Journal of Information Technology and Engineering
Vol. 1 No. 1 (June, 2016)

Design of a Scalable 2D Mesh-Torus Router for Network on Chip

Mohamed ATRI[1], Yahia SALAH[1], Kholdoun TORKI[2] and Rached TOURKI[1]

[1]Laboratory of Electronics and Microelectronics, Faculty of Sciences Monastir, 5000, Tunisia
E-mail: Mohamed.atri@fsm.rnu.tn, yahia_fsm@yahoo.fr, Rached.tourki@fsm.rnu.tn

[2]TIMA-CMP Laboratory, 46 Avenue Félix Viallet Grenoble, 38000, France
E-mail: Torki@imag.fr

Received: 15th December 2015 Revised: 10th January 2016 Accepted: 17th March 2016

Abstract: *New systems on chip (SoC) design allow to build heterogeneous systems with several functional units, distributed memories, and interconnections on the same chip. In order to achieve more reuse, flexibility, and performance, bus based interconnections are no more sufficient and Network on Chip concepts emerged.*

This paper presents the design of a scalable packet based router allowing data transfer and managing dynamically several communications in parallel. The designed router, described in VHDL on RTL level, was simulated in the case of topologies 2D-mesh and 2D-torus (2x2), (3x3) and then (4x4). The used design methodology is based on VHDL as a description language, simulation and synthesis tools.

Keywords: *Network on chip, 2D mesh-torus, router, VHDL, Design.*

1. INTRODUCTION

NoC is the communication media in a system-on-chip (SoC) environment. Traditionally on-chip communication has been conducted via dedicated point-to-point links or a shared media like a bus. Busses are not scalable with respect to speed and power; also they quickly become the bottleneck of a system. As described by [1], SoC design is based on intellectual property (IP) cores reuse. These cores can be analog/digital cores, memory blocks, DSP cores, and also new technologies. Cores do not make up SoCs alone; they must include an interconnection architecture and interfaces to peripheral devices [2]. The interconnection architecture includes physical interfaces and communication mechanisms, which allow the communication between SoC components.

This paper presents the study and design of a NoC topology with different topologies: 2D-mesh and 2D-torus. The basic element of this topology is a switch making possible the communication between the IPs cores.

2. RELATED WORKS

In any system on chip, the communication management becomes the bottleneck limiting the total performances. As a new paradigm of SoC, with a communication centric design style, Network-on-Chip (NoC) [1, 2] was proposed to meet the distinctive challenges of providing functionally correct, reliable operation of interacting SoC components. A few on-chip micro network proposals for SoC integration can be found in the literature. Several NoC interconnects have been proposed recently [1-12]. The current studies are interested mainly in the problem of topologies, buffering strategy, switching mode, contention resolution, network flow control, and traffic processing, to satisfy the new paradigm of SoCs.

2.1 Network Topology

The authors of [4] and [5] proposed mesh-based interconnect architectures. These architectures consist of an $m^* n$ mesh of switches interconnecting computational resources (IPs) placed along with the switches. This topology is predominant in the literature because of its easy implementation, simplicity of the XY routing strategy and network scalability. The short distance between switches eliminate a certain number of electric phenomena in comparison with the buses. On-chip networks will likely use networks with lower dimensionality to keep wire lengths short [1]. In [6], Dally et al. proposed the use of a torus interconnect architecture, to reduce the network diameter. A variation of the torus architecture, which eliminates the use of long Wrap around wires, called a folded torus, is described in [7]. A problem of torus and mesh topologies is the associated network latency. To overcome this problem, one proposes other NoCs alternatives. The fat tree topology has the following advantages: its diameter (maximum number of links between two subscribed) remains reasonable ($2*\log_4 n$, where n is the number of layer of network), the topology is scalable and uses a small number of routers for a given number of subscribers. It has a natural hierarchical structure which can

be useful in the embedded systems. In [1], the authors described an interconnect architecture based on a butterfly fat tree for a networked SoC, as well as the design of required switches and addressing mechanisms. However, the scalability of current proposed solutions, including honeycomb structure [1], and butterfly fat tree structure, is either insufficient or unexplored.

2.2 Buffering Strategy

The buffering strategy determines the location of buffers inside the router. We distinguish output and input queuing. Output queuing has the best performance among the buffering strategies. However, the interconnection will make the router wire dominated and expensive already for small values of number of inputs. In input queuing the queues are at the input of the router. A scheduler determines at which times which queues are connected to which output ports, such that no contention occurs [11].

Another significant parameter is the dimension of the queue which influences directly packet latency and the switch area. It is very important to minimize the amount of buffering space. The buffer size and location are the main aspects discussed. In [8] the authors present an algorithm which sizes the (input) buffers in a mesh-type NoC, on basis of the traffic characteristics of a given application. For three audio/video benchmarks it was shown how such intelligent buffer allocation resulted in about 85% savings in buffering resources, in a comparison with uniform buffer sizes, without any reduction in performance.

2.3 Switching Mode, Contention Resolution and Network Flow Control

We identify three important issues in the design of the router network architecture. These are: the *switching mode*, *contention resolution*, and *network flow control*.

The *switching mode* of a network specifies how data and control are related. We distinguish *circuit switching* and *packet switching*. In *circuit switching* data and control are separated. The control is provided to the network to *set up* a *connection*. This results in a *circuit* over which all subsequent data of the connection is transported. In *packet switching* data is divided into *packets* and every packet is composed of a control part, the *header*, and a data part, the *payload*. Network routers inspect, and possibly modify, the headers of incoming packets to switch the packet to the appropriate output port.

When a router attempts to send multiple data items over the same link at the same time *contention* is said to occur. As only one data item can be sent over a link at any point in time, a selection among the contending data must be made; this process is called contention resolution [11].

In circuit switching, contention resolution takes place at set up at the granularity of connections, so that data sent over different connections do not conflict. Thus, there is no contention during data transport, and time-related guarantees can be given.

In packet switching contention resolution takes place at the granularity of individual packets. Because packet arrival cannot be predicted contention can not be avoided. It is resolved dynamically by scheduling in which data items are sent in turn.

Network flow control, also called *routing mode*, addresses the limited amount of buffering in routers and data acceptance between routers. Many recent multi-computer networks use cut-through or wormhole routing, which is a technique to reduce message latency by pipelining transmission over the channels along a message's route [11].

In wormhole routing, each packet is further split into a set of units named flits (flow control unit), which is the smallest unit over which is performed the flow control. The flit size varies between the packet size and channel width according to the topology, architecture and protocol of NoCs. Each flit also consists of control information and data. Wormhole routing reduces the store-and-forward latencies of packet transportation. Only the header flit needs to be routed. If it goes through a switch successfully, other flits just follow it without any more routing. If the header flit is blocked, the other flits stop "in place". Wormhole routing requires the least buffering (buffer flits instead of packets) and also allows low-latency communication. However, it is more sensitive to deadlock and generally results in lower link utilization than virtual cut-through routing.

2.4 Arbitration and Flit Size

The router's internal policies for queuing, arbitration, and flow control affect network performance and implementation complexity by coordinating resource sharing amongst competing packets. When several queues vie for a resource, the router invokes an arbitration policy, such as round-robin or a priority based scheme, to select the winner. The arbitration policy may differentiate between application traffic classes to assign priority to more urgent packets. Closely tied to both queuing and arbitration is flow control, which affects latency and throughput by limiting the rate at which packets travel through the network [13].

To avoid alignment problems, the block size (B words) is a multiple of the flit size (F words, $B = -\ell F$) with $-\ell$ being constant. We prefer a small $-\ell$ to decrease the store-and-forward delay and reduce the buffer size for guaranteed traffic, and a small F for fine-grained switching and better statistical multiplexing. The router architecture contains a data path and a control path. The data path maximises throughput for high link utilisation, and the control path maximises the rate of scheduling and switching. They can be designed and optimised independently [10].

2.5 Quality of Service (QoS)

Quality of service [8, 9] offered by the network has lately received a lot of attention and can be classified in two main modes: *best effort* (no QoS) and *guaranteed traffic* (QoS) that can be implemented either statically, using virtual circuit

fixed at design time, dynamically, using handshake protocols at runtime.

Other NoC parameters are discussed such as packet length, network interface VCI, and power consumption [9].

3. THE ROUTER BLOCK DESIGN

3.1 Design Approach

The used design methodology is based on VHDL as a description language at RTL level. This description is simulated until obtaining the expected behaviour. Simulation tool uses, in addition to VHDL description, a test file containing the system input stimuli in order to visualize the output behaviour. This description with a *HDL* (Hardware Description Language) is synthesized using existent synthesis tool allowing passage to the next abstraction level until reaching either integration in an *ASIC* or implementation in a *FPGA*. Simulation of the description is required before synthesis at every abstraction level.

Automatic synthesis (Synopsys Design Compiler) has been applied to produce a standard cell netlist. The layout was carried out in a standard cell fashion with a semi-automatic tool environment based on the Cadence tools. A Clock tree synthesis approach was followed in order to minimize the clock skew. The clock buffer tree was generated, equalizing perfectly all different clock branches and sub-branches. To meet all the timing constraints, in-place optimization was used using *SDF* back-annotation.

3.2 Network Structure Model

A Network-on-Chip is composed by a set of routers and point-to-point links interconnecting routers in a structured way. Each router has a set of ports that are used to connect routers with its neighbour's routers and with the processing cores of the system (i.e. processors, DSPs, memories, and others).

The ports used to connect the processing cores are named local ports or terminals. A pair of parallel opposite channels composes each interconnecting link.

A NoC (Figure 1) can be described by its topology and by the strategies used for routing, flow control, switching, arbitration and buffering. The network topology is the arrangement of nodes and channels into a graph. Routing determines how a message chooses a path in this graph, while flow control deals with the allocation of channels and buffers to a message as it traverses this path.

Switching is the mechanism that removes data from an input channel and places it on an output channel, while arbitration is responsible to schedule the use of channels and buffers by the messages. Finally, buffering defines the approach used to store messages while the router arbitration circuits cannot schedule them.

These routers are addressable for communication among processors. The network uses a deterministic routing algorithm in the form of a lookup table inside each router for routing to the neighbouring node. Although flow control is not supported, a deterministic routing approach significantly reduces hardware complexity and overhead.

Reconfiguration of the network topology and placement of processing units only requires a modification to the routing table. We can reconfigure internal buffer size of each router, and in this way, trade area for speed. It also allows for more efficient use of routing resources, which is an important design factor in embedded system design.

Routing is performed over fixed shortest paths, employing a symmetric X-Y discipline whereby each packet is routed first in an "X" direction and then along the perpendicular dimension or vice versa. This scheme leads to a simple, cost-effective router implementation. Network traffic is thus distributed non-uniformly over the mesh links, but each link's bandwidth is adjusted to its expected load, achieving an approximately equal level of link utilization across the chip.

Links are uni-directional and each link is composed by 32-bit wide channels (one for each direction). In each channel 32 bits are reserved for data. Each router has five ports named *L* (Local), *N* (North), *S* (South), *E* (East) and *W* (West). Each port has two channels, one for packets incoming into the router and another for packets outgoing from the router.

Internal channels are 16-bit wide. The flow control bits and the data words are stored in the queues of routers. Other signals are used to control the internal flow inside the routers.

In a wormhole switched network, each packet is split in a set of units named flits (*fl*ow control un*it*). A flit is the smallest unit over which is performed the flow control. It can be as long as a packet or as short as the channel width account. Buffering is implemented by means of input queues. Each input channel in the router has a 5-flit FIFO buffer to store packets while they cannot be forwarded to an output channel.

Routers connect to up to five links, designed for planar interconnect to four torus/mesh neighbours and to one chip module. Each link comprises an input port and an output port. The router forwards packets from input ports to output ports.

Data is received in flits. Every arriving flit is first stored in an input buffer. On the first flit of a packet, the router determines to which output port that packet is destined. The router then schedules the transmission for each flit on the appropriate output port.

In the modeled NoC, a flit is equal to the internal channel width (16-bit), because the flow control signals are not taken into small buffers are allocated to each service level, capable of storing only a few flits. The routing algorithm is invoked when the first flit of a packet is received. The algorithm uses a simple routing function. For instance, relative routing is employed for XY routing. Routing information per each service level per each input port is stored in the Current Routing Table, until the tail flit of the packet is received, processed and delivered.

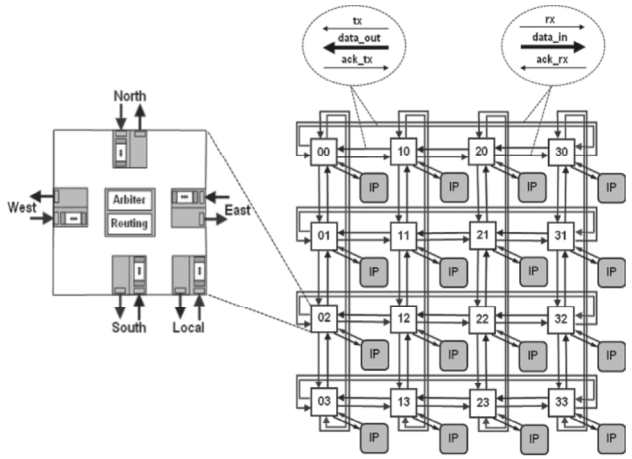

Figure 1: Router and NoC Representation

When a flit is forwarded from an input to an output port, one buffer becomes available and a *buffer-credit* is sent back to the previous router on separate out-of-band wires.

Each output port of a router is connected to an input port of a next router via a communication link. The output port maintains the number of available flit slots per each service level in the buffer of the next input port. These numbers are stored in the *Next Buffer State* table. The number is decremented upon transmitting a flit and incremented upon receiving a buffer-credit from the next router. When a space is available, the output port schedules transmission of flits that are buffered at the input ports and waiting for transmission through that output port, as detailed below.

We now turn to the mechanics of flit transfer inside the router. Flits are buffered at the input ports, awaiting transmission by the output ports. Flit routing is resolved upon arrival of the first flit of a packet and the output port number is stored in Current Routing Table for the pending flit per each input port and per each service level. Each output port schedules transmission of the flits according to the availability of buffers in the next router, the priority (namely service level) of the pending flits, and the round-robin ordering of flits within the same service level.

3.3 Router Architecture

The main objective of an on-chip switch is to provide correct packet transfer between different IPs cores. The NoC proposed, supposes that each switch has a bidirectional ports related to the neighbours switches and to the IP core. NoC characteristics are chosen in order to facilitate implementation. Recent packet-switching trends show that wormhole switching is the solution of choice for NoCs.

This scheme divides packets into fixed-length flow control units (flits), with I/O buffers storing only a few flits. Thus, unlike most other schemes, this design style minimizes the buffer space in the switches, and the switches used in a wormhole technique can be small and compact. The data packet size used is 32 words, with a byte in order to achieve routing. We have started with a 2D-mesh transformed to a 2D-torus topology. The used routing algorithm compares the actual switch address ($x_L y_L$) to the target switch address ($x_T y_T$) of the packet, stored in the header flit.

The exchange of data among the processors and storage cores is becoming an increasingly difficult task because of growing system size and non scalable global wire delay. To cope with these issues, we divide the end-to-end communication medium into multiple pipelined stages. In NoC architectures (Figure 2), the interswitch wire segments, along with the switch blocks, constitute a highly pipelined communication medium characterized by link pipelining, deeply pipelined switches, and latency-insensitive component design.

Switch design also depends on the routing scheme adopted. The two broad categories of routing are deterministic and adaptive. Deterministic routing algorithms always provide the same path between a given source and destination pair. Adaptive routing algorithms use information about routing traffic or to avoid the congested part of the network. In a deterministic routing scheme (used in this work), switches can be fast and compact.

The four internal buses (VC) can avoid deadlocks in a two dimensional Torus/Mesh network. In an output port, the buses selector decides which input bus will supply data to the header decoder by application of a priority algorithm. It consists of a priority multiplexer unit, four VCs and a register of 16 bits as shown in Figure 2.

Flow control deals with the allocation of channels and buffers to a packet as it travels along a path through the network. A resource collision occurs when a packet cannot be processed.

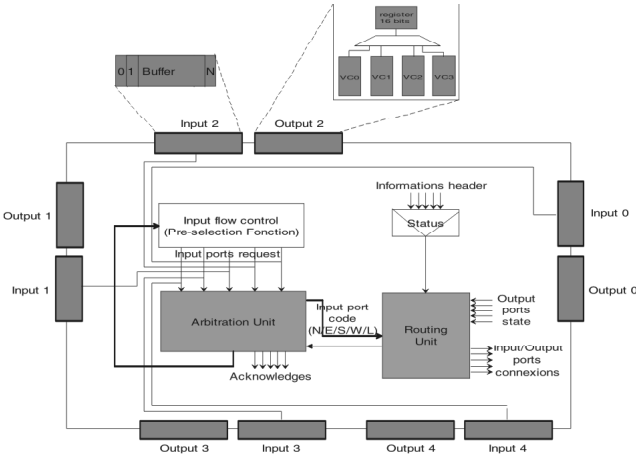

Figure 2: The Structure of the Proposed Router

Whether the packet is dropped, blocked, buffered, or rerouted through another channel depends on the flow control policy. A good flow control policy should avoid channel congestion while reducing the network latency.

The allocation of channels and their associated buffers to packets can be viewed from two perspectives. The routing algorithm determines which output channel is selected for a

packet arriving on a given input channel. Therefore, routing can be referred to as the output selection policy. Since an outgoing channel can be requested by packets arriving on many different input channels. An input selection policy is needed to determine which packet may use the output channel. Possible input selection policies include round, fixed channel priority, and first come first served. The input selection policy affects the fairness of routing algorithms.

An arbiter of a particular unit router (hE, hW, hN, hS, hL) receives requests from the header decoders of the other four unit routers and assigns an available output ports to one of the requesting header decoders using a centric (or round robin) arbitration mechanism.

In the current implementation, two types of physical data links have been used. External, are 32 bits wide for data that is used for the transmission of the packets between routers with its neighbour's routers and with the IPs processing cores of the system by a pair of parallel opposite channels. Inside the switch, buses width is fixed to 16 bits.

Packets are 32 words long (i.e., 64 bytes), including the flits header which represents the number of data words in the packet, control of errors, source and destination address, i.e., the (x_L y_L) and (x_T y_T) offsets in a 2D Torus/Mesh.

The routing decision unit associated to each input queue selects the output port taking into account the information in the packet header, the available local output ports and the status of the neighbour's buses (VC). This status is known through external signals which are processed by the arbiter. If the selected output port and buses are available, the routing decision unit updates the header flits so that the offset is decremented, and sent first. This decrement operation is carried out simultaneously for x and y offsets, and in parallel with the unit arbitration. However, only one of the modified offsets will be forwarded as the first flit according to the selected output.

The routing decision needs five cycles as they examine both header flits.

4. EXPERIMENT

4.1 Simulation Results

The developed switch can establish only one connection at the same time. However, a single switch can simultaneously handle up to five connections as illustrated by the simulation results presented in figure 3.

For example, three packets coming respectively from port 0 (Est), port 1 (West) and port 4 (Local) can be forwarded to the same output port (example port 3 (South)). The switch can simultaneously be requested to establish one of three connections as shown in figure 4.

Arbitration is used to grant access to an output port when one or more input ports simultaneously require a connection. A static arbitration scheme is used. The algorithm of arbitration is executed by using a fixed priority (by set of priorities).

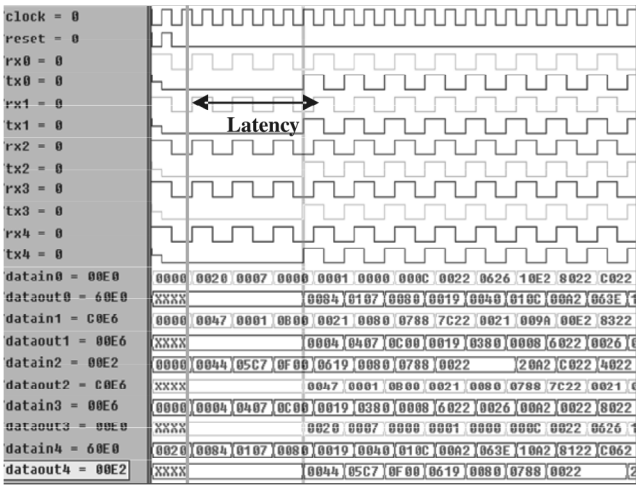

Figure 3: Establishment of Five Simultaneously Active Connections.

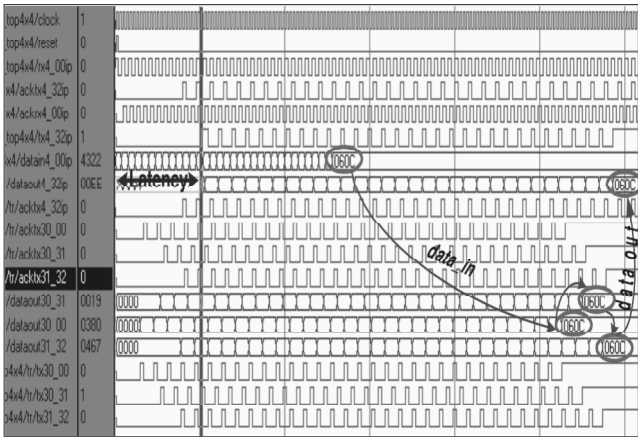

Figure 4: Simulation of a Packet Transmission from Switch 00 to Switch 32 in 2D Torus/Mesh Topology

The priority of a port is a function of the requests of the other ports having a granted request of the routing. The request processing requires five clock cycles.

The packet transmission in a 4x4 of this NoC was validated by a functional simulation. Figure 3 illustrates the transmission of this packet from switch 00 to switch 32. In fact, the input and output interface behaviours of switches 30 and 31 are shown in the simulation results.

4.2 FPGA Implementation

A switch infrastructure prototype, implemented in an FPGA occupies 87% of the device area and can be clocked at 127 MHz. To investigate the feasibility of the concept presented here, we have implemented a switch on an FPGA Virtex IIV500fg456. Table 1 presents the results and the percentages of used resources.

Table 1
Results and % of the Switch Synthesis on FPGA

Resources	*Used*	*Accessible*	*% of use*
IOs	182	264	68%
FGs	5382	6144	87%
CLB Slices	2691	3072	87%
Dffs	3064	6936	44%

4.3 ASIC Implementation

We have designed the entire switch using the *Synopsys V-2004.06-SP1* synthesis tool. Then, we mapped our design into 0.35 µm (4 metal layers) technology from AMS. Although the obtained results for area and time are estimated by *Synopsys* in a pessimistic way, they provide us with values very close to the physical domain.

Table 2 shows both delay and area for each one of the switch blocks. As the design is pipelined, the block in the critical path with highest delay determines the maximum clock frequency of the device. In this particular case it is the control logic unit (routing and arbitration units), due to its arbitration complexity, which imposes a cycle time of 0.5 ns. For the global port block (reception and emission modules) we cover a cycle time of 0.35 ns, taking into account the impact that queue depth has on its management time.

4.4 Latency and Performance

We fixed a number of 32 flits in a packet crossing the network of the source to the target. The flit processing time is two clock cycle. The operating frequency is 127MHz allowing a theoretical peak performance of:

127 MHz /2 * 5 ports * 32 bits= 10 Gbits/s

It is noticed that latency varies neither according to the size of the buffers nor according to the size of the packets, but only according to the number of intermediate switches.

The topology of the network depends on the manner of connecting the switches between them with bi-directional ports.

In a mesh network, for example, the number of ports by switch depends on its position in the network as explained previously. The designed router, described in VHDL on RTL level, was simulated in the case of topologies 2D-mesh and 2D-torus (2x2), (3x3) and then (4x4).

The basic latency of a packet-switched message is:

$$latency = \sum_{i=1}^{n} R_i + P * 4$$

n is the number of switchs during the trajectory of communication (the source and target included), $\mathbf{R_i}$ is the time required by the routing algorithm for each switch (at least 6 cycle's clock), **P** is the size of packet. P is multiplied by 2 because each word requires 2 cycle's clock so that it is sent of one port to the other in the same switch. This number is multiplied by 4 because each word requires 4 cycle's clock so that it is sent of a switch to the other.

Following figure 5 shows the effect of the number of cores IPs (***N***) in the latency of NoC-based systems, without taking into account the impact of die size and number of cores in the clock cycle. Based on dimension of NoC and their topologies, the total latency to delivery $N.(N\text{-}1)$ messages in both networks is presented.

Table 2
Characteristics of the Router Modules

Module	*Area (mm^2)*	*Critical path (ns)*	*Power (mW)*
Buffer (6 flits)	0.0187	-	-
Control logic	0.0641	0.5	-
Global port East	0.3648	0.3518	-
Global port West	0.3648	0.3518	-
Global port North	0.3650	0.3519	-
Global port South	0.3648	0.3518	-
Global port Local	0.3568	0.351	-
Total	1.8803	0.63	486.8319

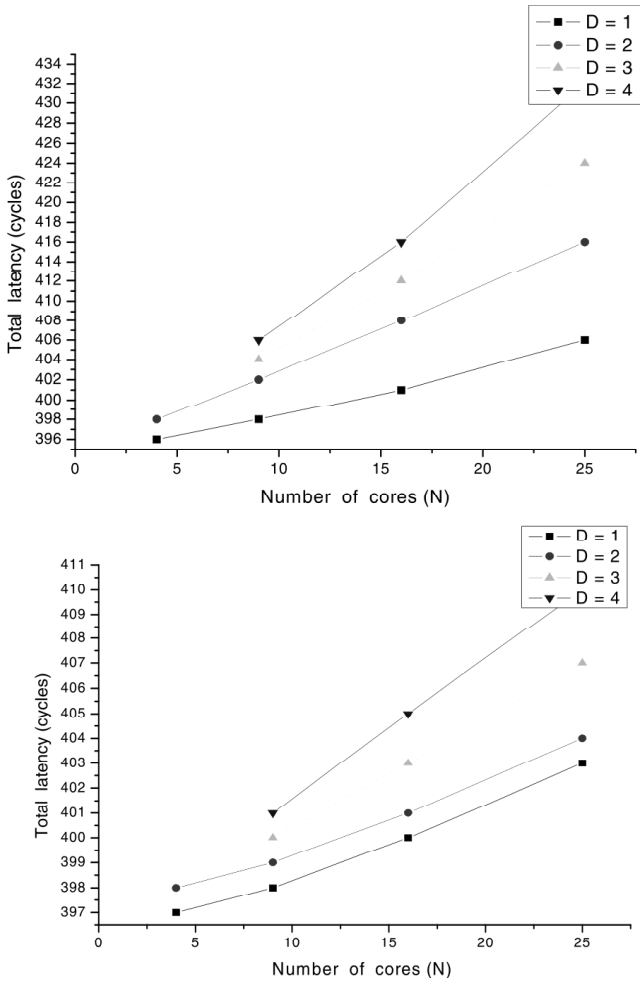

Figure 5: Latency for m=1 and m=4, load=N.(N-1)

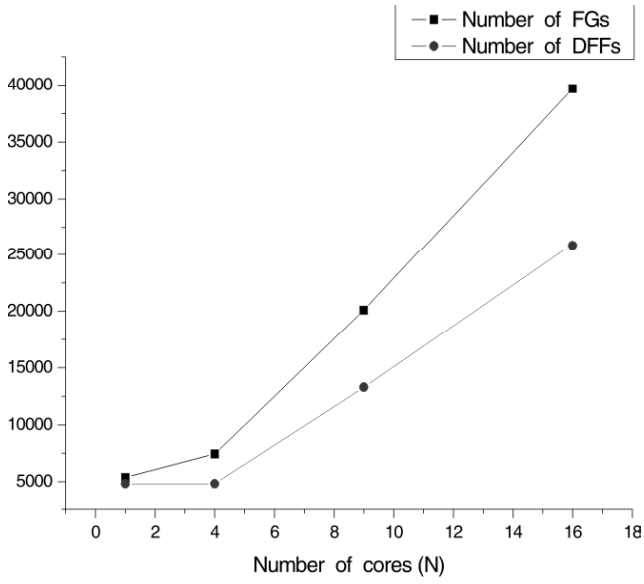

Figure 6: Variation of NoC Surface According to a Number of Switches of Topology

Each message has **P**=1+*2m* 32-bit words on the NoC, due to the packet header and data. Each graph includes several curves for different communication locality patterns. As smaller the parameter ***D*** is, the greater is the locality. For instance, by making *D* equals 2, we consider that all the messages are exchanged among cores separated by two routers and a single inter-router link.

Figure 5 shows that for short messages and at the maximum workload, the NoC is effective when the locality is great (*D* small) or when the system grows (*N* increases). Although smaller workload and if the message size is increased, the parallelism and the pipeline of the NoC produce a performance enhancement.

The time spent to deliver packets grows linearly with the number of intermediate switches (Figure 6). This happens because each switch spends some clock cycles to execute the arbitration and switching algorithms.

For buffers of a small number of flits, the delivery time does not vary. If the buffer is too small, the switch cannot receive new *flits* until the destination port is chosen. Therefore, the minimum buffer size must be equal to the number of write operations that can be executed during the arbitration and switching algorithms execution.

In the case of a NoC (4x4), these algorithms use 6 clock cycles and each write operation takes 2 clock cycles. The minimum size of the buffer is then of 6 flits.

4.5 Placement and Routing

Cadence SoC Encounter integrates the solution of Synthesis/ Placement and Route (SP&R) with new functions and the technology of Silicon Perspective Corporation (SPC). It provides a hierarchical solution RTL-GDSII complete. We chose SoC Encounter as a tool of placement, routing and installation hierarchical us to decrease the time of design of our SoC application.

The complete communication infrastructure is made in the topology 2D-Torus/Mesh (3x3) by inter-connecting 9 routers with 9 modules (different MACs). These modules are responsible of data computing.

Design of the 2D-Torus/Mesh (3x3) topology occupies 11 % of the device area. If large components are placed on the device, they will cover a large set of routers, thus reducing the total area used by the routers. The router has also been implemented as described previously.

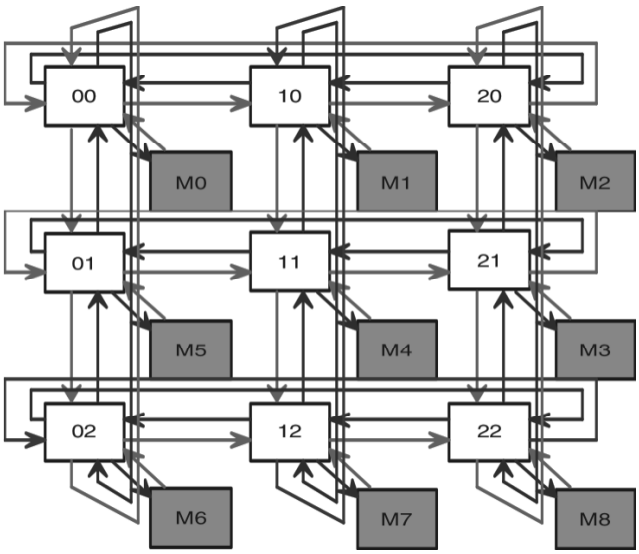

Figure 7: Representation of the SoC 3x3 Application in the Case of a 2D-Torus/Mesh 3x3 Network Topology

Each router occupies less than 0.9 % of the system device area and has a latency of 6 cycle's clock corresponding to a frequency of 127 MHz. Because the maximum number of routers to traverse is 5, two components running on the device with a frequency of 100 MHz can send and receive the packets without delay in their execution, if the network is free.

5. CONCLUSIONS

In this paper, we are interested by torus and mesh topologies, which seem to dominate the literature of the networks on a chip. We studied several key points such as memorization, routing algorithm and arbitration. A VHDL based methodology is used to implement a switch for data transfer from an IP source to an IP destination.

The architecture was verified in several NoC cases such as 2x2, 3x3 and 4x4 architecture.

REFERENCES

[1] E. Carara, A. Mello, and F. Moraes, "Communication Models in Network on Chip". In: *18th IEEE/IFIP International Workshop on Rapid System Prototyping (RSP '07)*, 2007, pp. 57-60.

[2] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," *INTEGRATION, The VLSI Journal*, Vol. 38, No. 1, 2004, pp. 69–93.

[3] Kumar S., *et al.* "A Network on Chip Architecture and Design Methodology". In: *IEEE Computer Society Annual Symposium on VLSI*. (ISVLSI'02), Apr. 2002, pp. 105-112.

[4] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm", *IEEE Computer*, Jan. 2002, pp. 70-78.

[5] S. Murali and G. De Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures", *Design, Automation and Test in Europe Conference and Exhibition*, Vol. 2, 2004, pp. 896-901.

[6] D. Ching, P. Schaumont and I. Verbauwhede, "Integrated Modeling and Generation of a Reconfigurable Network-on-Chip," *18th International Parallel and Distributed Processing Symposium*, 2004, pp. 139-145.

[7] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist, "Network on Chip: An Architecture for Billion Transistor Area". In *Proc. IEEE NorChip Conf.*, 2000, pp. 3531-3538.

[8] G. De Micheli, "Reliable Communication in Systems on Chip," in *Proc. DAC*, 2004, pp. 77.

[9] Jingcao Hu, R. Marculescu: "Application-Specific Buffer Space Allocation for Networks-on-Chip Router Design". *Proceedings of the 2004 IEEE/ACM International Conference on Computer-Aided Design*, November 2004, pp. 354-361.

[10] J. Henkel, W. Wolf, and S. Chakradhar: "On-chip Networks: A Scalable, Communication-Centric Embedded System Design Paradigm", in *Proc. 17th Int'l Conf. VLSI Design*, 2004, pp. 845-851.

[11] D. Bertozzi, et al.: "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 2, February 2005, pp. 113-129.

[12] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage and E. aterlander: "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip". *IEEE Proc.-Comput. Digit. Tech.*, Vol. 150, No. 5, 294, September 2003, pp. 294-302.

[13] J. Rexford, W. Feng, J. Dolter, and K. Shin: "PP-MESS-SIM: A Flexible and Extensible Simulator for Evaluating Multicomputer Networks". *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 1, January 1997, pp. 25-40.