International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-4, February 2019

Designing Network Interface Component for Peripheral IP cores in Networks-on-chip

Kulkarni Rashmi Manik, S Arulselvi, B Karthik

Abstract*: The Network-Interface-Componant (NIC) is required for IP cores for interconnecting IPs to Routers in NoC. In implementation of NoC, Interface Component is very crucial for adapting IPs in NoC. NIC as a software component occupies processor's considerable execution time. Processor can be relieved from this overload by introducing separate hardware as NIC. A hierarchical topology for NoC is considered in this research article. In hierarchical topology, each router can connect to eight nodes (IP) of same hierarchy and to a router in next hierarchy. Each node is connected to router port with NIC. The fixed address based routing is implemented in the NOC. The network packet switching based transactions among various nodes is assumed. The implementation of NIC design with options for different IPs (considering existing bus based interfaces) is attempted in this work.

***Index Terms*: NoC, NIC, IPs, PE, ASIC and NS/CS.**

I. INTRODUCTION

A basic NoC architecture is composed of routers, communication links between routers, and a Network-Interface Component (NIC) between a router and a node having either processing element (PE) IP or Peripheral IP core [21]. The NIC offers packatization / depacketization services for IPs. The IP Cores are available with standard on-chip bus interfaces like AXI, OCP, . Etc. These IP cores need to be interfaced with Router in NoC in complex Application Specific System On Chips. Present SoCs integrates different IP cores like, processor element (PE) cores, special function cores or accelerators(Graphics, Video, AES, etc), memory controllers (SRAM, DDR SDRAM, FLASH) and communication controllers (UART, Ethernet, USB, I2C, I2S, SPI, PCIe etc). The number of PE cores may be more than one and may of different type depend on the targeted performance of an application. The PE cores initiate the transaction with other IP cores to achieve the goals of the application. Each IP is connected in Network-on-chip via Network-Interface-Component. The topology of NOC is given in figure 1. In this research article, the section II is comprehensive literature survey of related work. Proposed methodology is formulated in section III. Section IV is about NIC modelling and result analysis. Section V outlines conclusion of the research work. Section VI is acknowledgement, section VII is a list of references and section VIII is about authors.

Manuscript published on 28 February 2019.
*Correspondence Author(s)
Kulkarni Rashmi Manik, Research Scholar/ECE, Bharath Institute of Higher Education and Research, Chennai, India.
S Arulselvi, Associate Professor, Department of ECE, Bharath Institute of Higher Education and Research, Chennai, India.
B Karthik, Associate Professor, Department of ECE, Bharath Institute of Higher Education and Research, Chennai, India.

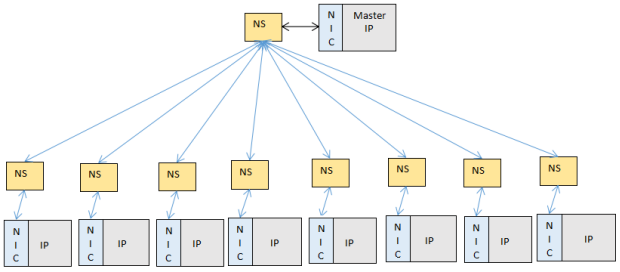

Fig.1. Hierarchical Topology Connecting 8 cores

II. LITERATURE REVIEW

Brahim Attia and co-authors proposed a"Low Latency and Power ASIC Design of Modular Network Interfaces for Network on Chip." The low latency and jitter reduction between successive packets are obtained by a separation between header and payload memories. The low power is obtained through the implementation of these NIs using low power library with 130 nm technology. [1] Masoud Daneshtalab and co-authors proposed "CARS: Congestion-Aware Request Scheduler for Network Interfaces in NoC-based many core Systems". The idea is to use the global congestion information as a metric in network interfaces to reduce the congestion level of highly congested areas.[2] Wang Jianand co-authors proposed"Design of network adapter compatible OCP for high-throughput NOC". In this paper, a network adapter compatible with OCP interface protocol is designed to enable the integration of IP cores from different providers into an on-chip interconnection network.[3] Azad Fakhari proposed "Designing Customizable Network-on-Chip with support for Embedded Private Memory for Multi-Processor System-on-Chips". This research proposes a custom-designed NoC specifically for MPSoCs on FPGAs. The proposed design allows the communication infrastructure to seamlessly scale as the numbers of processors within the chip increases. The design adds a new level of abstraction to remote-access transactions. The design also considers support for the partitioned global address space model with support for optional embedded local memories embedded in the network interface.[4] Masoumeh Ebrahimi and co-authors proposed "Efficient Network Interface Architecture for Network-on-Chips". It is anovel network interface architecture for on-chip networks to increase memory parallelism and to improve the resource utilization. The proposed architecture exploits AXI transaction based protocol to be compatible with existing IP cores.[5] Leandro Fiorin and co-author proposed, "Fault-Tolerant Network Interfaces for Networks-on-Chip".

In this work, authors propose a functional fault model for the NI components by evaluating their susceptibility to faults. They present a two-level fault-tolerant solution that can be employed for mitigating the effects of both permanent and temporary faults in the NI.[6] Tung Nguyeny and co-authors proposed "High-Performance Adaption of ARM Processors into Network-on-Chip Architectures". This paper presents an efficient AXI (Advanced eXtensible Interface) compliant network adapter for 2D mesh Wormhole-based NoC architectures, named AXI-NoC adapter. [7] Rachid Dafali and co-authors proposed "Self-Adaptive Network-on-Chip Interface". This paper presents an original approach of bandwidth-oriented self-adaptively in the domain of Network-on-Chip, where reconfiguration is handled by network interfaces offering traffic with guarantee of service. [8] Ahmed H.M. Soliman and co-authors proposed "Designing a WISHBONE Protocol Network Adapter for an Asynchronous Network-on-Chip". In this paper, an Asynchronous NoC has been implemented on a SPARTAN-3E® device. The NoC supports basic transactions of both widely used on-chip interconnection standards, the Open Core Protocol (OCP) and the WISHBONE Protocol. [9] Vijaykumar R Urkude and co-author proposed, "Low Power 2-D Mesh Network-on-Chip Router using Clock Gating Techniques". Clock gating is high-level technique for decreasing the power consumption of a design. For designs, that have large multi-bit registers, clock gating can save power and reduce the number of gates in the design.[10]Alexandros Daglis and co-authors proposed "Many core Network Interfaces for In-Memory Rack-Scale Computing". This paper proposes and evaluates network interface architectures for tiled many core SoCs for in-memory rack-scale computing.[11] Marcelo Ruaro and co-authors proposed "DMNI: A Specialized Network Interface for NoC based MPSoCs". This paper presents a specialized communication interface for NoC-based MPSoCs, called DMNI(Direct Memory Network Interface). To avoid stalls in the communication, the design of the DMNI supports ssimultaneous packet reception and transmission for software layer. Results show a reduction in the silicon area and performance improvement in the packet transmission[12] Ruxandra Pop and co-author proposed, "A Survey of Techniques for Mapping and Scheduling Applications to Network on Chip Systems". In this report, author has survey the techniques for mapping and scheduling concurrent applications to NoC platforms.[13] Brahim Attia and co-authors proposed "Design and implementation of network interface compatible OCP For packet based NOC". It is the idea of using on chip packet switched networks for Interconnecting a large number of IP cores is very practical for designing complex SoCs since it gives possibility of not only reusing IP cores but also the interconnection infrastructure.[14] Sujay Gejji and co-author proposed "Designofre-configurableandmodularNOCinterfacewithadvancednetworking functionalities".In this paper, the design of re-configurable and modular NoC Interface (NI) with the advanced networking functionalities has been presented.[15] Jens Sparsproposed, "Design of Networks-on-Chip for Real-Time Multi-Processor Systems-on-Chip". This paper discusses ongoing research conducted as part of the project "Time-predictable Multi-Core Architecture for Embedded Systems" (T-CREST), supported by the European Commissions seventh framework programme. The aim of this project is to develop a general-purpose multi-core platform for real-time systems as well as tools supporting its use (compiler, simulator, and worst-case execution time analysis tool).[16] Nauman Jaliland co-authors proposed "Routing Algorithms, Process Model for Quality of Services (QoS) and Architectures forTwo-Dimensional4 x 4 Mesh Topology Network-on-Chip". In this paper, authors have provided routing algorithms, process model for quality of service (QoS) and architecture for new Timer based adaptive routing algorithm for a generic network, based on a two-dimensional mesh topology. [17] Ryuya Okadaand co-author proposed, "Design of Core Network Interface for Distributed Routing in OASISNoC". Here, authors proposed an architecture and a design of a Core Network Interface (NI) for distributed routing on an Altera FPGA board. The designed NI occupies less than 1%of area utilization of Cyclone II FPGA. [18] Calin Ciordas and co-authors proposed, "Transaction Monitoring in Networks on Chip: The On-Chip Run-Time Perspective". Authors show the versatility of their NoC analyzer by run-time monitoring user connections and the Master IP in the NoC.[19] Chenxin Zhangand co-author presented, "A presentation on Network-on-Chip (NoC)" [20] P. Gratz and co-authors proposed "Implementation and evaluation of on-chip network architectures". [21] Glovanni De Michel, Luca Benini gives complete idea about concept "Networks on chips". [22]

Many research studies are focused on implementation of FIFO buffer architectures [23]. The studies on network architectures for accelerator-rich architectures are covering memory latency and communication latency issue [24]. Other studies on router architectures focus on buffer technique [25], and dynamic re-configurable NoC architectures[26], layered switching techniques[27].

The design and implementation of NIC is explored in this work and packet formats are worked out for interfacing peripheral cores.

III. PROPOSED METHODOLOGY

To realize the NoC for complex SoCs, the NIC design is carried out in step by step starting from analysis of communication transactions to design of formats for Packet and Flit. The different variants of NIC are identified and designed for Processor core, Peripherals and Memory Controller.

A. Transaction Identification in SoC

The PE generally have following types of transactions:-

- ✓ Memory Access Request
- ✓ Cache Coherence
- ✓ Data Fetch
- ✓ Data Update
- ✓ IO Access and Interrupt

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

From peripheral perspective, NIC may handle following type of packets as: -

- ✓ Register Read(Control/Status) Packet (RRP)
- ✓ Register Write(Control) packet (RWP)
- ✓ Data Buffer Read packet (DBRP)
- ✓ Data Buffer Write packet (DBWP)
- ✓ Memory Data Receive packet (MDRP)
- ✓ Memory Read packet (MRP)
- ✓ Memory Write packet (MWP)
- ✓ Register Data Out packet (RDOP)
- ✓ Data Buffer Out packet (DBOP)
- ✓ Interrupt packet(InP)

The communication peripherals (UART, SPI, I2C) may not need Memory Read, Memory Data Receive and Memory Write. Out of these 10 types of packet, the first 5 are packets coming from Router to NIC and other five packets are generated by NIC to Router. The complex cores, which traditionally need direct memory accesses for large data transfer need NIC support for Memory Read packet and Memory Write packets (e.g. USB, Ethernet, PCIe).

B. Transaction Analysis

Analysis of the transaction types of Processor cores (PE), Peripheral IP (PIP) cores and Memory Controller Core for designing NIC for each of them is formulated here. Three types of NIC are required for implementing complex SoCs namely C-NIC for processor cores, P-NIC for peripheral cores and M-NIC for memory controllers.

1. Transaction analysis for C-NIC:

The Processor cores required to interact with the other cores and these interactions generate the communication in the NoC. The processor transactions are mainly Memory Read, Memory Write, Peripheral Read and Peripheral Write and Interrupt Receive. In addition, in muti-processor SoC, processor cores also receive cache-invalidation transaction request from other processors. The C-NIC is responsible for packetizing the outgoing transactions and de-packetizing incoming transactions for processor.

2. Transaction analysis for P-NIC

The PIP cores are designed with set of Control Registers (CR) to configure different modes or functions of operation. The PIPs are designed with set of Status Registers (SR) to hold the status of function being performed. The PIPs may have set of Data Registers (DRs), may be implemented as Registers/Memories/FiFOs) for holding set of data to be transmitted or holding a set of received data. The existing IPs are designed with bus based interface for accessing these internal registers using predefined address range. In addition if IPs have capability to become bus master, the PIP will have additional control/status registers (CRs) for supporting direct memory access (DMA). The NIC implementation for PIPs with DMA need to have additional functional support for generating Memory Read packet and Memory Write packet. In case of PIPs with DMA capability a control register is required to be implemented in NIC for holding destination node address of required memory controller. The NIC for PIP should generate the Interrupt Receive Packets if interrupt signal is generated by the corresponding PIP. If only one PE exists in SoC, the Interrupt packet has only one fixed destination node address. In case of SoC having multiple PEs, the NIC should have control register for configuring destination node address in Interrupt Receive Packet. The P-NIC is responsible for packetizing the outgoing transactions and de-packetizing incoming transactions for PIP.

3. Transaction analysis for M-NIC

The Memory Controllers in SoC provide read and write access to on chip memory to processors and PIPs. The Memory Controller in SoC has to respond to Memory Read transactions with the memory data and to complete the Memory Write transactions on Memory. The M-NIC translates the received packets from router to memory read/writes accesses and injects packets to router with data read from memory. With detailed analysis of transactions, signal interfaces are identified and related analysis is given in the following section.

C. Signal Interface Analysis:

A 32-bit Processor Core having Address Bus, Data Bus and Control Bus is considered here. The C-NIC signal interface is shown in figure 2. The C-NIC has processor bus interface on one side and LINK (IN and OUT) interface with router to send the packets. The processor core generic signals are AddrOut [31:0], DataIn[31:0], DataOut[31:0], RdOut, WrOut, ByteEnable[3:0], SoT (Start of Transaction), EoT (End of Transaction) and IRQ[7:0].

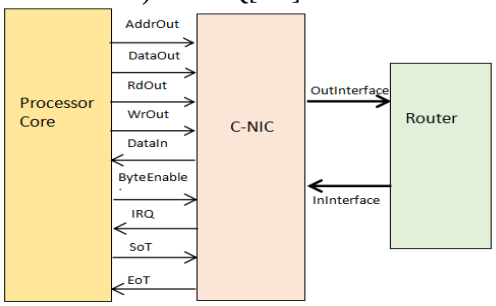

Fig.2. C-NIC Interface Diagram

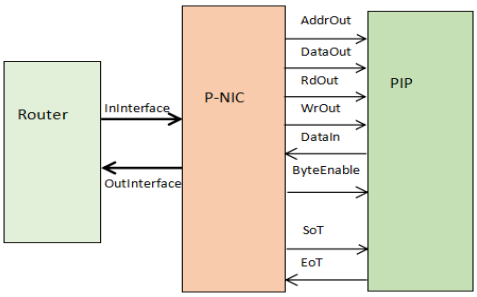

Fig.3. P-NIC Signal Interface

The PIP is interfaced to Router through P-NIC.

The figure 3 shows the interface signals. The PIP cores need AddressOut, DataIn[31:0], DataOut[31:0], RdOut, WrOut, ByteEnable, SoT, EoT. For PIP having direct memory access feature, AddressIP[31:0] and ByteEnable_IP[3:0] are additional signals from PIP to P-NIC. The packets received from Router are processed by a control state machine in P-NIC.

The Memory Controller is interfaced to Router through M-NIC. The transactions with memory controller are Memory Read access and Memory Write access.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The packets received from Router are converted into signals required to complete the transactions in M-NIC. The data read from memory is sent to Router in the packet format by M-NIC. Following figure 4 shows the interface signals for M-NIC. The signals are AddressOut, DataIn[31:0], DataOut[31:0], RdOut, WrOut, ByteEnable[3:0].

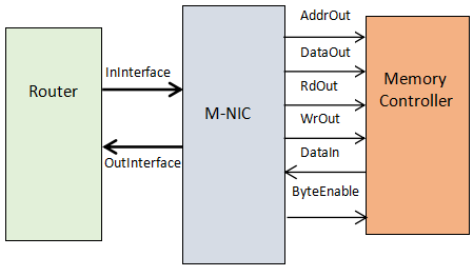

Fig.4. M-NIC Signal Interface

A complete interfacing of the processor with memory controller and peripheral IP cores is shown in figure 5.

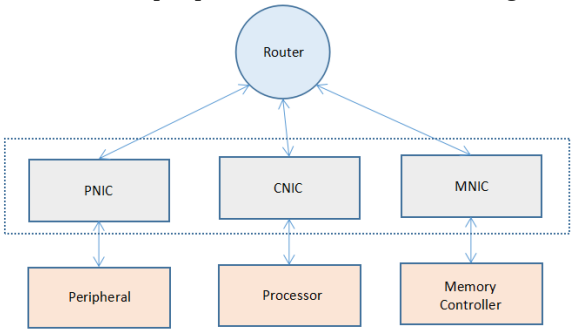

Fig.5. NIC as a combination of PNIC, CNIC and MNIC

D. Packet Format Design:

The various fields of the packet are as follows. Source Node Address(SNA) and Destination Node Address(DNA) represented as 6-bits for 64 nodes (arranged as 8 X 8), 8 bit for 256 nodes (arranged as 16 X 16) and 10 bits for 1024 nodes (arranged as 32 X 32). Packet type(PT) represented as 6 bits even though 4 bits are sufficient as different packet types are only 13. The Packet Length is represented as 8 bits and Byte Enable(BE) represented as 4 bits for 32-bit transfer. Packet Address Value (PAV) represented as 32 bits and Packet Data Value (PDV) represented as 32 bits. In the case of the burst data transfer the Packet Data Value may be 128 bits. Various Packet types and formats designed are listed below:-

1. Register Read Packet (RRP)

This packet is generated by C-NIC and for purpose of informing P-NIC to respond with contents of a specific register. The packet format has following fields {PL, SNA, DNA, PT, BE and PAV}. The P-NIC after receiving the packet from Router performs read cycle on PIP and send RDOP to Router. The source node address and destination nod address fields are reversed from RRP.

2. Register Write Packet (RWP)

This packet is generated by C-NIC for purpose of writing register of the PIP. The packet format has following fields {PL, SNA, DNA, PT, BE, PAV and PDV}. The P-NIC after receiving the packet from Router performs write cycle on PIP.

3. Data Buffer Read Packet (DBRP)

This packet is generated by C-NIC and for purpose of informing P-NIC to respond with contents of data buffer register. The packet format is same as RRP. The P-NIC after receiving the packet from Router performs read cycle on PIP and send DBOP to Router. The source node address and destination nod address fields are reversed from DBRP.

4. Data Buffer Write Packet (DBWP)

This packet is generated by C-NIC for purpose of writing data buffer register of PIP. The packet format is same as RWP. The P-NIC after receiving the packet from Router performs write cycle on PIP.

5. Memory Data Receive Packet (MDRP)

This packet is generated by M-NIC node for purpose of writing data buffer register of PIP, in response to Memory Read packet send by P-NIC. The packet format is same as RWP. The P-NIC after receiving the packet from Router performs write cycle on PIP.

6. Memory Read Packet (MRP)

This packet is generated by P=NIC to Router to access the memory controller in NOC for DMA read requested by PIP. The packet format is {SNA, DNA, PT, BE and PAV}. The DNA field needs to be programmed in the P-NIC control register. The PIP control registers provides PAV to NIC.

7. Memory Write packet (MWP)

This packet is generated by P-NIC to Router to access the memory controller in NOC for DMA write requested by PIP. The packet format is {PL, SNA, DNA, PT, BE, PAV and PDV}.

8. Register Data Out Packet (RDOP)

This packet is generated by P-NIC to Router in response to received RRP. The packet format is {SNA, DNA, PT, BE and PDV}. The SNA and DNA fields from RRP are interchanged in this packet.

9. Data Buffer Out Packet (DBOP)

This packet is generated by P-NIC to Router in response to received DBRP. The packet format is same as RDOP. The SNA and DNA fields from DBRP are interchanged in this packet.

10. Interrupt Receive Packet (IRP)

This packet is generated by NIC to Router in response of interrupt generated by PIP. The packet format is {PL, SNA, DNA and PT}.

11. NIC Register Write packet (NRWP)

This packet is generated by C-NIC to program control registers for DNA fields required for MRD, MWR and IRP. This packet is consumed by P-NIC and no read/write cycle is generated for PIP. The packet format is similar to RWP.

12. NIC Register Read Packet (NRRP)

This packet is generated by C-NIC to read control registers values in P-NIC. This packet is consumed by P-NIC and no read/write cycle is generated for PIP. The packet format is similar to RRP.

13. NIC Register Out Packet (NROP)

This packet is generated by P-NIC in response to NRRP received from PE node. The packet format is {PL, SNA, DNA, PT, BE and PDV}. The SNA and DNA fields from NRRP are interchanged in this packet.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

E. Packet Size and Flit Size Estimation:

Every packet has SNA and DNA fields and the number of bits required for SNA and DNA depends on the size of the network and number of nodes in the network. In the case of NoC having 64 nodes, 6 bits are required for SNA and DNA fields. Similarly for NoC having 256 nodes, 8 bits are required for SNA and DNA fields, and so on. As we have defined 13 types of packets, 4 bits are sufficient for Packet Type. If we consider PAV and PDV fields of 32 bits and BE field as 4 bit, the packet sizes for all types of packet are given below table 1.

Table.1 Packet sizes for all types of packet

Packet Type	Packet format	Size in bits	
		64 nodes	256 nodes
RRP, DBRP, MRP, NRRP	{PL,SNA, DNA,PT,BE,PAV}	60	64
RWP, DBWP, MDRP, MWP, NRWP	{PL,SNA,DNA, PT,BE,PAV, PDV}	92	96
RDOP, DBOP, NROP	{PL,SNA, DNA,PT,BE,PDV}	60	64
IRP	{PL,SNA,DNA, PT}	24	28

The packets are transmitted in NoC as a set of flits. The flit width size is chosen as 34 bits, then packets of type (RWP, DBWP, MDRP,MWP, NRWP) need three flits, packets of type (RRP, DBRP, MRP, NRRP, RDOP, DBOP,NROP need two flits and IRP needs just one flit. The flit is formed with flit identifier (2 bit) and 32 bits of payload from packets. When we need three flits for packet, first flit has flit is formed as {00, first 32 bits of packet}, second flit is formed as {01,next 30 bits of packet} and last flit is formed as{11,remaining bits of packet, zero padding}. When we need two flits for packet the first flit is formed as {11, first 32 bits from packet}, second flit is formed as {11,remaining bits of packet,zero padding}. When one flit is required for packet flit is formed as {10, packet bits, zero padding}

F. Algorithm For Hardware Design of NIC

Even though NIC provides interface of cores to Routers, the algorithm complexity for handling packets between cores and routers varies for P-NIC, C-NIC and M-NIC. The algorithm implemented in each NIC is explained below:-

1. Algorithm for C-NIC

The C-NIC has look-up table for identifying destination node address. The request from processor is classified depending on the values on interface signals from Processor Core. The algorithm for operation is given below

Step 1: Is Start Signal received? Then go to Step 4

Step 2: Is Interrupt Packet Received from Router? Then go to Step 8

Step 3: Classify the access and destination node

Step 4: Generate appropriate packet and send to Router

Step 5: If response packet is expected go to step 6 else assert over signal and go to Step1

Step 6: Wait for response packet from Router

Step 7: Assert DataIn and Over signal and go to Step 1

Step 8:Decode the Interrupt Packet and Assert appropriate IRQ signal and go to Step 1

2. Algorithm for P-NIC

The P-NIC initiates action based on packets received from Router or interrupt generated by the peripherals. The P-NIC has a set of control registers for supporting configurable parameters for packets. The algorithm for operation is given below

Step 1: Check packet ready from router and interrupt signal from PIP, if no packet or interrupt wait here.

Step 2: If Interrupt received from PIP, go to Step7

Step 3:If packet received from Router, classify the packet

Step 4: If read type of packet then read register or data buffer of PIP and send response packet to Router go to step1

Step 5: If write type packet then complete write register or data buffer on PIP and go to step 1

Step 6:If memory access is required send memory read/write packet to Router and go to step 1

Step 7: Send interrupt packet to Router and go to step 1

3. Algorithm for M-NIC

Memory Controller connected to NIC is simplest one in design complexity as it supports memory read and memory write access received from Router. The algorithm for operation is given below

Step 1: Check packet from Router else wait here

Step 2: Decode packet, for write type of packet assert write signals to Memory Controller and go to Step 1

Step 3: For read type of packet, assert read signals to Memory Controller

Step 4: Generate packet after receiving data from Memory Controller

Step 5: Send the packet to Router and go to Step 1.

G. NIC State Machine Design

The NIC hasone side interface with Router for receiving packets and other side interface with core (Processor or PIP or Memory Controller) completing access. A block diagram common to C-NIC, P-NIC and M-NIC is shown in figure 5. The Core Interface Block (CIB) has all signal interface specific to Core connected. Two FIFOs are required for interfacing with router, one for outgoing packet Out FIFO (OFF) and other for incoming packet In FIFO (IFF). The Link Control Block generates handshake signals between Router and NIC for packet transfers. The The Control State Machine is designed to regulate the transaction between core and Router. The Control State Machine (CSM) implemented in C-NIC is shown in figure 6. Similar Control State Machine is implemented for P-NIC and M-NIC with required states.

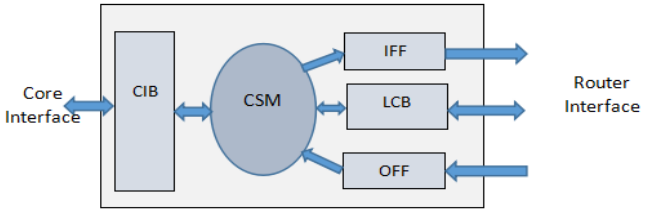

Fig. 5 NIC Block diagram

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Initially Control State Machine is in state ST0 and waits for either IRQ packet arrival from Router or access request from Processor Core. On IRQ packet arrival, the state changes to ST1, decodes packet for the interrupt number. It generates interrupt to processor and goes back to initial state through safe transit state ST4. On read request from Processor Core, the CSM goes to ST3 and sends appropriate packet to Router. In state ST6 it waits for response packet from Router and then goes back to initial state ST0. On write request from Processor Core, state changes to ST2, sends appropriate packet to Router. Then it returns to initial state ST0 through safe state ST5.

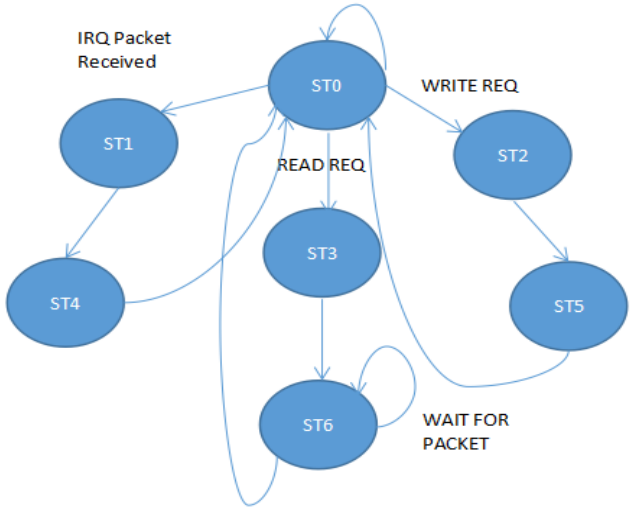

Figure 6. State Diagram for CSM

H. Detailed Action Steps of Transaction

All transactions in typical Processor based SoC can be analyzed considering proposed NIC implementation. In test example considered for SoC implementing hierarchical NoC connecting total up to 64 cores. All possible transactions and complete flow of data in SoC is explained as follows:-

1. Control/Status register read by Processor Core

The C-NIC connected Processor sends RRP to P-NIC of respective PIP. The P-NIC state machine reads the required register from peripheral IP and dispatches response RDOP to Router with appropriate destination address.

2. Data Buffer read by Processor

The C-NIC connected to Processor Core sends DBRP to P-NIC of respective PIP. The packets travels through hierarchy of Routers to reach up to P-NIC. The P-NIC state machine reads the required data from PIP and dispatches response DBOP to Router with appropriate destination address.

3. Control register write by Processor

The C-NIC of Processor Core sends WRP to P-NIC of PIP. The P-NIC in turn completes register writing of PIP.

4. Data Buffer write by Processor

The C-NIC of Processor send DBWP to P-NIC of PIP. The NIC state machines in turn complete Data Buffer writing of PIP.

5. DMA (Memory to Peripheral)

If Peripheral IP has DMA capability then it interacts with P-NIC for memory transfers. The P-NIC has the control register holding destination node address for memory reads initiated by PIP. The P-NIC converts the memory read request of PIP to packet for read memory with appropriate destination address of memory controller node. The M-NIC receives packets and initiates read cycle on Memory Controller. The M-NIC sends response packet MDRP to P-NIC of PIP. The P-NIC in turn writes to data buffer in PIP.

6. DMA (Peripheral to Memory)

The PIP write the data from its data buffer to P-NIC. The P-NIC generates MWP with destination node address of Memory controller. The C-NIC receives the packets and initiates write cycle on Memory Controller.

7. NIC read/write register

The Processor Core configures registers in P-NIC for DMA transactions.

IV. NIC IMPLEMENTATION AND RESULT ANALYSIS

Three modules have been developed in Verilog one for each C-NIC, P-NIC and M-NIC. The Verilog code has been synthesized for Xilinx Virtx-5 FPGA for prototype implementation. The resource utilization in terms of Look-up-tables (LUT) is observed as very less compares to that of Routers (less than 8%). The operating frequency achieved is 356 MHz for C-NIC and 350 MHz for P-NIC and 370 MHz for M-NIC. For performance analysis, abstract functional model of Hierarchical Router having 9 ports and abstract bus functional models of Processor, Memory Controller and Peripherals are used. With interconnecting Eight Hierarchical Routers over all 64 different NIC are interconnected with these abstract models.

Following cases were simulated;

A. Case 1: SoC-A

Four Processor Core with C-NIC, Eight Memory Controller with M-NIC and 48 peripheral cores with P-NIC based SoC. In simulations for each Peripheral and Memory 10 transactions were generated from Processor Core. The overall time required in terms of cycles for completing all the accesses through C-NIC, P-NIC and M-NIC are recorded. Here 10700 cycle are required to complete all transactions

B. Case 2: SoC-B

Eight Processor cores with C-NIC, Eight Memory Controllers with M-NIC annd 48 peripherals cores with P-NIC based SoC. Again equal number of transactions were generated. Here since eight processor cores were used for transactions, the less number of cycles were observed for completing all transactions. Considering ideal situation of no simultaneous access of peripherals by processor cycles required to complete the transactions are 5400 cycles.

For comparing results of proposed model with earlier reported work, uniform traffic pattern considered for case of SoC-B. The proposed model shows latency reduction when request rate is increased as shown in figure 7.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

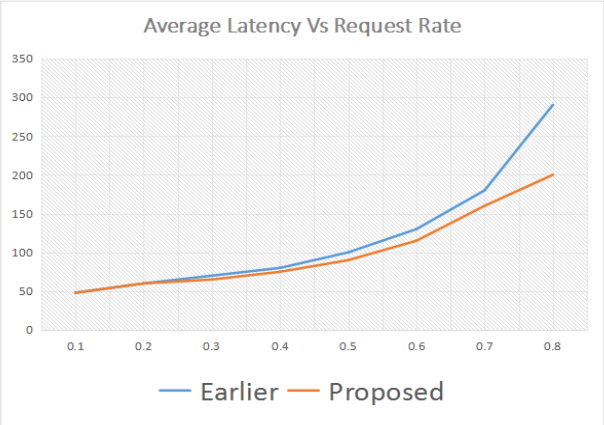

Figure 7. Latency comparison with earlier work

The latency comparison of case SoC-A and case SoC-B is shown in figure 8.

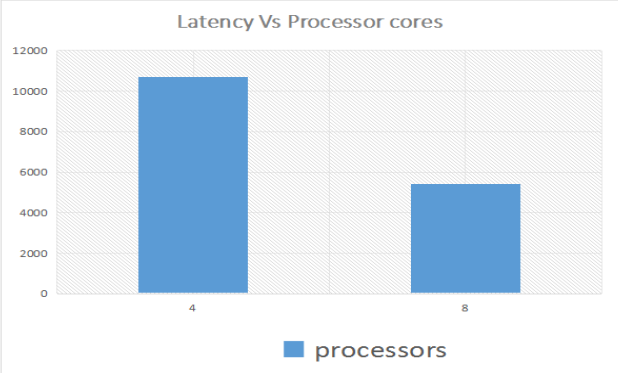

Figure 8. Latency comparison with processor cores

V. CONCLUSION

The NIC implementation for adapting different IP cores to NOC based SoC is proposed here. A generic processor interface is considered in this study. Our approach propose three types of simplified NIC for SoC. A separate NIC design for Peripheral P-NIC and Memory Controller M-NIC is implemented rather than a common slave NIC approach suggested in earlier works. The P-NIC having DMA support reduces the latency of packets as P-NIC accesses the memory directly and no need for processor to generate additional packets. There is overall improvement in communication latency with simple and fast NIC implementation. The embedded SoCs integrating large number of on-chip Peripherals are the target applications of the proposed NIC.

ACKNOWLEDGMENT

We would like to thank Bharath University for encouraging us to write this research article. We thank Mr. Anil Terkar, Scientist, D.R.D.O. for his valuable help and guidance.

REFERENCES

1. Brahim Attia, Abdelkrim Zitouni, Kholdoun Torki and Rached Tourki "A Low Latency and Power ASIC Design of ModularNetwork Interfaces for Network on Chip", IJCSES International Journal of Computer Sciences and Engineering Systems, Vol. 5, No. 4, October 2011.
2. Masoud Daneshtalab, Masoumeh Ebrahimi, Juha Plosila, Hannu Tenhunen, "CARS: Congestion-Aware Request Scheduler forNetwork Interfaces in NoC-based Manycore Systems".
3. Wang Jian, Yang Zhijia,"Design of network adapter compatible OCP for high-throughput NOC", Applied Mechanics and Materials Vols. 313-314 pp 1341-1346, Trans Tech Publications, Switzerland, 25 March 2013.
4. Azad Fakhari, "Designing Customizable Network-on-Chip withsupport for Embedded Private Memory for Multi-Processor System-on-Chips", Thesesand Dissertations,University of Arkansas, Fayetteville, May 2014
5. Masoumeh Ebrahimi, Masoud Daneshtalab, N P Sreejesh, Pasi Liljeberg, Hannu Tenhunen, "Efficient Network Interface Architecture forNetwork-on-Chips", Department of Information Technology, University of Turku, Turku, Finland.
6. Leandro Fiorin, Mariagiovanna Sami, "Fault-Tolerant NetworkInterfaces for Networks-on-Chip", IEEE Transactionson Dependableand Secure Computing, VOL. 11, NO. 1, Jan/Feb 2014.
7. Tung Nguyeny, Duy-Hieu Buiy, Hai-Phong Phany, Trong-Trinh Dangand Xuan-Tu Trany, "High-Performance Adaption of ARM Processorsinto Network-on-Chip Architectures", ySIS Laboratory, VNU University of Engineering and Technology,Cau Giay, Hanoi, Vietnam.
8. Rachid Dafali, Jean-Philippe Diguetand Jean-Charles Creput "Self-Adaptive Network-on-Chip Interface", Submittedto IEEE Embedded Systemsletters, Vol. X, No. X, Month Year.
9. Ahmed H.M. Soliman, E.M. Saad, M. El-Bablyand Hesham M. A. M. Keshk, "Designing a WISHBONE Protocol Network Adapter for an Asynchronous Network-on-Chip",IJCS (International Journal of Computer Science), Issues, Vol. 8, Issue 4, No 2, University of Helwan, Cairo, Egypt 11795, Helwan, July 2011.
10. Vijaykumar R Urkude1, Dr. P. Sudhakara Rao, "Low Power 2-D Mesh Network-on-Chip Router using Clock Gating Techniques",IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 6, Issue 6, Ver. I, PP 85-91,Nov -Dec 2016.
11. Alexandros Daglis, Stanko Novakovi´c, Edouard Bugnion, Babak Falsafi, Boris Groty,"Manycore Network Interfaces for In-Memory Rack-Scale Computing" In Proceedings of the 42nd International Symposium on Computer Architecture (ISCA 2015), EcoCloud, EPFL yUniversity of Edinburgh
12. Marcelo Ruaro, Felipe B. Lazzarotto, César A. Marcon, Fernando G. Moraes "DMNI: A Specialized Network Interface for NoCbasedMPSoCs" PUCRS University, Computer Science Department, Porto Alegre, Brazil
13. Ruxandra Pop and Shashi Kumar, "A Survey of Techniques forMapping andScheduling Applications toNetwork on Chip Systems", ISSN 1404 – 0018, Research Report 04:4, Embedded Systems Group, Department of Electronics and Computer Engineering, School of Engineering, Jönköping University, Jönköping, SWEDEN
14. Brahim Attia,Abdelkrim Zitouni and Rached Tourki, "Design andimplementation of network interfacecompatible OCP For packet based NOC", International Conference on Design & Technology of Integrated Systems in Nanoscale Era, Faculty of Sciences of Monastir, Laboratory of Electronic and Micro-Electronic (LAB-IT06), Monastir, 5019, Tunisia, 2010.
15. Sujay Gejji & Tripti Kulkarni, "DesignofreconfigurableandmodularNOCinterfacewithadvancednetworking functionalities",Department of E&C, PESIT,IRD India Bangalore, Karnataka.
16. Jens Spars, "Design of Networks-on-Chip for Real-TimeMulti-ProcessorSystems-on-Chip", Department of Informatics and Mathematical ModellingTechnical University of Denmark.
17. Nauman Jalil, Adnan Qureshi, Furqan Khan, and Sohaib Ayyaz Qazi, "Routing Algorithms, Process Model for Quality ofServices (QoS) and Architectures forTwo-Dimensional4 x 4 Mesh Topology Network-on-Chip", International Journal of Computer Theory and Engineering, Vol. 4, No. 1, February 2012.
18. Ryuya Okada, Prof. Abderazek Ben Abdallah "Design of Core Network Interface for Distributed Routing in OASISNoC", ASL - Parallel Architecture Group, 2012.
Calin Ciordas, Kees Goossens, Twan Basten, Andrei Radulescu, Andre Boon, "Transaction Monitoring in Networks on Chip:The On-Chip Run-Time Perspective"

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

19. Design Methodology for Electronic Systems, Eindhoven University of Technology, Eindhoven, Embedded Systems Architectures on Silicon, Philips ResearchLaboratories, Prof. Holstlaan 4, NL-5656 AA Eindhoven.
20. Chenxin Zhang & Xiaodong Liu, "A presentation on Network-on-Chip(NoC)"
21. P. Gratz, C. Kim, R. McDonald, S.-W. Keckler, and D. Burger, "Implementation and evaluation of on-chip network architectures", Proceedings of the International Conference on Computer Design, pages 477–484, October 2006.
22. Glovanni De Michel, Luca Benini, "Networks on chips", Morgan Koufmann Publications
23. Masoud Oveis-Gharan, Gul N. Khan, "Statistically adaptive multi FIFO buffer architecture for Netwrok on chip, Microprocessor and Microsystems 39(2015).
24. Jason Cong, Michael Gill, Yuchen Hao, Glenn Reinman, Bo Yuan, "On chip interconnection network for Accelerator-Rich Architectures". DAC'15.
25. Wan-Ting Su, Jih-Sheng Shen, Pao-Ann Hsiung, "Network on chip router design with buffer stealing", 2011 IEEE.
26. Sudeep Pasricha, Nikil Dutt, Fadi J. Kurdahi, "Dynamically Re-configurable On-Chip Communication Architectures for Multi Use-Case Chip Multiprocessor Applications", 2009 IEEE.
27. Zhonghai Lu, Ming Liu, Axel Jantsch, "Layered Switching for Network On Chips", DAC 2007.

AUTHORS PROFILE

Kulkarni Rashmi Manik (Terkar Rashmi Anil) has received M. Tech. in Embedded Systems from Jawaharlal Nehru Technological University, Hyderabad, India in 2013 and B.E. in Electronics Engineering from Walchand College of Engineering, Sangli affiliated to Shivaji University, Kolhapur, Maharashtra, India. She is presently working as Assistant Professor in Noble College of Engineering and Technology for Women, Hyderabad. Her research areas of interest are embedded systems, System-on-Chip, ASIC design, Micro-controller architectures and Network-on-Chips.

Dr. S Aruselvi has completed B.E. (ECE) from Periyar Maniammai College of Technology for Women, Vallam (Tamil Nadu) in 1996. Subsequently she completed M.E.(C.C.E) from same college in 2007. She completed her PhD from Bharath University in 2017. Presently she is working as associate professor in Bharath university, Chennai. Her areas of interest are computer networking, Network-on-chips, System-on-Chips, wireless networks etc.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)