Design and Physical Implementation of a Data Transfer Interface Used in Network on Chip

A. A. EL OUCHDI[1.2], N. TAHIR[1.3] and A. BOUOUDEN[1.4]

[1] Centre de Développement des Technologies Avancées, Division of Microelectronics & Nanotechnologies, Cité du 20 Août 1956, BP.17 Baba Hassen, 16303, Alger, Algérie.
[2] University of Tlemcen, Faculty of Technology, Electronic Department, URMER Laboratory, Tlemcen, Algeria.
[3] University USTHB, Faculty of Electronic and Computer Science, LCPTS Laboratory, Algiers, Algeria.
[4] University of Constantine, Faculty of Science and Engineering, Electronic Department, Constantine, Algeria.
aelouchdi@cdta.dz, ntahiri@cdta.dz, abououden@cdta.dz

Abstract

The implementation of an efficient network on chip (NOC) requires the design of efficient network interfaces (NI) which connects the intellectual's properties (IPs) to the routers. According to the NOC topologies, these NIs may be different. In this paper, we present at first the design of a novel architecture of a data transfer interface (DTI) which will be used in a new paradigm of network on chip, and then, we present the results of the verification and the physical implementation of this DTI. Because of the frequencies between the IPs and the network on chip are often different, the DTI is used for synchronizing data and avoiding the metastability risk. It is also used for storing the data in a FIFO in order to avoid losing them during the transfer. The DTI was designed with Verilog language, the verification was done by the Modelsim tool and finally the logical synthesis and the physical implementation were achieved by Cadence RTL Compiler and Encounter Digital Implementation respectively.

***Keywords*— Network on chip NOC, Data Transfer Interface DTI, FIFO, System on chip SOC and Metastability.**

1. Introduction

The development of microelectronics fields has led to the integration of millions of transistors in the same integrated circuit IC. This allowed the integration of complex systems on chips (SOC) that are composed of many IPs blocks in the same chip. But, on the other hand, since most of the communication inside the SOC is based on buses, it has its inherent limitations [1], and in the future SOCs, the increase of wiring delays, noise and power dissipation will play a negative role in the behavior of the IC.

To overcome to this problem, the NOC architectures were proposed as a promising solution due its reusability, scalability and reliability. It's obtained by replacing communication based on buses by a network inside the chip. It allows the integration of more IPs, high performance communication, high throughput and low power consumption compared to the bus-based communication. NOCs are generally composed of routers or switches, IPs, links and networks interface s as it is shown in Fig.1.

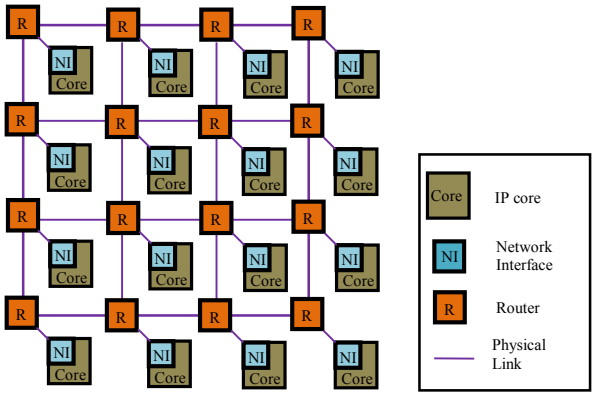

Fig.1. Block diagram of a network on chip

The routers carry the data from one node to another, the IPs are the cores that either execute a request or a response, the links provide the connection between the routers and finally the networks interfaces (NI) connect the IPs to the routers by converting request and response transaction into packets and vice v ersa [2].

There are a number of published works on the design of the interfaces between the IPs and the NOC; each design is based on its proper NOC topology. In [1] a low latency NI compatible with the AMBA AHB IPs standard was designed. In [2] a NI compatible with the OCP IPs standard was designed. In [3] a low power NI for NOC was designed with a stoppable clock in order to reduce the power dissipation. In [4] a high speed generic NI using Ping Pong Buffers was designed. In [5] a low power NI with a flexible run time configuration controller in order to reduce the power by enabling and disabling the entire asynchronous FIFO based on the traffic conditions was designed. In [6] an efficient NI that decouples computation from communication offering guaranteed services, shared-memory abstraction, and flexible network configuration was designed. In these previous works, the majority of the NOC topologies are according the mesh network topology. In [7] a new client interface architecture for network on chip based on the modified fat tree (MFT) topology was designed which is characterized by a high numbers of links at the client interface; if there are 'n' clients, there will be 2^{n-1} links, and each link is characterized by a FIFO buffer as called by the original FT architecture [8]. So, instead the use of several FIFOs, only one single centralized FIFO memory is used for which all the incoming links are connected through a single 2^{n-1} to 1 multiplexer, and then the

output of this centralized FIFO is connected to an interface that transfers data between two different clocks with a Handshaking mechanism that performs the data transfer. In [9] a new FIFO for transferring data between two unrelated clock domains FIFO was designed, but instead of having cells made of regular latches, it uses simple asynchronous pipelines as the basic cell.
In this proposed work we have designed a new and easy to use data transfer interface for network on chip based on the new methodology proposed in [10-12]. It's used as a bridge between the IPs and the NOC, its purpose is to make the synchronization between two different clock frequencies during the transfer for avoiding the metastability risk [13-14]. Because the routing algorithm is known in the advance and each IP communicates with only one IP in a given time, the advantage of this new topology of NOC is the need of only a simple DTI that manages the transfer between the IP and the NOC compared to other topologies that need complexes interfaces, more areas and more time of transfer. This interface was designed from the RTL codes to the final GDSII file respecting all the steps of the logical synthesis and the physical implementation.
This paper is divided in four parts; in the second part we present an overview of the NOC topology, in the third part we present the structure of the proposed interface, in the fourth part we present the result both of the simulation, the logical synthesis and the physical implementation of the DTI.

2. The used NOC topology

Many methodologies of NOC design have been proposed. They are mainly mesh networks, fat tree-based topologies and their variants. A new design methodology has also been proposed [10-12], given a system to be implemented by an ASIC, the methodology aims at solving the routing issue at design time. At run time, there is only a need to make a 0-1 bit configuration to enable or prevent a data transfer through a given interconnection. Thus, almost the entire timing budget is left to the application itself, which renders such a methodology a good candidate to either design real-time system or high-throughput ones. In [10-12], an approach FPGA-like is proposed for on-chip communication with a design methodology where switches are avoided and where two any IPs are connected if and only if they communicate between them. It avoids the problem of the cost time, the area and the energy. It avoids also the intermediate hop counts of on-chip networks (NOC) between any two source-destination pairs. The second novelty is that without knowing the overlap/disjointness of communication packets between any two IP blocks, a designer may use unnecessary resources (wasting time, space, energy, resources, etc). So, it is looked into the temporal aspect of communication and then it is possible that some communication phases don't overlap, thus a designer needs no provision resources for such cases. Also, the CAD tool related to this methodology aims at designing a NOC subject to bandwidth, area and energy constraints. The main components of this architecture are:

- The IPs (that constitute the target system).
- The optimal number of interconnects allowing to achieve all the data transfers without any conflict (the CAD tool automatically assigns an interconnection to each data transfer so that simultaneous transfers are achieved without any conflict).
- CMOS pass transistors that allow or prevent data transfers (the signals that feed the gates of the PMOS and the NMOS transistors are delivered according to the behavior of the system).
- OR gates (the same CMOS pass transistor may be used to allow or prevent many data transfers).
- Inverters (for each signal feeding the gate of the NMOS transistor, its complement feeds the gate of the PMOS transistor of the same CMOS pass transistor).
- A control unit for each IP (such a part mainly configures the bits of the signals feeding the CMOS pass transistors controlling the transfers of the data sent by the interested IP). Note that because the number of IPs composing a complex system could be large, it would not be interesting to design a single control unit for all those IPs (the interconnections would be long, the number of gates would be important, leading to both weak throughput and high power dissipation).
- The data transfer unit (DTI): because the IPs and the NoC don't necessarily operate with the same clock frequency, the metastability issue may arise. One then needs to design an interface so that to avoid such a problem.

The advantage of this new architecture is that Each IP has its proper controller that enables or disables a given interconnection by the way of the pass transistors so that when an interconnection is enabled, there is only one transfer that occurs between two IPs, so an IP always sends (receives) to (from) only one another IP in a given time. Unlike the mesh network architecture, the same interconnection is used to transfer the entire message directly (there are not flit in our case) because the routing algorithm is known in the advance. After the transfer, the IP can communicate with another IP by the same way that the first operation. This theory has led to the use of only a simple DTI to perform the transfer since there are no conflict in the interface between the NOC and the IPs.
Fig.2. shows how the DTI is integrated with the other parts of the whole design.

3. The proposed data transfer interface

In our work, we have designed a data transfer interface DTI that plays the role of the network interface. It is used in one hand for the synchronization of the data to avoid the metastability risk because the frequencies of the IPs and the network on chip are often different, and in other hand, it is used for the storage of the data in a FIFO to avoid to losing them during the transfer.

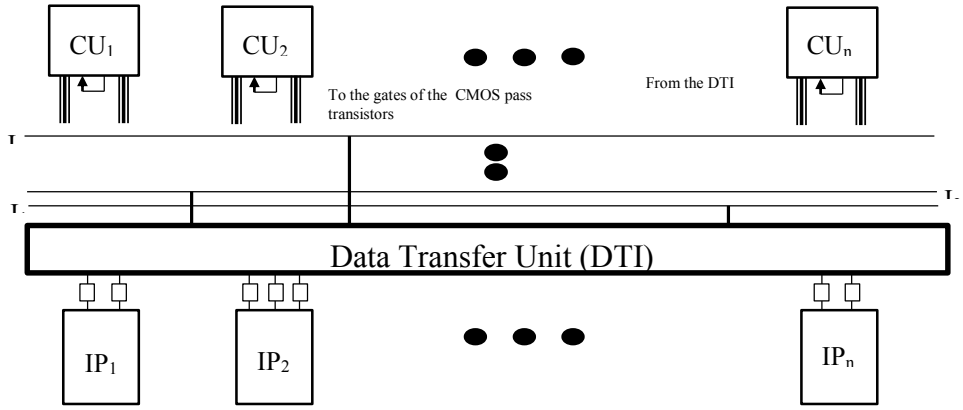

Fig.2. The used NOC topology

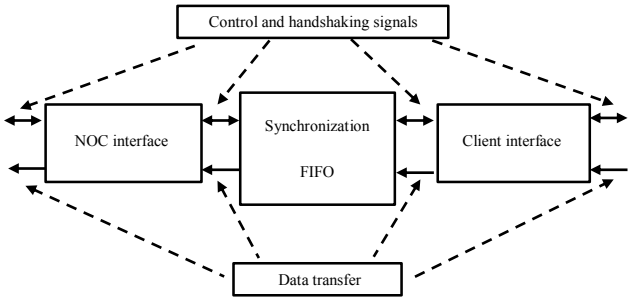

Fig.3. The DTI's architecture

The DTI is composed of one FIFO for the storage of the transferred data with a combination of flip flops that ensure the synchronization of the signals, an interface with the client or the IP that manages the data transfer from the client to the FIFO while considering the handshaking protocol, an interface with the network that manages the data transfer from the FIFO to the network while considering the handshaking protocol and two control signals that control the client interface and network interface respectively.

Metastability generally occurs when the input signal is asynchronous i.e. its changes violate the setup and hold time of the flip-flop. It can also occur when we interface two domains operating at different frequency. To overcome to this problem, a synchronization using two flip flops is needed to make the input synchronous [13-14].

In order to avoid writing in a full FIFO or reading from an empty FIFO, the FIFO's size must be calculated correctly. In fact, to calculate the FIFO's size we must consider the worst case scenario for the data transfer across the FIFO i.e. the difference of the data rates between the write and read operations should be maximal; for the write (read) operation, a maximum (minimum) data rate should be considered. If B is the number of data and if T1 and T2 are the writing and the reading periods respectively, the needed FIFO's size would be: $S = B - B.T1/T2$.

4. Simulation results

The RTL description of the DTI is designed by the Verilog language. It's composed of the top module that instantiates 5 sub modules that are:

- The FIFO described previously.
- The client interface.
- The network interface.
- Two finite state machines that control the client interface and the network interface.

The FIFO is composed of several stages which represent one unit of its allocation. Each stage allows for one data to be synchronized from the client's frequency to the network's frequency. It manages also the handshaking mechanism inside the module in both the client side and the network side respectively.

4.1 Design verification

In order to verify the design, a test bench file was created so as to manage the clock signals of both the client and the network and makes the simulation by instantiating the top module of the designed DTI. The simulation was achieved by the Modelsim tool 5.7 and it is explained by the bloc diagram of fig.4.

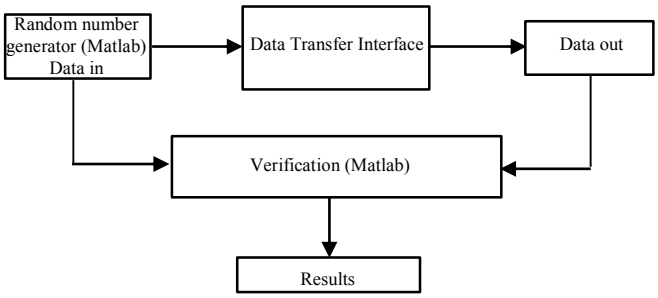

Fig.4. Simulation with Modelsim tool

The Matlab file generates random numbers; in our case, they are hexadecimal and are considered as an input data that feed the DTI. These input data cross the DTI and the output data are compared with those of the input using the Matlab file which outputs the comparison results. Table 1 shows the comparison between the input data and the output ones while indicating if there exists an error (in such a case its position and its percentage are given).

Table 1. Comparison between input and output data

Data_in	8	16	32	8
Data_out	8	16	32	8
err	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 0 0 0 1 0 0 0

position_error	Empty matrix: 1-by-0	Empty matrix: 1-by-0	Empty matrix: 1-by-0	1 5
Total error	0	0	0	2
Error data report	0	0	0	0.25
Data error percentage	0%	0%	0%	25%

In the three first cases, the input data and the output ones are the same and there is no error during the transfer. So, both the position and the error percentage are null. In the last case, an intentional modification was made in the output file in order to check whether the simulation works well, it appears that some errors happen during the transfer which are indicated by their position and their total percentage. The total time of simulation is given by Table 2. The total time of transfer increases with the augmentation of the transferred data. It increases also by the augmentation of the clock period as is given by Table 3.

Table 2. Total time of simulation

Number of transferred data	Simulation time (µs)
8	250
16	490
32	970
64	1930
128	3850
256	7690
512	15370
1024	30730

Table 3. Influence of the clock period in the case of the transfer of 8 data

Client clock period (ns)	Simulation time(µs)
1000	250
1500	372.5
2000	495
2500	617.5
3000	740

4.2 Logic synthesis

After the verification of the design, the logic synthesis was achieved using Encounter RTL Compiler from Cadence tool. It builds the module by transforming the HDL files into combinatorial and sequential logic while using a library of standard cell (0.18µ in our case) and respecting the step of synthesis. Fig.5. shows the schematic of the synthesized DTI.

We note that we can verify the blocks forming the module from the top until the simple logical gates. The output file of the logical synthesis is a Netlist file in logical level that represents the instantiation of the logical gates forming the design with the interconnections respecting the target library. It is used as an input in the physical implementation. The logical synthesis generates some reports that inform us about the area, the number of gates and power consummation. These information are shown in table (4).

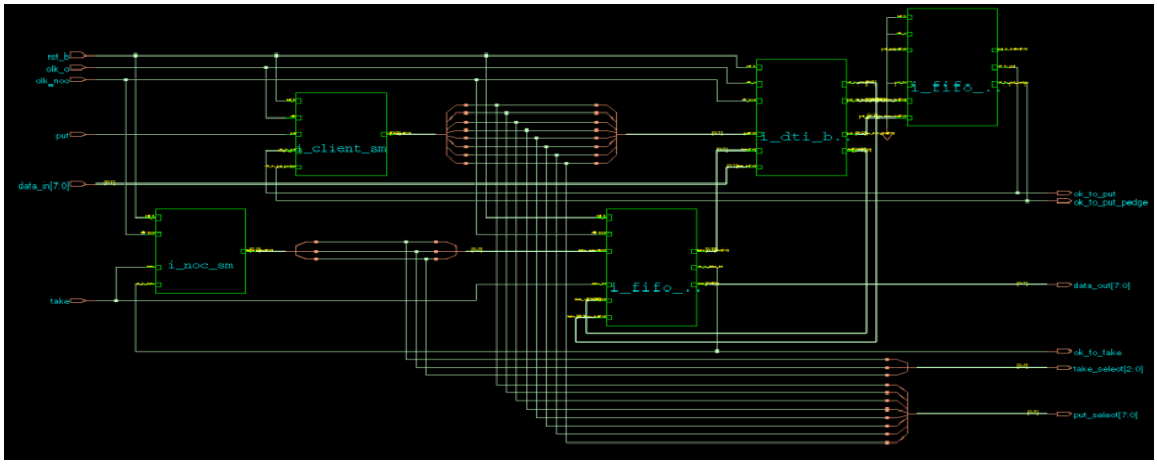

Fig.5. Logic synthesis of the DTI (by Encounter RTL Compiler)

Table.4. The generated reports

FIFO's size	Area (µm²)	Gates	Power (nw)
L08W08	14686	717	785673.658
L08W16	19333	1030	943902.954
L16W08	25280	1292	1263606.581
L16W16	33850	1897	1576010.651
L32W08	46237	2470	2237352.419
L32W16	62974	3683	2690721.724

Knowing that L and W represent respectively the length and the width of the DTI_FIFO, we note that the area, the number of gates and the power consummation increase with the augmentation of the size of the DTI_FIFO.

4.3 Physical implementation

After the logical synthesis, the physical implementation was made by Encounter Digital Implementation EDI from Cadence tool. It carries out the placement and the routing by placing the resulting Netlist of the logical synthesis into a layout while respecting the

steps of the physical implementation. The obtained Layout is shown in Fig.6.
The output file of this step is a GDS2 file that can be exported in different formats for further processing outside the Encounter tool.

The GDS2 format is a standard format for integrating the block in the top-level layout, doing DRC/LVS checking, or delivering the layout to the foundry.

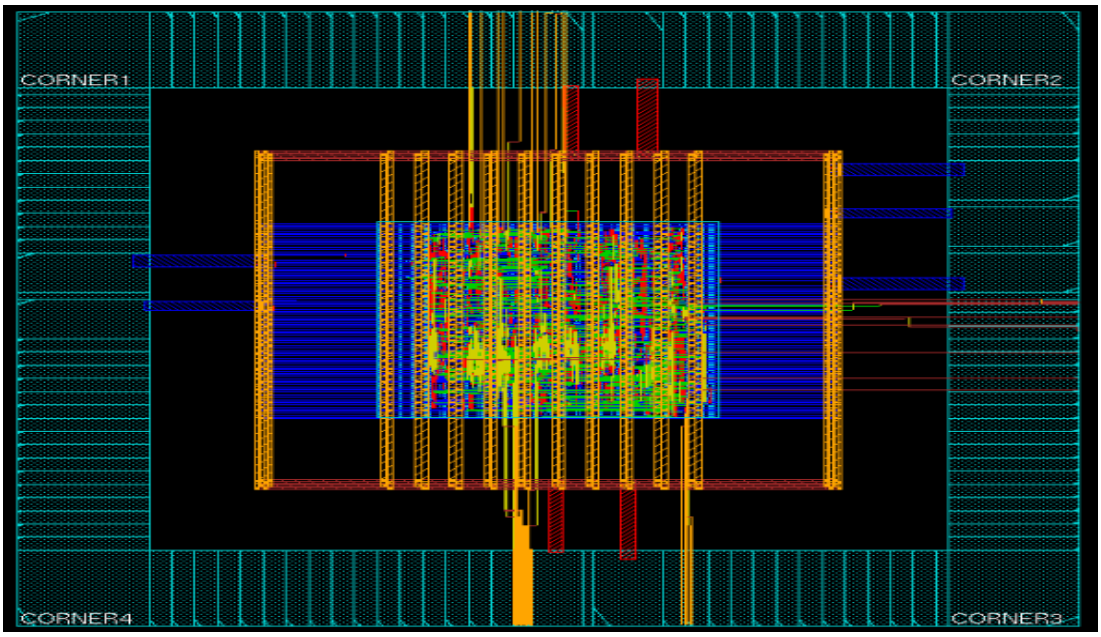

Fig.6. Layout of the DTI

5. Conclusion

The DTI have been studied with different sizes of length and width as well as with different clock periods. In all cases it worked well. After all these steps, the designed DTI will be implemented on the proposed network on chip to get the final tool.

6. Acknowledgement

We would like to thank Dr Samir TAGZOUT for his support and discussions to achieve this paper.

7. References

[1] B. ATTIA, W. CHOUCHENE, A. ZITOUNI, N. ABID and R. TOURKI, "Design and implementation of low latency network interface for Network on Chip", International conference on Design and Test Workshop (IDT), P37-42, 2010.
[2] B. ATTIA, A. ZITOUNI and R. TOURKI, "Design and implementation of network interface compatible OCP For packet based NOC", International conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), P1-8, 2010.
[3]W. CHOUCHENE, B. ATTIA, A .ZITOUNI, N. ABID and R. TOURKI, "A Low Power Network Interface For Network on Chip", 8th International Multi-Conference on Systems, Signals & Devices (SSD), P1-6, 2011.
[4] K. SWAMINATHAN, G. LAKSHMINARAYANAN, S.B. KO, "High Speed Generic Network Interface for Network on Chip using Ping Pong Buffers", International symposium on electronic system design (ISED), P72-76, 2012.
[5] K. SWAMINATHAN, G. LAKSHMINARAYANAN, F. LANG, M. FAHMI, S.B. KO, "Design of a low power network interface for Network on chip", 26th IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), P1-4, 2013.
[6] A. RADULESCU, J. DIELISSEN, K. GOOSSENS, E. RIJPKEMA, AND P. WIELAGE, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration", Design, Automation and Test in Europe Conference and Exhibition, Proceedings (Volume 2), P878-883, 2004.
[7] M. E. S. ELRABAA, A. BOUHRAOUA, "A hardwired NoC infrastructure for embedded systems on FPGAs", Microprocessors and Microsystems, Volume 35 Issue 2, P200-216, March, 2011.
[8] A. BOUHRAOUA, M. E. S. ELRABAA, "An efficient network-on-chip architecture based on the fat-tree (ft) topology", International conference on Microelectronics, ICM '06, P-31, 2006.
[9] M. E. S. ELRABAA, "A new FIFO for transferring data between two unrelated clock domains", International Journal of Electronics, 2012.
[10] A. MAHDOUM, "A New Design Methodology of Networks on Chip", IEEE/Asian Symposium on Quality Electronic Design (ASQED'12), July 10-11 2012, Penang, Malaysia.
[11] A. MAHDOUM, "Architectural Synthesis of Networks On Chip", IEEE /International Conference on Industrial Electronics and Applications (ICIEA'13), June 19-21 2013, Melbourne, Australia.
[12] A. MAHDOUM, "Networks On Chip Design for Real-Time Systems", IEEE/International System On Chip Conference (SOCC'14), September 2-5 2014, Las Vegas, Nevada, USA.
[13] A. CANTONI, J. WALKER and T. D. TOMLIN, "Characterization of a Flip-Flop Metastability Measurement Method", Circuits and Systems I: Regular Papers, IEEE Transactions on, Volume:54 , Issue: 5, 2007.
[14] D. KINNIMENT, K. HERON, and G. RUSSELL, "Measuring Deep Metastability", 12th IEEE International Symposium on Asynchronous Circuits and Systems, 2006 Grenoble, France.