Indian J.Sci.Res. 17(2): 505 - 509, 2018 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

IMPLEMENTATION OF FPGA BASED NOC USING ADVANCED NETWORKING TECHNIQUES

[1]B.Hareesh Kumar,[2]T.Eedara Rao,[3]G.Jyothi
[1,2,3]Department of Electronics and Communication Engineering, Aurora's scientific, Technological And Research Academy, Hyderabad

Abstract- This paper presents the design and implementation of FPGA based Network on chip (NoC) which is scalable packet switched architecture with advanced Networking functionalities such as store & forward transmission, error management, power management and security. All these features are built on basic NI core, which includes data packetization/depacketisation, frequency conversion, data size conversion and conversion of protocols with limited circuit complexity and cost.

Keywords-Network-on-Chip (NoC), Network-Interface (NI), VLSI Architectures, Intellectual Property (IP), Multi-Processor System-on-Chip (MPSoC).

I. Introduction

Network on chip (NoC) [1] is an emerging design technology which is used for developing a packet switched communication infrastructure, which includes hundreds of IP cells, connected on a single Multi-processor System on Chip (MPSoC) [2]. NoCs provides a design methodology for interconnect architecture for connecting hundreds of IP cores which can be used for general purpose processors, application specific processors, digital signal processors and so on. Network interface (NI) will be considered as the key element of NoC, which makes IP macrocells to be connected to on-chip communication backbone in plug and play fashion. These NI's are also considered as building blocks of the NoC. Basically NI's takes care of data packetization / depacketisation to and from the NoC; it encodes all the packet header and guarantees a successful end to end data delivery between the IP cores.

A NoC packet consists of a header and a data payload and they are splitted into units called as flits. And all these flits are routed in the same path across the Network. The header field is composed of header field is composed of Network layer header whose content is find out by NI, according to the node map network configuration, and a transport layer header (TLH) which contains the information used by the NI's for end to end transaction management. Some of the researchers proposed that they can implement the conversion of the data size, protocol and frequency between the original IP bus and NoC. But the IP bus can be standardised one such as AMBA (Advanced microcontroller Bus architecture), AXI (Advanced eXtensible Interface) or OCP (Open Core Protocol) [3] or a custom bus such as ST Bus [4].The latest researches on the NI architecture focus on implementing more features to directly support advanced Networking functionalities, the challenge here in doing so is by keeping NI area, power and latency overhead as low as possible with respect to connected IP cores. But integrating all these above said features with limited circuit complexity will be difficult.

To overcome these draw backs we have designed and implemented a FPGA based NoC which is scalable packet switched architecture with advanced Networking functionalities such as store & forward transmission, error management, power management and security.

II. Related Work And Contributions

Design and implementation of NOC with hardware support of advanced networking functionalities on FPGA can be defined as the framework employed to enhance some networking functionalities, which in turn enhance the parameters such as store and forward, security, EMU (Error management Unit) for initiator, EMU and PM (Power management) for target. Pier.S.Paolucci, et al [2] proposed that nano scale systems on chip with tiled scalable hardware and software design for future CMOS technologies. Tiled architectures suggests a possible path "small "processing tiles connected by "short wires" a typical shapes tile containing a very long instruction word (VLIW), floating point DSP, a RISC, a DNP(distributed network processor), distributed on chip memory plus an interface for DXM (distributed external memory). But here, there is no processing power ceiling for low consumption, low cost, dense numerical embedded scalable systems for future embedded audio, video and human -centric applications.

Heikki Kariniemi and Jari Nurmi [5] came out with a new approach to a NOC Interface (NI) called Micro Switch Interface (MSI) designed for message passing communication with a light-weight micron message passing (MMP) protocol on micromesh MPSOC platform.

[1]Corresponding Author

The operation of the MSI hardware and software are tightly coupled with that of the MMP protocol in order to improve communication performance. But the operation of the MSI hardware & software is tightly connected to the operation of the MMP protocol and it cannot give protection against packet losses and errors in the NOC and also if modifications are made which will increase the cost of the MSI hardware.

Srinivasan Murali et al [6] proposed that NoC's are necessary to efficiently handling the 3D interconnect complexity, while satisfying the 3D technology constraints. But fails to meet the constraints for the 3D approach it is a great challenge for the designers. One of the most important problems is to design the most power performance efficient NoC topology that satisfies the application characteristics and 3D technology requirements.

Mohammed Anis Ur Rahman et al [7] introduced that "Network on Chip (NoC) design paradigm, where nodes communicate by exchanging packets through an interconnection network, which consists of routers and network interfaces, here 2D mesh node communication architecture is proposed in GALS approach. While using GALS approach the nodes has to obey some important rules for a guaranteed better performance, also NoC is best used in high bandwidths, large scale environment. This limits the implementation fields and the design is not capable of handling QoS traffic.

III. Design Of Core Network Interface

A. Top level Architecture of NI.

In NOC interface IP cores are commonly classified as Master and Slave IPs. Initiator NIs are connected to Master IPs, which will convert the IP request transaction into NOC traffic, and also translates the packets received from NOC into IP response transactions. Similarly, Target NIs also exists, which are connected to slave IPs, here the target NIs represents a mirrored architecture where the requests are decoded from the NOC and the responses are encoded.

In both NI types, two main domains are identified Fig (1(a)) which shows the top view of an initiator NI. And Fig (1(b)) shows the top view of the target NI, here the shell is considered as IP specific, and Kernel in NOC specific, where each one has its own peculiar functionality.

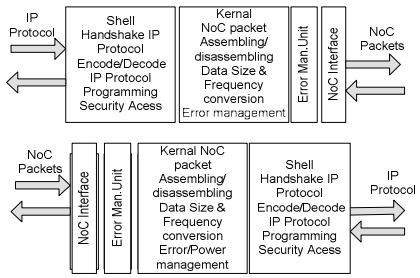

Fig 1. Top view of the NI design: (a) Initiator and (b) Target

Fig 1(a) and 1 (b) highlights some advanced networking functionalities such as programming, security, error and power management. The main aim of the shell and Kernel separation is to abstract IP specific properties (such as bus protocol and data size) from NOC side properties.

In this manner the NOC becomes an IP-Protocol agnostic interconnect which includes protocols, bus size, clock frequency, the master or slave IP is using and all modules in the system may communicate with each other.

The conversion features should be implemented in two directions, one in request path which is from master to slave IPs and one more in response path from slave to master respectively. Here the Kernel and all its associated NOC interface is IP protocol independent and its common to all possible NI's. This supports the following IP bus protocols. AMBA, AXI, which is a de-facto standard in embedded systems ST-Bus type used by ST microelectronics, DNP a distributed network processor. Mainly used by ATMEL to build a multi-tile MPSOS architecture.

Fig 2 highlights the shell, the Kernel and the NOC interface respectively, and here the top of this figure represents the request path, while the bottom part represents the response path. The NOC interface consists of Upstream (US) section, to transmit packet to the interconnect, and a downstream (DS) section which receives the packets from the NOC.

The NI shell part deals directly with the bus protocol which implements bus specific handshaking rules by means of dedicated Finite State Machines (FSMs). Before passing the data to the kernel, the shell builds the network and transport layer headers, which are required by subsequent NOC components.

B. Kernel-Shell Interface by BI-Synchronous FIFO

The Kernel is interfaced to the shell by means of a FIFO like interface. As Shown in figure 2, encoded data which in coming from the shell are stored in two FIFO's. One is header FIFO (which holds transport layer and network layer headers) and a payload FIFO (which holds bus raw data). Each FIFO consists of its own read and

write manager which will updates FIFO pointers and status and provides frequency conversion mechanism. The Kernel is connected to the NOC interface through two additional FSMs. In the request path, an output FSM (OFSM) which will reads the headers and payloads and converts them to the packets according to the need of NOC protocol.

Similarly in the response path, an Input FSM (IFSM) collects the packets and splits the header and payload flits into their respective FIFO's. Here it should be noted that the NI encodes both TLH and the NLH, while decoding it takes only TLH into account, because as already the packets have been reached the destination and the routing data is also not required.

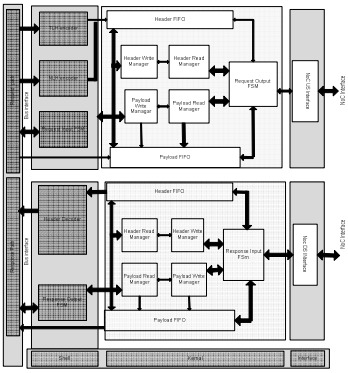

Fig 2: Main Blocks In NI Micro Architecture.

Bi-synchronous FIFOs are used in the NI scheme. The frequency conversion mechanism is accomplished between NOC and connected IP. Each read (write) FIFO manager re-synchronizes in its own clock domain, the pointer of the write (read) manager in the other clock domain, hence the status of the FIFO that is empty or full is find out using comparing synchronized pointers, and the header and the pay load FIFO's can be correctly managed. To increase the robustness of the synchronization pointers they adopt gray encoding. The read (RD) and write. (WR) managers can access a FIFO by different basic storage elements also, data size conversion is possible between IP and NOC domain the conversion mechanism is carried out by exploiting FIFO rows and columns. A FIFO location is sized according to the larger data size between data in and data out. A FIFO row is sized according to the smaller data size between data in and data out as shown in Fig 3.

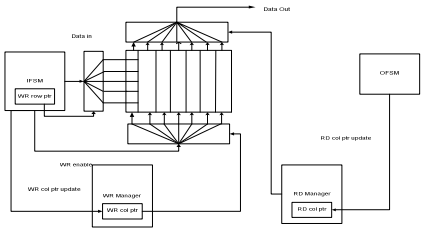

Fig 3: Upsize conversion in single FIFO.

Up size conversion is accomplished by writing by rows and reading by columns and downsize conversion is exactly opposite to that. In particular cases when Shell / Kernel frequency conversion nor size conversion is not required, it is possible to remove the bi-synchronous FIFOs, by setting their size to zero, by this which will save area and power consumption this feature is known as Zero kernel FIFO. As to increase the maximum operating clock frequency, an optional pipeline stages can be added at the IP and NOC interfaces.

IV Advanced Network Interface Features

A. Store and Forward (S & F)

Kernel FIFOs in the request and response paths contains flits, which are either received from the NOC and to be decoded towards the IP bus, or encoded from a bus transaction and to be transmitted over the interconnect.The default NI behaviour is that as soon as flit is available from the NOC it is extracted as a result the original traffic at an interface (NOC or bus) has an irregular shape, as soon as store and forward is enabled, flits are kept into the internal Kernel FIFOs, until the whole packet is encoded /received and then they are transmitted / decoded all together. In this way an irregular traffic is converted to bubble free traffic thus improving overall system performance.

S & F may be enabled on both request path and response path independently, at NOC- to- bus level, it is possible to enable per-packet S & F, while several S & F options can be selected at NOC-to bus level. The mechanism of handling pre-packet S&F implementation is quite simple, after completion of a packet, the FSM controlling the FIFOs reading is in a state where only the header FIFO is checked , to extract the beginnings of the new packet. Here the concept to handle per-packet S & F, in both directions is to keep packet header hidden to the reading logic by simply not updating the header FIFO write pointer. When the entire packet is stored in the FIFO (both header and payload), the header is unmasked and made visible by updating the header FIFO pointer, and the reading logic detects the presence of a new packet.

The management of the bus-to-NOC store and forward compound transactions is bit more complex, as the compound transactions consists of number of packets, that is number of headers, the header write pointer must constantly updated upon arrival of any new packets, to avoid overwriting the previous one, therefore the headers become visible to the reading logic. Here the concept is that a field in the header that to mark the packets header as "hidden" (the first packets of compound transaction) or "visible" (only last packets of the compound transaction) the reading logic evaluates this field, in each header and starts extracting the FIFO's content only when the last packet of the compound transaction is detected in the FIFO.

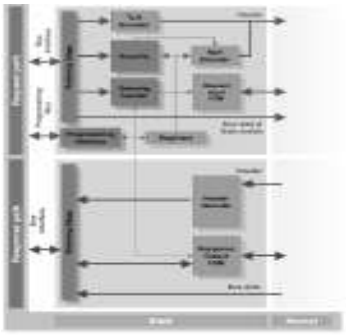

Fig 4. Advanced Features In A NI Initiator.

B.Error Management Unit (EMU)

The EMU is an optional stage which can be implemented between the Kernel and NOC interface. The behaviour of the EMU is different for both initiator and target NIs. In an initiator NI the EMU can handle bad address errors and security violations. When the address of the master IP transaction is not in the range of assigned memory map, or when the transaction is trying to access the protected memory zone without having permission the packet in its header is flagged as an error packet.

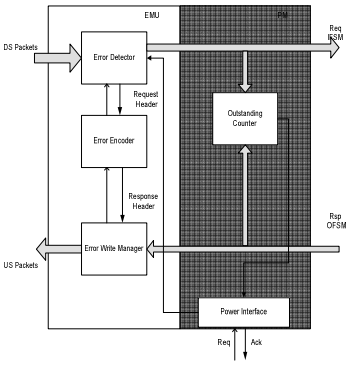

Fig 5: EMU And Power Manager In Target NI.

In the later stages the EMU filters the packet directed to the NOC US-interface to avoid it to enter the network, and creates a response packet. Which remaps the request header on a new response header, and if required adds a dummy payload.The EMU also take care of properly managing the incoming traffic at the DS NOC interface during the power down mode. All the traffic which is received in request during power down mode is flushed by the EMU, so that it never reaches the slave IP.

The EMU itself generates an error response to the master originating the request. As shown in Fig 5, the EMU is composed of 3 blocks which are

Error detector, which flushes all error traffic. In Initiator NIs, the outgoing error traffic is identified by a flag in the header, while in Target NIs all incoming packets are flushed if the connected Slave IP is in power down mode;

Error encoder, which assembles a new NoC packet to be channelled in the response path;

Error write manager, which is basically a traffic-light to avoid simultaneous traffic to the US NoC Interface from Kernel Response and from the EMU in a Target NI, while in an Initiator NI it avoids interference between the DS NoC Interface and the EMU both trying to access the Kernel Response.

If a request packet does not contain an error, the EMU behaves transparently and does not add any clock cycles of latency.

C.Power Manager (PM)

This feature is available only for Target NIs connected to slaves which may be turned off to save power. The PM is always coupled to an EMU block which rejects incoming NoC packets trying to access the Target NI when the connected slave IP is in power down mode as shown in fig 5. A simple req/ack protocol controls the power up/down state of the NI, by means of a dedicated interface: each request (req set to 1) acknowledged by the PM unit (ack set to 1) makes the NI power state switch from UP to DOWN and vice versa. It may happen that a request for power down is sent to the PM while the slave IP is still elaborating a number of pending transactions. In this case the Target NI stops accepting packets from the network and waits for all pending transactions to be processed before acknowledging the request and switching to power down mode. The power manager is a completely new feature introduced by the proposed NI.

D. Security

The security service, available only in NI Initiators, acts as a hardware firewall mechanism. It introduces a set of rules that transactions coming from the Master IP must satisfy to gain access to the network. The security rules involve:

i. Lists of memory intervals under access control;

ii. Lists of Master IPs that may have access to a certain memory region;

iii. Lists of access types.

Security rues are applied in the Security block during packet encoding. If a test fails the security check, the corresponding transaction is marked as an error in the NLH and it is detected by the EMU, which must be activated as well to properly manage security violations. The illegal packet is then discarded and does not consume network bandwidth, and the error response to the Master IP is directly generated by the EMU itself. The rules that allow a transaction to access the network are described by means of a security map. In this map a number of memory regions are defined, and associated to region IDs the same map defines how these regions can be accessed. Access to

these zones can be allowed only to Master IPs belonging to specified groups. Source and for each protected region depending on the address and the Source, the memory access can be immediately allowed or immediately denied, or go through a security rule check. Naturally, there is also a control to block malicious memory accesses to a non-protected zone that transfer a number of bytes such that the operation overflows in a protected region.

A region can be any sub range of the address space in the whole system interconnected by the network. The security map may be statically defined at design time or changed dynamically through the programming interface; in the latter case the NI programming interface must be configured to instantiate the registers related to the security category. The implemented security mechanism supports up to 8 protected memory regions, up to 8 access rules with Read/Write/Execution permissions, up to 16 Source groups to classify Masters.

V. NOC Implementation Results

A. NI configuration space.

The proposed NI is designed to support a wide configuration space, not only the advanced feasters can be enabled or disabled, also some of the basic characteristics can be configured like flit size, bus data size, pay load & header FIFO size, conversion of frequency and data size. By changing the configurations set different trade-offs between performance and complexity are achieved.

B. Verification and synthesis flow.

The exact functionality of the proposed NI design in multiple configurations has been verified at different abstractions levels.First a random functional verification environment has been created and applied to multiple configuration of HDL database.

The various building blocks of the NOC are exploited using verilog HDL language and Xilinx ISE tool. Which is a software tool produced by xilinx for synthesis and analysis of HDL designs, which enables the developer to synthesize (Compile) their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction, and it is simulated and debugged using modelsim and the simulation waveform can be obtained which will be further useful for analysis.The NI data base is verified and validated at different level by both synthesis and simulation. Finally, the design is configured on SPARTAN 3 family FPGA kit with the help of Verilog HDL.

VI. Conclusion

An advanced design of Network interface for on chip communication has been presented in this paper ,the proposed design supports a wide set of with advanced networking functionalities such as store & forward transmission, error management, power and security. The capability to support all these above said features in the same hardware reduces the complexity overhead.The proposed Network interface represents a complete solution which can be used in different scenarios from real time applications to multimedia broadcasting based on the system requirements by modifying some features to obtain an optimised hardware description.

References

[1] NoC Advantages for SoC Prototyping on Big FPGA BoardsJonah ProbellArteris, Inc jonah@arteris.com

[2] P. S. Paolucci, F. LoCicero, A. Lonardo, M. Perra, D. Rossetti, C. Sidore, P. Vicini, M. Coppola, L. Raffo, G. Mereu, F. Palumbo, L. Fanucci, S. Saponara, and F. Vitullo, "Introduction to the tiled HW architecture of SHAPES," in *Proc. Design, Automation and Test in Europe*, 2007,pp. 77–82.

[3] B. A. A. Zitouni and R. Tourki, "Design and implementation of network interface compatible OCP for packet based NOC," in *Proc. 5th Int Design and Technology of Integrated Systems in Nanoscale Era (DTIS) Conf*, 2010, pp. 1–8.

[4] T. Tayachi and P.-Y. Martinez, "Integration of an STBus Type 3 protocol custom component into a HLS tool," in *Proc. 3rd Int. Conf. Design and Technology of Integrated Systems in Nanoscale Era DTIS 2008*, 2008, pp. 1–4.

[5] "NoC Interface for fault-tolerant Message- Passing communication on Multiprocessor SoC platform," in *Proc. NORCHIP*, 2009, pp. 1–6.

[6] "Synthesis of networks on chips for 3D systems on chips," in Proc. Asia and South Pacific Design Automation Conf. ASP-DAC 2009, 2009, pp. 242–247.

[7] "Efficient 2DMesh Network on Chip (NoC) considering GALS approach," in Proc. Fourth Int. Conf. Computer Sciences and Convergence Information Technology ICCIT '09, 2009, pp. 841–846.