.

The supported transaction groups in ACE include non-snooping, coherent, memory update and cache maintenance transactions. Non snooping transactions can be ReadNoSnoop and WriteNoSnoop transactions. Coherent transaction group include ReadClean, ReadShared, ReadOnce, ReadNotSharedDirty, ReadUnique, CleanUnique, MakeUnique, WriteUnique, WriteLineUnique transactions. Cache maintenance transaction group includes CleanShared,CleanInvalid and MakeInvalid transactions. Memory update transaction groups include WriteClean,WriteBack and Evict transactions.

IV. ARCHITECTURE DESIGN

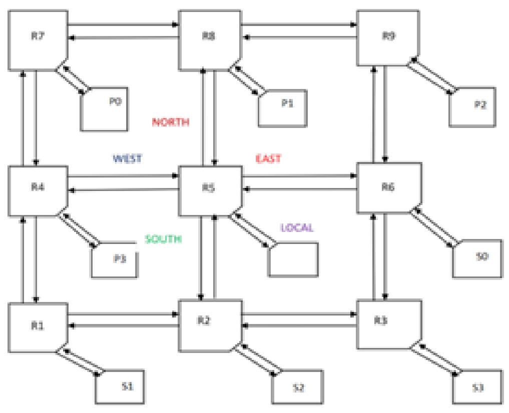


Fig. 2. Single channel with 3x3 mesh based Architecture

The design has 9 routers which can be connected to any master or slave as shown in Fig.2. As the protocol specification lists 8 channels first stage involves design of router suitable for a single channel. The channel under consideration is write address channel. Once a channel has been properly designed this can be extended to include all 8 channels by making certain modifications.

The micro router architecture is shown in Fig.3. To use the router, all the signals in the channel get converted to 32 bit packets. It is possible to extend packet size by using multiple flits of 32 bits to represent a single packet. The message in form of packet on reaching router gets stored in the respective buffers.

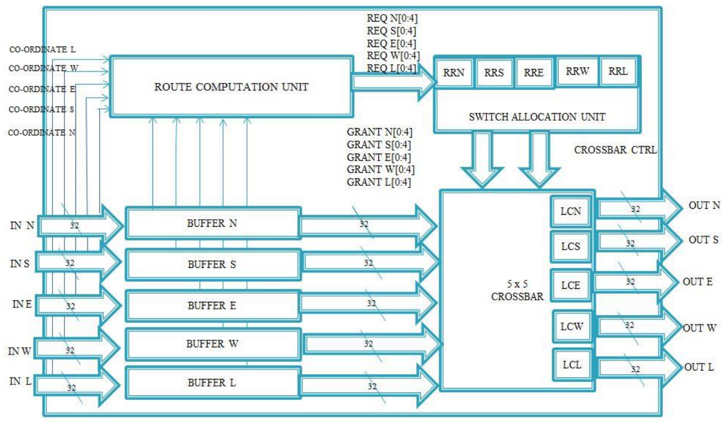


Fig. 3. Router Micro-architecture

The mesh topology based routers have bi directional links in North, South, East, West and Local directions. The local corresponds to the home node. The packet structure is arranged in such a manner that it is possible to get routing information. This is extracted using the route computation module which provides information to switch allocator. Switch allocator grants requests to each output port based on the requests. Round robin arbitration technique is employed for each output port grant. It generates crossbar control signals which establish connection to the next router. When the ready and valid handshake signals as a part of link control are active, packet transfer takes place.

Dimension ordered routing is a form of deterministic routing algorithm in which the path from source to destination is fixed. In YX dimension ordered routing algorithm first Y coordinate of destination and current router is compared and then next router is decided. If the value is positive go northwards else southwards. When the y coordinate is same for both current router and destination router, x coordinates are compared. If the value is positive go eastwards else westwards. When both coordinates are same, the packet is taken to local node.

Store and forward flow control makes each node wait until an entire packet has been received before forwarding any part of the packet to the next node.

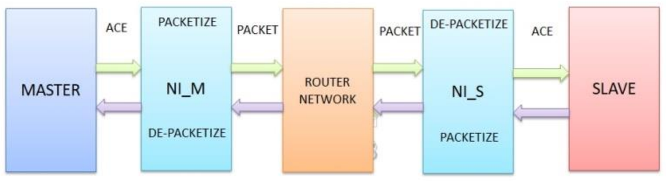


Fig. 4. Overall Single master – single slave System Architecture

The router network is capable of handling information signals in form of packets. The master and slave components are capable of sending and receiving only ACE signals. Network interface unit is responsible for ensuring proper packetization and de-packetization of information signals.

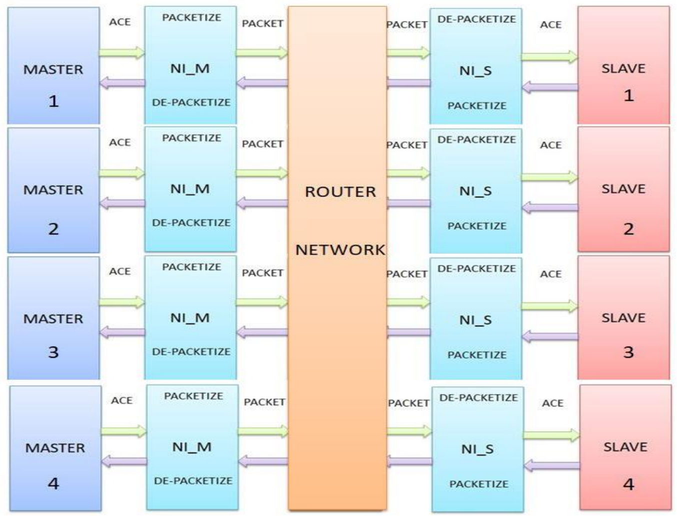


Fig. 5. Overall Multi master – multi slave System Architecture

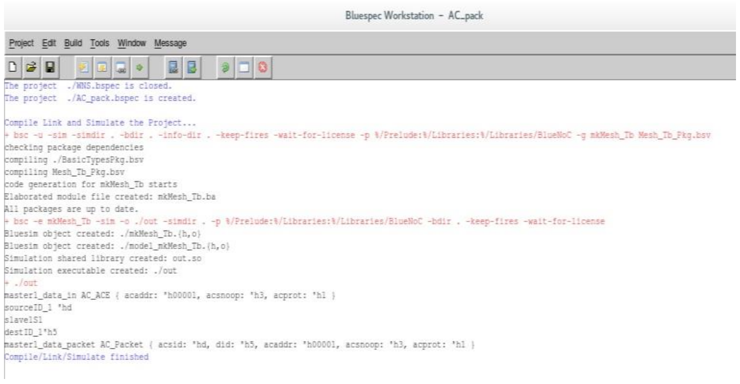
Important fields common for all channels include source ID, Destination ID, Transaction ID. Four bits are allocated for each field. A single transaction includes usage of multiple channels and each packet has common critical fields.

- Source ID refers to the unique ID of the home router. This detail is explicitly available at the router.
- Destination ID refers to the router of target slave and is obtained from the destination address listed as a part of ACE signal.
- Transaction ID refers to unique ID allocated by the master for each transaction and is a part of original ACE signal.
- Additional signals for it formation include BOP (Beginning of Packet) as well as EOP (End of packet).

Fig.4 shows overall architecture for single master- single slave design. Fig.5 shows overall architecture for multi master – multi slave design. Clearly the important components include network interface unit as well as router network unit.

V. RESULTS

The micro router module, network interface module as well as snoop filter module have been designed and developed in Bluespec SystemVerilog. Two transactions associated with the protocol have also been verified.


Bluespec Workstation – AC_pack
Project Edit Build Tools Window Message

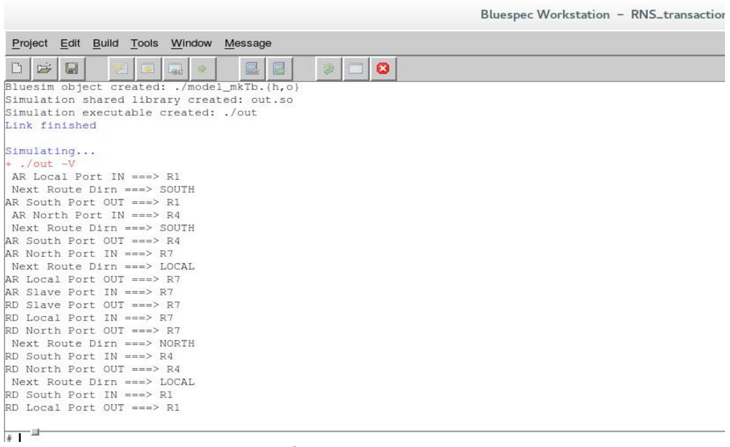
```
The project  ./WNS.bspec is closed.
The project  ./AC_pack.bspec is created.

Compile Link and Simulate the Project...
+ bsc -u -sim -simdir . -bdir . -info-dir . -keep-fires -wait-for-license -p %/Prelude:%/Libraries:%/Libraries/BlueNoC -g mkMesh_Tb Mesh_Tb_Pkg.bsv
checking package dependencies
compiling ./BasicTypesPkg.bsv
compiling Mesh_Tb_Pkg.bsv
code generation for mkMesh_Tb starts
Elaborated module file created: mkMesh_Tb.ba
All packages are up to date.
+ bsc -e mkMesh_Tb -sim -o ./out -simdir . -p %/Prelude:%/Libraries:%/Libraries/BlueNoC -bdir . -keep-fires -wait-for-license
Bluesim object created: ./mkMesh_Tb.{h,o}
Bluesim object created: ./model_mkMesh_Tb.{h,o}
Simulation shared library created: out.so
Simulation executable created: ./out
+ ./out
master1_data_in AC_ACE { acaddr: 'h00001, acsnoop: 'h3, acprot: 'h1 }
sourceID_1 'hd
slave1S1
destID_1'h5
master1_data_packet AC_Packet { acsid: 'hd, did: 'h5, acaddr: 'h00001, acsnoop: 'h3, acprot: 'h1 }
Compile/Link/Simulate finished
```

Fig. 6. Network Interface for AC channel

Fig.6 shows packetization using Network Interface module associated with the router network. Along with the information signals associated with each channel, additional information in the form of source ID and destination ID are calculated and appended to the signal to form required packet. Source ID corresponds to the router ID of the master and destination ID corresponds to the router ID of the slave.

For AC channel, acaddr, acsnoop and acprot are the associated information signals. Required data packet is obtained by combining source ID and destination ID to the signals and is shown in master1_data_packet.


Bluespec Workstation – RNS_transactio
Project Edit Build Tools Window Message

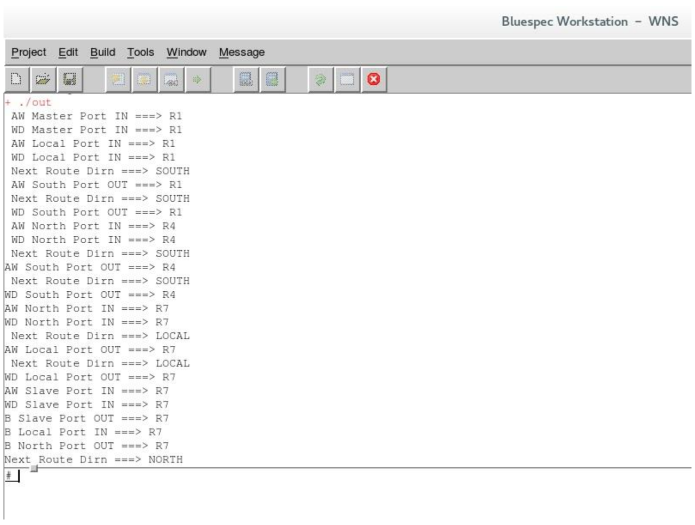
```
Bluesim object created: ./model_mkTb.{h,o}
Simulation shared library created: out.so
Simulation executable created: ./out
Link finished

Simulating...
+ ./out -V
 AR Local Port IN ===> R1
 Next Route Dirn ===> SOUTH
AR South Port OUT ===> R1
 AR North Port IN ===> R4
 Next Route Dirn ===> SOUTH
AR South Port OUT ===> R4
AR North Port IN ===> R7
 Next Route Dirn ===> LOCAL
AR Local Port OUT ===> R7
AR Slave Port IN ===> R7
RD Slave Port OUT ===> R7
RD Local Port IN ===> R7
RD North Port OUT ===> R7
 Next Route Dirn ===> NORTH
RD South Port IN ===> R4
RD North Port OUT ===> R4
 Next Route Dirn ===> LOCAL
RD South Port IN ===> R1
RD Local Port OUT ===> R1

#
```

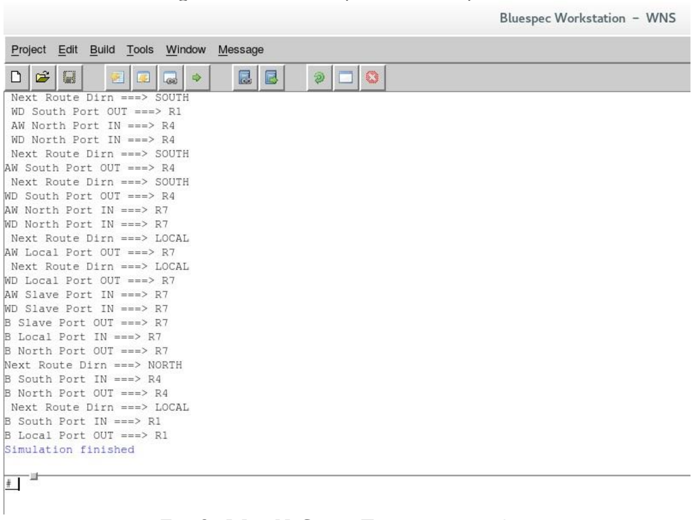
Fig. 7. ReadNoSnoop Transaction

The developed router network consists of 9 routers per channel forming a 3 x 3 mesh network. Fig.7 shows read request sent from master1 to slave1 via Read Address (AR) channel. The home node corresponds to R1 which is connected to master1. After route computation, the next direction is chosen to be south and the data packet moves from local port to south port. This is taken as input by the north input port of router R4. After route computation, the next direction is again found to be south and the data packet moves from north input port to south output port. This is fed to the north input port of router R7 which is connected to required slave S1. After route computation, the data packet moves from north input port to local output port and is fed to the slave node. The requested data response from the slave node reaches master via RD channel. The data packet moves from local port of R7 to north output port of R7. This is fed to south input port of R4 which gets connected to north output port of R4. This is fed to the south input port of R1 which connects to local output port and the data is fed to the requested master node. This completes the ReadNoSnoop transaction.


Bluespec Workstation – WNS
Project Edit Build Tools Window Message

```
+ ./out
 AW Master Port IN ===> R1
 WD Master Port IN ===> R1
 AW Local Port IN ===> R1
 WD Local Port IN ===> R1
 Next Route Dirn ===> SOUTH
 AW South Port OUT ===> R1
 Next Route Dirn ===> SOUTH
 WD South Port OUT ===> R1
 AW North Port IN ===> R4
 WD North Port IN ===> R4
 Next Route Dirn ===> SOUTH
AW South Port OUT ===> R4
 Next Route Dirn ===> SOUTH
WD South Port OUT ===> R4
AW North Port IN ===> R7
WD North Port IN ===> R7
 Next Route Dirn ===> LOCAL
AW Local Port OUT ===> R7
 Next Route Dirn ===> LOCAL
WD Local Port OUT ===> R7
AW Slave Port IN ===> R7
WD Slave Port IN ===> R7
B Slave Port OUT ===> R7
B Local Port IN ===> R7
B North Port OUT ===> R7
Next Route Dirn ===> NORTH
#
```

Fig. 8. WriteNoSnoop Transaction part1


Bluespec Workstation – WNS
Project Edit Build Tools Window Message

```
 Next Route Dirn ===> SOUTH
 WD South Port OUT ===> R1
 AW North Port IN ===> R4
 WD North Port IN ===> R4
 Next Route Dirn ===> SOUTH
AW South Port OUT ===> R4
 Next Route Dirn ===> SOUTH
WD South Port OUT ===> R4
AW North Port IN ===> R7
WD North Port IN ===> R7
 Next Route Dirn ===> LOCAL
AW Local Port OUT ===> R7
 Next Route Dirn ===> LOCAL
WD Local Port OUT ===> R7
AW Slave Port IN ===> R7
WD Slave Port IN ===> R7
B Slave Port OUT ===> R7
B Local Port IN ===> R7
B North Port OUT ===> R7
Next Route Dirn ===> NORTH
B South Port IN ===> R4
B North Port OUT ===> R4
 Next Route Dirn ===> LOCAL
B South Port IN ===> R1
B Local Port OUT ===> R1
Simulation finished
#
```

Fig. 9. WriteNoSnoop Transaction part2

Fig.8 and Fig.9 shows WriteNoSnoop transaction performed on 3 x 3 mesh network. Master1 requests write transaction on AW and WD channel. AW channel carries information regarding the address location to which data is to be written to. The actual data to be written is provided via the WD channel. The master1 is connected to R1 router and the packet moves from R1 local input port to south output port after route computation. This is fed to north input port of the router R4 which connects to south output port. This is connected to north input port of router R7 which is the home node of slave1. The packet then reaches slave via local output port. The response from the slave showing successful write transaction is conveyed via B channel. The response from R7 reaches R1 via R4 and is fed to the local output port and gets received by the requested master. This completes the WriteNoSnoop transaction.

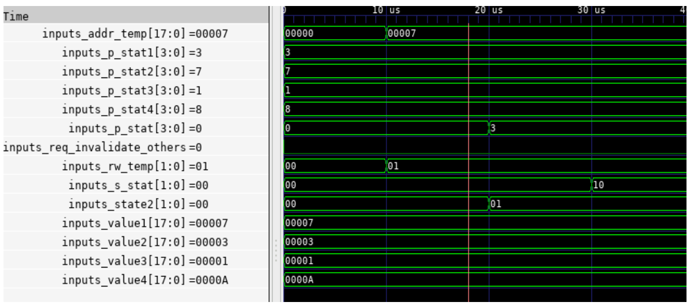


Fig. 10. Snoop Filter Read Operation

Snoop filter is essentially designed to ensure cache coherency in multi master systems. The current model is capable of storing array of information with cache line address and status of each of the masters as the various fields associated with it. For representation purpose, Fig.10 shows four possible cache line values and the associated master status. Signal inputs_valuei refers to the cache line address at location i. Signal inputs_p_stati refers to the status of the masters names M1,M2,M3,M4 for the corresponding cache line address. If status is 1, this signifies presence of valid data in the corresponding cache of the master. Based on the requested cache line address, the snoop filter checks the status of other masters.

In case of read operation, if any of the master has valid copy of the cache line, it is used to give response to the requested master. This saves time from accessing the slave and providing the response. For given example, requested cache line address is 00007 which correspond to the first data in the array. The corresponding status of the masters is 3 which mean valid data is present in M3 and M4. Signal inputs_s_stat is used to signify whether data needs to be taken from slave. This gets high only when none of the masters have a valid cache line. Signal inputs_rw_temp is used to identify whether it is a write request or a read request. Signal inputs_req_invalidate_others is used to suggest whether the data in other masters need to be invalidated after responding to the request. This needs to be set high after a write request.

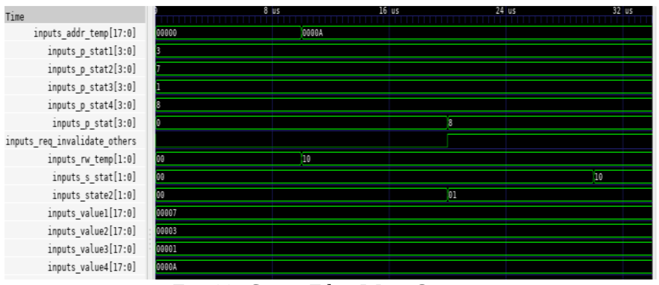


Fig. 11. Snoop Filter Write Operation

Fig.11 shows write operation being handled by the snoop filter. The requested cache line address is 0000A which corresponds to fourth value in the array. The corresponding master status is 8. Only Master1 has valid copy of the data. Since this is a write operation, the data in other masters need to be invalidated after successful transaction. Thus the signal inputs_req_invalidate_others is set to high. The data needs to be updated to the slave and hence signal inputs_rw_temp has value 10.

VI. CONCLUSION

Cache coherent interconnect is important while designing a multi-processor system. Cache coherency can be achieved using hardware or software based. Hardware based cache coherency is achieved using design of cache coherent interconnect. The proposed project involves designing cache coherent interconnect using mesh topology, store and forward flow control with dimension ordered routing algorithm. The proposed work involves realization of multiple channels using independent 2D mesh topology based on ACE protocol specification. The work is implemented using Bluespec SystemVerilog language. Micro router module, network interface module, snoop filter module, 3 x 3 network for each channel have been designed and verified. Two transactions namely ReadNoSnoop and WriteNoSnoop have been verified.

REFERENCES

[1] David Wentzla_, Patrick Gri_n, Henry Ho_mann, "On-Chip Interconnection Architecture of the Tile Processor" ,IEEE MICRO ,VOL.9,NO.1,SEPTEMBER 2007.

[2] Bhavya K. Daya, Chia-Hsin Owen Chen, Suvinay Subramanian, Woo-Cheol Kwon, Sunghyun Park, Tushar Krishna, Jim Holt, Anantha P. Chandrakasan, Li-Shiuan Peh, "SCORPIO: A 36-Core Research Chip Demonstrating Snoopy Coherence on a Scalable Mesh NoC with In-Network Ordering " ,IEEE, VOL.4, NO.4, APRIL 2014.

[3] ARM,"AMBA AXI and ACE: Protocol Specification ", Protocol Specification, MARCH 2011.

[4] Giorgos Dimitrakopoulos, Psarras, "Microarchitecture of Network on Chip Routers ",Springer, FEBRUARY 2015 .

[5] Tobias Bjerregaard and Shankar Mahadevan, " A survey of research and practices of network on chip", ACMCOMPUTER SURVEYS, VOL.4, NO.4, MAY 2006.

[6] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny, " QNoC: QoS architecture and design processor for cost-effective network on chip ",IEEE, VOL.2, NO.3,FEBRUARY 2004.

[7] N. Genko,D.Atienza, G. De Micheli, J. Mendias,R. Hermida, and F. Catthoor, " A complete network-on-chip emulation framework " , ACM COMPUTER SURVEYS, VOL.5, MAY 2005.

[8] Christopher J. Glass and Lionel M. Ni, " The turn model for adaptive routing" , Springer, VOL6, JUNE 2006.

[9] Nitin Godiwala, Jud Leonard, and Matthew Reilly, " A network fabric for scalable multiprocessor systems", IEEE MICRO, VOL.5, 2008.

[10] WaiHongHo andTimothy Mark Pinkston" A design methodology for efficient on-chip interconnects" ,IEEE Transactions on Parallel and Distributed Systems, FEBRUARY 2006.

[11] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar, " A 5- GHz mesh interconnect for a Teraflops processor " , IEEE MICRO, 27(5):51–61, 2007.