SCAC-Net: Reconfigurable Interconnection Network in SCAC Massively parallel SoC

Hana Krichene, Mouna Baklouti, Mohamed Abid, Philippe Marquet, Jean-Luc Dekeyser, Samy Meftali

SCAC-Net: Reconfigurable Interconnection Network in SCAC Massively parallel SoC

Hana Krichene[1,2], Mouna Baklouti[2], Mohamed Abid[2], Philippe Marquet[1], Jean-Luc Dekeyser[1], Samy Meftali[1]
[1]Univ.Lille, CNRS, Centrale Lille - UMR 9189 - CRIStAL
Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
[2]ENIS - CES lab, University of Sfax, Tunisia

***Abstract*—Parallel communication plays a critical role in massively parallel systems, especially in distributed memory systems executing parallel programs on shared data. Therefore, integrating an interconnection network in these systems becomes essential to ensure data inter-nodes exchange. Choose the most effective communication structure must meet certain criteria: speed, size and power consumption. Indeed, the communication phase should be as fast as possible to avoid compromising parallel computing, using small and low power consumption modules to facilitate the interconnection network extensibility in a scalable system. To meet these criteria and based on a module reuse methodology, we chose to integrate a reconfigurable SCAC-Net interconnection network to communicate data in SCAC Massively parallel SoC. This paper presents the detailed hardware implementation and discusses the performance evaluation of the proposed reconfigurable SCAC-Net network.**

I. Introduction

Modern embedded multimedia applications that include data-parallel task, require high performance on-chip systems with high capacity of integration. To address these data-parallel applications, massively parallel SoCs become widely used. When processing nodes execute in these systems, some data manipulations have to be performed in the interconnection network. If the number of processing nodes increases, the communication control process becomes a bottleneck for system performance. To address these problems, we propose a reconfigurable and flexible interconnection Network-on-Chip (NoC) that allows large scale regular synchronous communication. Our NoC, called SCAC-Net network, is designed and integrated into SCAC architecture [1].

The objective of this work is to define SCAC-Net architecture based on reused IP blocks. The appropriate configuration is defined using generic parameters to establish the needed design to execute a given application. SCAC-Net network based on SCAC is compared with some other interconnection networks on massively parallel architectures through FIR filter and parallel summing applications to prove its high efficiency in terms of area cost, bandwidth and latency.

The remainder of this paper is organized as follows. Section 2 presents some existing interconnection networks. Section 3 introduces the SCAC design. Section 4 describes the implementation of SCAC-Net network and its functionality. Finally, Section 5 shows SCAC-Net experiment results. Section 6 summarizes our contribution.

II. Related works

Several interconnection networks have been proposed in the literature to meet different needs of massively parallel system scalability, flexibility, consumption, and reuse [2]. In particular, the two-dimensional mesh networks were widely used in the conventional SIMD machines, allowing regular inter-PE data communications. We note the fully connected array in SIMD Imagine system [3] and the BP-mesh hierarchical interconnection network in CMPs architecture [4]. They are high scalable networks, but they are costly in terms of interconnection delay and power consumption. The recent SIMD systems tend to use reconfigurable interconnection network. Among which, we find the RMESH [5] network, where the reconfiguration is achieved by the PEs. Each PE controls its local switch element (SE), independently of the others PEs in the system. This autonomous decision spends several interconnection delays when synchronisation is required. We also find, the reconfigurable neighbourhood interconnection network [6] in MppSoC architecture. This network leads its limit with distant communication, which needs several external links that consume a lot of chip area with a huge number of PEs. The irregular approach, proposed by NoC [7] in SPINNAKER GALS system, makes the control tasks difficult to achieve, requiring specific rooting algorithm.

Our work differs from previous works in that, we propose a reconfigurable and modular SCAC-Net NoC, which takes advantage of the regular and synchronous communication and provides both efficiency and flexibility. This SCAC-Net is an on-chip structure using parametric IP blocks for rapid system reconfiguration. In the section below, we present our SCAC massively parallel system, where we will integrate our proposed SCAC-Net network.

III. SCAC architecture

A SCAC system [1] is composed of a First-Level Control Unit (FLCU) connected to its sequential instructions memory, called FLCU-memory, and a grid of Second-Level Control Units (SLCUs). Each SLCU is connected to a cluster of Computation Elements (CEs). The set (SLCU + CEs) is called a parallel execution *node*. Each CE is connected to its local instructions and data memories, called *Mi* memory. The parameter *i* is relative to the CE number. The FLCU and the

SLCUs grid are connected via a bus with a single hierarchical level. Fig. 1 shows the overview of the SCAC architecture.

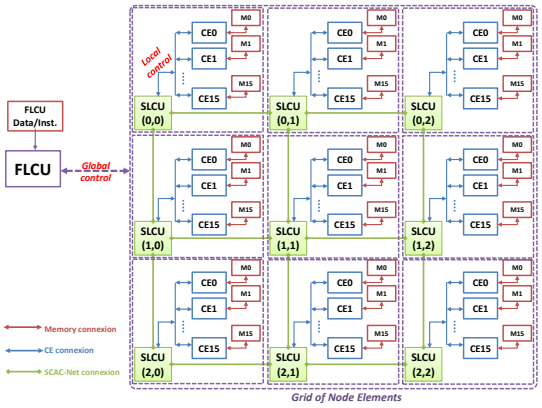

Fig. 1. SCAC architecture

The SCAC architecture is characterized by hierarchical control structure distributed on two levels [8], in order to separate the control functions. The SCAC execution model is based on *Asynchronous Computation* to allow autonomous processing, and *Synchronous Communication* to ensure simple data transfer without compromising the computation.

Because the inter-node communications are important in the intensive processing algorithms and must be fast in order to ensure good performance, we define a reconfigurable neighbourhood interconnection network, called SCAC-Net. This network is clocked synchronously with the SLCUs and respectively with the CEs. Below, we will detail the SCAC-Net network design.

IV. SCAC-Net design

To ensure the flexibility of SCAC-Net network, we have defined a communication structure based on reused IP blocks.

This section present the hardware implementation and the assembly programming to allow regular and synchronous communication.

A. SCAC-Net overview

To ensure synchronous communication in SCAC system, we define an on-chip regular interconnection network inspired from the network used in MasPar [9] machine: called SCAC-Net Network-on-Chip.

In SCAC system, all SLCUs have the same communication direction. In fact, each SLCU can simultaneously send a data to the northern neighbour and can receive another data from its southern neighbour. The SCAC-Net uses a bit-state signal to identify nodes that participate in communication. Inactive SLCUs can be used as pipeline stages to achieve distant communication.

The SCAC-Net network directly connects each CE with its 8 nearest neighbours in bidirectional ports, through the SLCU-COM module in SLCU component [1]. This module is composed of COM-Control and SLCU-router sub-modules.

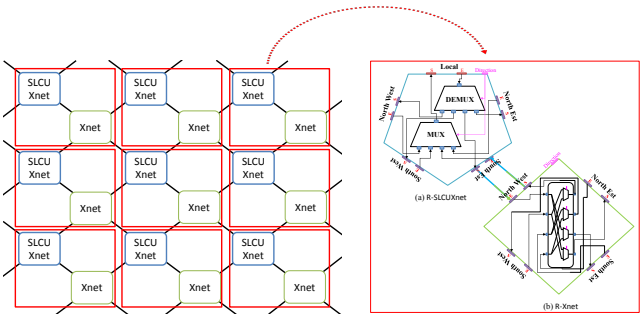

Fig. 2. SCAC-Net network

1) COM-Control: is the communication controller. It is implemented as a Mealy state machine, based on four states: Idle, Read, transfer and Write. The control phase begins with the "Idle" state. When the COM-Control receives the communication micro-instructions sent by the SLCU instructions decoder, the control phase moves to the "Read" state. The COM-Control decodes this micro-instruction and sends the data and the activity bit to the SLCU-router, in order to activate the communication ports in a given direction. This step takes a single clock cycle to be executed. If the communication distance is > 0, then the next state is "Transfer" and the control phase loops until the distance counter reaches 0. This step takes $(Dist \times cycle(s))$ to be executed. When distance is equal to 0, the control phase moves to the "Write" state. The COM-Control stores the received data from SCAC-Net network in the R_COM register, to be used by the local CE. This step takes one clock cycle.

2) COM-Router: is the communication router. It is composed of couple of routers (R-SLCUXnet, R-Xnet), as detailed in fig. 2. Each router of this couple can take 4 directions (North-East, South-East, North-West or South-West), allowing together the data transfer in 8 directions, as detailed in table I.

TABLE I
SCAC-Net directions

Direction		Code	R-SLCUXnet		R-Xnet	
↖	North-West	0	0	↖	0	↖
↑	North	1	0	↖	1	↗
↗	North-East	2	1	↗	1	↗
→	East	3	1	↗	2	↘
↘	South-East	4	2	↘	2	↘
↓	South	5	2	↘	3	↙
↙	South-West	6	3	↙	3	↙
←	West	7	3	↙	0	↖

Based on a SCAC-Net network, all the communications occur in the same direction. In fact, on a same output port there will never be a messages congestion, which simplify the routing elements design:

- (a) R-SLCUXnet: is a couple of 4:1 mux/demux, to perform synchronous communication. It has 4 inputs, 4 outputs and two arbiters, which manage the priorities between the different communication requests;

- (b) R-Xnet: is a 4x4 crossbar module. It connects non-occupied ports according to a given direction. It has 4 data inputs, 4 data outputs and 4 arbiters to manage communications priorities.

The routing elements in SCAC-Net are connected, forming an X pattern as shown in fig. 2. Given the determinism of the routing system, router architecture has a simple structure and small size, consuming 128 LUTs and 49 registers of the FPGA Virtex 6 ML605 ($< 1\%$).

SCAC-Net network uses various configurations according to the algorithms needs. Therefore, we define different bus sizes (1 bit, 4 bits or 16 bits) and different network topologies 1D (linear and ring) and 2D (mesh and torus). To choose a network configuration, the designer has to fix the generic parameters of topology and bus size. When the network parameters are selected, the SCAC-Net can be generated.

This data transfer through SCAC-Net occurs without conflicts and is achieved by SEND and RECEIVE instructions that allow all the nodes to communicate in the same direction and distance. These instructions will be detailed in the next sub-section.

B. SCAC-Net instructions

A given SCAC-Net communication allows all the execution nodes to communicate with their neighbours in a given direction at a given distance. Direction and distance are the same for all the nodes. These parameters are managed by the communication instructions (SEND or RECEIVE). The execution of these instructions requires three steps:

- **Open:** the routing elements are configured according to the direction to take, as presented in the table I. They activate two ports that validate the given direction. For SEND instruction, only the routers of the active nodes are configured, the others are idle. For RECEIVE instruction, all the nodes are active and their routers are configured.
- **Transfer:** the routers of the active nodes send their data in the SCAC-Net network.
- **Store:** Only active nodes store the received data. After a communication phase, all the routers ports must be closed.

V. Applications and experimental results

In this work, we have implemented two applications: FIR filter (using two algorithms) and summing example, written in FC16 assembly language [10]. The FC16 instructions are coded in VHDL packages to be easily integrated in the design bit-stream and then, be executed by the FPGA SCAC implementation. Different interconnection networks are evaluated and compared with the implemented SCAC-Net network.

A. FIR Filter

The digital Finite Impulse Response (FIR) [11] filter is used in digital signal processing to hide: noise, regularity, etc. It is defined by the equation (1):

$$y(n) = \sum_{k=0}^{N-1} h(k) \times x(n-k) \quad (1)$$

X and Y are, respectively, the input and the output signal. N is the filter order and $H(k)$ are the filter coefficients. n is the number of inputs/outputs data.

To evaluate SCAC-Net network performance, we have implemented the FIR filter with two methods:

1^{st} method: 2D configuration

For this configuration, the system is composed of a FLCU, a grid of (4×4) *execution nodes* and a torus SCAC-Net network. The FIR filter takes 16 $H(k)$ and 64 $X(n)$ inputs. The algorithm [12] will be defined by:

1) Data storage: each *PE(i,j)* local memory receives all $X(n)$ and $H(k)$ element, where $k = 4 \times i + j$.
2) Multiplication: of all inputs with local *H(k)*.
3) Communication: the first multiplication elements are first sent to *West* by all PE, then *North* by only last column PEs.
4) Addition: the received data are added to the second multiplication element in each *PE(i,j)* .
5) Repeat 4) until we have all results that will be stored in *PE(0,0)* memory.

To execute this algorithm, we need (64 × 2) communications. Multiplication and addition operations are performed in parallel.

2^{nd} method: 1D configuration

For this configuration, the system is composed of a FLCU, 16 *execution nodes* and a linear SCAC-Net network. The FIR filter takes 16 $H(k)$ parameters and 64 $X(n)$ inputs. All $H(k)$ are stored in FLCU and sent to *PE(i)* when they are needed in the algorithm. Only West-communication is needed to shift $X(n)$ input between execution nodes. The n outputs are calculated using multiplication and addition instructions, in the alternative way. To implement this algorithm, only 63 communications are required.

Experimental results

The experimental results in table II show the communication delay in FIR filter execution.

TABLE II
16-order FIR implementation

Inputs (n)	COM. execution time (cycle)			
	G-MPSoC (1)	G-MPSoC (2)	ESCA	re-SIMD on-chip
8	126	72	255	332
16	270	144	351	744
64	1134	576	1158	-

As expected, the SCAC architecture based on SCAC-Net network allows more rapid communication than both re-SIMD on-chip [13] and ESCA [12]. This can be explained by SCAC-Net features. In fact, the data transfer through the routing elements occurs simultaneously in a single clock cycle, because the implementation of the SCAC-Net network is based

on a combinatorial logic. In addition, the communication in the SCAC-Net network are synchronous, managed by communication instructions. At the same time, all SLCUs perform the same parallel communication instruction in the same direction and at the same distance. This feature facilitates the transfer of data without the need of buffers/FIFOs or complex routing algorithms to manage congestion and synchronization.

B. Parallel Summing

The parallel summing application [14] is implemented to show data transfer in SCAC-Net network, using communication instructions in different directions. A 2D configuration (4×4 nodes) is generated to facilitate the neighbourhood and distant communications.

In traditional massively parallel system, the parallel summing execution requires several clock cycles for communication phases. In particular, distant data transfers are achieved via external links [14], which congests the system.

Using simple router design and internal pipelined distant communication in SCAC-Net network, decrease the execution time and improve system scalability. The data transfer only costs d cycles: the source-destination distance.

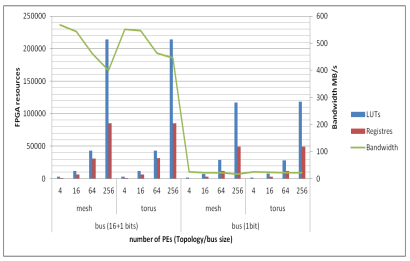

Fig. 3. Influence of bus sizes and topologies on system scalability

Fig. 3 presents synthesis and bandwidth results on different SCAC-Net topologies and bus sizes. We note a compromise between area and bandwidth. Indeed, the configuration of SCAC-Net with (16+1) bits buses gives efficient data transfer with large bandwidth, but it occupies a large chip area (2 times higher). In addition, if the number of the nodes is multiplied by a factor of 16, the bandwidth decreases by factor of 2 and the FPGA area increases by a factor of 4, which is an acceptable rate.

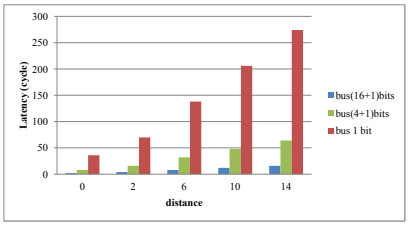

Fig. 4. Influence of buses size on communication delay

Fig. 4 shows the communication delay with the use of the different bus sizes. The more the bus size is large the more communication delays are decreased. However, with small bus size (1-bit) we can increase the system scalability.

Through these experimental results, we note the flexibility of SCAC-Net network. In fact, the designer can select the network configuration according to the application needs.

VI. Conclusion

In this paper, we present a new NoC for massively parallel systems. This network is mostly used to manage regular synchronous communications present in most data parallel applications. It is a reconfigurable and flexible network, using various topologies and various bus sizes needed to satisfy the requirements of the data-parallel algorithms. Communication instructions are defined to manage the data transfer among the nodes and the detailed hardware implementation in Xilinx virtex6 board has been discussed. Based on several SCAC-Net design configurations, we evaluate the performance of our network through the parallel Summing and FIR filter applications. SCAC-Net is compared with some others interconnection networks on existing massively parallel architectures to prove its high efficiency in term of latency, area cost and bandwidth.

References

[1] H. Krichene *et al.*, "G-mpsoc: Generic massively parallel architecture on fpga," *WSEAS Transactions on circuits and systems*, vol. 14, pp. 457 – 468, 2015.

[2] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," vol. 38, 2006.

[3] B. Khailany *et al.*, "Imagine: Media processing with streams," vol. 21, Mar. 2001, p. 3546.

[4] D. Wu *et al.*, "A high efficient on-chip interconnection network in SIMD CMPs," in *Algorithms and Architectures for Parallel Processing*, May 2010, pp. 149–162.

[5] H. Giefers and M. Platzner, "An FPGA-based reconfigurable mesh many-core," *IEEE Transactions on Computers*, vol. 63, no. 2919 - 2932, 2014.

[6] M. Baklouti *et al.*, *Reconfigurable Communication Networks in a Parametric SIMD Parallel System on Chip*. Springer Berlin Heidelberg, 2010, vol. 5992, ch. Reconfigurable Computing: Architectures, Tools and Applications, pp. 110–121.

[7] G. Liu *et al.*, "Network traffic exploration on a many-core computing platform: Spinnaker real-time traffic visualiser," in *11^{th} Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*. Glasgow, United Kingdom: IEEE, 2015, pp. 228 – 231.

[8] H. Krichene *et al.*, "master-slave control structure for massively parallel system on chip," in *Euromicro Conference on Digital System Design (DSD2013)*, Santander, Spain, Sep. 2013, pp. 917–924.

[9] E. Zehendner, "Simulating systolic arrays on maspar machines," in *the 23rd EUROMICRO Conference*, Budapest, Hungary, 1997, pp. 394–401.

[10] R. Haskell and D. Hanna, "A VHDL forth core for FPGAs," in *Microprocessors and Microsystems*, Apr. 2004, pp. 115–125.

[11] S. M. Kuo and W. S. Gan, *Digital Signal Processors: Architectures, Implementations, and Application*. Prentice Hall, 2005.

[12] P. Chen *et al.*, "Parallel algorithms for FIR computation mapped to ESCA architecture," in *International Conference on Information Engineering (ICIE)*, Beidaihe, Hebei, Aug. 2010, pp. 123–126.

[13] J. Andersson *et al.*, "A reconfigurable SIMD architecture on-chip," in *Master Thesis in Computer System Engineering - Technical Report*, School of Information Science, Computer and Electrical Engineering - Halmstad University, Jan. 2006.

[14] M. Leclercq and P. Aquilanti, "Réseau X-Net pour MPPSOC," Master1 - Computer Science, university of Lille1, Lille, France, Dec. 2006.